Bayesian Annotation Networks for Complex Sequence Analysis

Henning Christiansen, Christian Theil Have, Ole Torp Lassen and Matthieu Petit

Research group PLIS: Programming, Logic and Intelligent Systems
Department of Communication, Business and Information Technologies
Roskilde University, P.O.Box 260, DK-4000 Roskilde, Denmark
E-mail: {henning,cth,otl,petit}@ruc.dk

Abstract

Probabilistic models that associate annotations to sequential data are widely used in computational biology and a range of other applications. Models integrating with logic programs provide, furthermore, for sophistication and generality, at the cost of potentially very high computational complexity. A methodology is proposed for modularization of such models into sub-models, each representing a particular interpretation of the input data to be analysed. Their composition forms, in a natural way, a Bayesian network, and we show how standard methods for prediction and training can be adapted for such composite models in an iterative way, obtaining reasonable complexity results. Our methodology can be implemented using the probabilistic-logic PRISM system, developed by Sato *et al*, in a way that allows for practical applications.

1998 ACM Subject Classification I.2.6 Learning

Keywords and phrases Probabilistic Logic Bayesian Sequence Analysis

Digital Object Identifier 10.4230/LIPIcs.ICLP.2011.220

1 Introduction

Analysis of DNA is an important example of a complex sequence annotation task which has motivated our approach. The sheer size of data instances and the degree of ambiguity in such tasks pose great challenges for efficient probabilistic analysis. Furthermore, most systems for DNA-analysis used in practice are implemented in low-level programming languages, optimized and tweaked for very specific procedures, thus leading to systems with an unclear semantics and lack of flexibility for the modeling part. A possible shift to using probabilistic-logic systems and languages provides obvious benefits in terms of clear semantics and flexibility, but also introduces potential problems concerning complexity and scalability. We present here a modular approach, in which complex probabilistic-logic models are defined in terms of separate sub-models, each representing a particular interpretation (or "signal") of the input data to be analyzed. The dependencies among the results of analyses performed by these sub-models are described in terms of edges in a Bayesian network. This allows for an implementation based on incremental application of standard methods for prediction and training, one sub-model at a time, thus possibly leading to acceptable complexity. We refer to such modularized models for sequence analysis as *Bayesian Annotation Networks*. We demonstrate an implementation based on PRISM [16], which is a probabilistic extension of Prolog.

Technical Communications of the 27th International Conference on Logic Programming (ICLP'11).
Editors: John P. Gallagher, Michael Gelfond; pp. 220–230

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2 Probabilistic Annotation Models

Probabilistic-logic models for sequence annotations will be presented in two steps, first the logical part, and then probabilities are added. Notice also, that we abstract away the details of any actual modeling language and the format of probability parameters.

▶ **Definition 1.** An *annotation program*, or just *a program*, is a logic program *prog*, that defines a set of atoms, each of the form:

$$prog(s, a, parents),$$

where

- s is called *the sequence*, and represents the data sequence to be annotated by the program.
- a is called an *output annotation*, and
- *parents* represents zero or more *conditioning annotations*.

The name "*parents*" anticipates the introduction of Bayesian Annotations Networks in section 3 below. They represent annotations produced by other sub-models, serving as conditions for the analysis associated with *prog*.

▶ **Definition 2.** A *probabilistic annotation model*

$$m = \langle prog, \theta \rangle$$

consists of a probabilistic annotation program *prog* and a *parameter* θ. The parameter is element of some data domain which is not specified further, but which gives rise to a well-defined conditional probability distribution for atoms $prog(s, a, parents)$ as follows:

$$P(a \mid s, parents, \theta)$$

The intuition is that θ that associates probabilities to the detailed choices made within *prog* to produce the output annotation a, given a specific sequence s and parent annotations. Notice that our framework captures also analyses that are not necessarily written in a probabilistic-logic language. Notice that our framework captures also analyses that are not necessarily written in a probabilistic-logic language.

▶ **Definition 3.** A *deterministic annotation model* is a program

$$prog(s, a, parents)$$

where, for specific sequence s^0 and $parents^0$, there exists exactly one output annotation a^0, i.e.,

$$P(a^0 \mid s^0, parents^0, \theta) = 1,$$

where θ, in this case, refers to an (empty) parameter which is ignored.

The empty parameter is included for uniformity of notation only. A deterministic annotation model with empty parents may represent an analysis provided by an external tool that, e.g., searches for similarities in a database of related sequence data.

3 Organizing Annotation Models as a Bayesian Network

Our overall idea for prediction is to evaluate one model at a time, fix its output annotation to a single "best" one which, then, is used as parent for subsequent analyses. This is very similar to the way forward analysis takes place in Bayesian networks, which we thus take as our central paradigm for putting sub-models together to a whole. A Bayesian Network (BN) is defined as a directed acyclic graph as follows [15].

- Its *nodes* are random variables.
- An *edge* from node A to node B indicates that B is directly dependent on A, and A is called a *parent* of B; the notation $parents(B)$ refers to the sequence of parent nodes of B.
- Each node A has an associated *conditional probability distribution*, *CPD*, $P(A \mid parents(A))$.

For many applications of BNs, the CPDs are given in the form of tables, but since the random variables in our case range over huge sets of alternative annotations, this is infeasible, and we use probabilistic models instead.

▶ **Definition 4.** A *Bayesian annotation network* (BAN) is a set of probabilistic annotation models $\{M_i \mid i = 1, \ldots, n\}$, with $M_i = \langle m_i(s, a_i, parents_i), \theta_i \rangle$, numbered in such a way that $parents_i \subseteq \{a_1, \ldots, a_{i-1}\}$.
The model M_n is a designated *top model*, and it is assumed that the parent relationship induces a path from any other M_i to M_n.

A BAN in itself is not a BN, but it induces a BN in the following way.

- Nodes are labelled a_i, $i = 1, \ldots, n$ and s.
- Whenever $a_j \in parents_i$, there is an edge from a_j to a_i, and there is an edge from s to any a_i.
- The CPD associated with a_i is given by the model M_i, i.e., $P(a_i \mid s, parents_i, \theta_i)$.

For ease of terminology, we refer to a suitable set of annotation programs as a BAN, when below, we talk about training a BAN, i.e., finding parameters such that it actually becomes a BAN, as per the present definition. When presenting a BAN as a graph, we typically leave out s and the n edges going out from it. When doing predictive inference below, the sequence is always fixed, so we can leave it out, assuming instead a particular BN for each sequence.

4 Predictive inference

Predictive inference refers here to the process of identifying a best proposal for top output annotation that characterizes a given sequence. The fundamental assumption when using probabilistic models is that quality of a solution is intimately coupled to its probability, in other words, we should be searching for a top output annotation with a relatively high probability, ideally the one with highest probability.

Below, we give first a precise, declarative characterization of the best top output annotation, and then an approximative calculation method which, under certain circumstances, may reduce computational complexity drastically. Examples and detailed arguments for this claim will be given later.

We assume a BAN $\{M_i \mid i = 1, \ldots, n\}$ with $M_i = \langle m_i(s, a_i, parents_i), \theta_i \rangle$ and a fixed sequence s^0 to be analysed. We use Θ to refer to the set of all parameters in the BAN, $\{\theta_1, \ldots, \theta_n\}$. Considering the BAN as an entire model, we can describe the best solution as follows.

$$ideal_n(s^0, \Theta) =_{\text{def}} \operatorname*{argmax}_{a_n} P(a_n \mid s^0, \Theta)$$

where the term inside the argmax can be unfolded as follows.

$$P(a_n \mid s^0, \Theta) = \sum_{\langle a_1, \ldots, a_{n-1} \rangle} P(a_1, \ldots, a_n \mid s^0, \Theta) \tag{1}$$

$$= \sum_{\langle a_1, \ldots, a_{n-1} \rangle} \prod_{i=1}^{n} P(a_i \mid s^0, parents_i, \theta_i) \tag{2}$$

Standard methods for reasoning in Bayesian networks, see, e.g., [6], is of very little use here due to the unmanageable size of the random variables' outcome spaces, which in practice are impossible to iterate over.

We are not aware of any reasonable way to reduce this formula, although we do not have a formal proof that this is not possible. Instead, we propose an approximative, iterative algorithm that fixes one particular best annotation $a_i = \mathit{approx}_i(s_0, \Theta)$ for each sub-model and applies it subsequently in the prediction of those a_j with $a_i \in \mathit{parents}(a_j)$.

$$\mathit{approx}_i(s^0, \Theta) = \operatorname*{argmax}_{a_i} P(a_i \mid s^0, \mathit{approx}_{\mathit{parents}_i}(s^0, \Theta), \Theta), i = 1, \ldots, n \tag{3}$$

where $\mathit{approx}_{\mathit{parents}_i}(s, \Theta)$, for some sequence s, stands for the sequence of parent annotations $\mathit{approx}_j(s, \Theta)$ for all $a_j \in \mathit{parents}_i$.

Specifically, we take $\mathit{approx}_n(s^0, \Theta)$ as an approximated value for $\mathit{ideal}_n(s^0, \Theta)$; the possible conditions under which this may be considered a good approximation will be discussed among our conclusions, section 8.

Notice, that there is no circularity in this definition and $\mathit{approx}_n(\cdots)$ can be calculated in a single sweep calculating $\mathit{approx}_1(\cdots)$, $\mathit{approx}_2(\cdots)$, ... in that order. The "argmax" in (3) may be calculated by existing algorithms as we demonstrate below.

In practical applications of our methodology, we expect the number of sub-models in a BAN to be a relatively small number (say, arbitrarily, < 10), but lengths of sequences and their annotations are expected to be huge. Measured in sequence length, the complexity of approximate prediction with the entire BAN coincides with the complexity for the most complex sub-model.

5 Training the network

In order to obtain the probabilistic parameters Θ for a BAN, we rely on existing training algorithms for supervised learning, e.g., as built into the PRISM system [7], [17]. Such algorithms require a sufficiently large and representative collection of ground atoms for each sub-model, each representing a sequence with its correct annotation, which in our motivating application domain means annotations verified in the lab by the biologists.

To this end, we assume the availability of some state-of-the-art training algorithm $T^{\mathit{supervised}}$, described as a function mapping a particular program together with its training data into a parameter. Notice that we are not interested here in the actual details of how the training algorithm works.

For doing supervised training of any sub-model in a BAN, we need in principle ground data that exemplifies the relation between sequence, parent annotations, and output annotation. We define, thus, a *conditional training data set* for program m_i as a set

$$CTD_i = \{m_i(s_i^j, a_i^j, \mathit{parents}_i^j) \mid j = 1, \ldots\}.$$

It is called "conditional" since it includes parent annotations $\mathit{parents}_i^j$ for each output annotation a_i^j.

In practice, however, we cannot expect such conditional training sets always be available as this assumes that the signals represented by the different sub-models has been analyzed consistently for the same set of sequences. In other words, we can only assume that the following sorts of training data are available in a more traditional format without explicit parent annotations.

$$TD_i = \{\langle s_i^j, a_i^j \rangle \mid j = 1, \ldots\}$$

However, if we train the different models one by one in the order M_1, M_2, ..., we can use the already trained models to supply parent annotations. We can thus specify an iterative BAN training algorithm as follows.

$$\theta_i = T^{supervised}(m_i, CTD_i)$$

where

$$CTD_i = \{m(s_i^j, a_i^j, approx_{parents_i^j}(s_i^j, \{\theta_1, \ldots, \theta_{i-1}\})) \mid \langle s_i^j, a_i^j \rangle \in TD_i\}$$

There is no circularity in these equations which may be evaluated in one sweep θ_0, θ_1, ...

This strategy can be adapted to handle cases where training data TD_i are unavailable for some non-top model M_i, i.e., $i < n$. Here we may use unsupervised training, or even set the parameters manually, and still hope for good results. It is not essential that model M_i is a faithful mirror of some physically measurable signal (call this M_i^{true}): the necessary property is whether M_i represents *some* annotation that can help the models M_j of which M_i is a parent to discriminate the details of the sequence under consideration. To see this, notice that such an M_j consistently applies annotations produced by M_i (rather than M_i^{true}) for its own training *and* prediction.

We postulate the following rule of thumb for checking the relevance of a specific model M_i within a given BAN.

(*) – Whether a model M_i contributes an interesting signal to M_j can be checked by inspection of the parameter to check whether different values for a_i provide any significant variation in the magnitude of $P(a_j \mid s^0, a_1, \ldots, a_i, \ldots, a_{j-1})$.[1]

However, we expect that models designed according to biological expert knowledge, that are trained using a sufficient set of authoritative data, and whose position in the hierarchy is based on the same biological expert knowledge, will have the best chance to constitute an interesting signal according to (*). In case of a biologically justified model, for which sufficient amounts of data are available, it will be natural also to check it with standard precision and recall methods.

We can summarize some of the practical consequences of these arguments as follows.

- M_i may for reasons of performance, or to avoid over-training, be programmed in a rather coarse way, which gives only a very rough approximation of M_i.
- We may introduce an arbitrary sub-model in a BAN, be it based on only little or no biological insights; it may be trained unsupervised or the parameter may be set by hand, and we can apply (*) to check whether it is of any use.
- We may introduce alternative models for the same biological signal, and use another model as a voting mechanism to combine the different signals and check its contribution according to (*).
- Having a collection of candidate sub-models, we can experiment with different topologies for dependencies, and validate it internally according to (*) as well as using precision and recall tests for the top model.

We will discuss some these points below in relation to our experiments.

[1] For a trained PRISM model, we may compare the different conditional `msw` probabilities produced by the training of M_j.

6 Implementation in PRISM – the LoSt Framework

The methodology described so far is supported by an implementation built on top of the PRISM system [16], which is a probabilistic extension to Prolog which provides a wide range of learning and prediction algorithms.

In this section, we first explain our own system, called the *LoSt-framework* [9], which is basically a collection of scripts that control the ordering of different runs of PRISM for prediction and training plus a file management system that keeps track of the different models, their parameters, the connections that tie them together to a BAN as well as all data files involved (catalogues of sequences, training data, files of predicted annotations, etc.). We also show a simplified, implemented example that illustrates different aspects of our methodology.

6.1 Embedding BANs in PRISM

The PRISM system [16] realizes a probabilistic extension to Prolog and is equipped with a comprehensive collection of facilities for prediction and training.

The PRISM language extends Prolog with so-called multi-valued switches: a call `msw(`*name*`, X)` represents a probabilistic choice of a value to be assigned to `X`.[2] The semantics of a PRISM program is given as a probabilistic Herbrand model, determined by a parameter which is a file of probability declarations for the individual switches. For this semantics to be well-defined, any choice point in the program must be governed by an msw.

The program part of a sub-model $m(s, a, parents)$ may be represented by a PRISM program with a main predicate

$$\texttt{m(}s\texttt{, }a\texttt{, }parents\texttt{)}$$

where *parents* are a arguments corresponding to the number of parents of m_i. A typical sequence model is implemented as a recursive predicate which relates the s and a arguments in a probabilistic fashion conditioned by given parent annotations and involving myriads of `msw` calls.

PRISM contains algorithms for training based on suitable generalizations of EM learning and Variational Bayesian learning [17] which can be used for both supervised and unsupervised learning; the LoSt environment keeps track of training data and generated parameter files for the individual sub-models.

Prediction using a PRISM program, representing a trained sub-model, can be performed using one of PRISM's generalized Viterbi algorithms. Specifically, we use a minor extension to PRISM, described in [3], which makes it possible to analyse longer sequences in reasonable time. The following call,

$$\texttt{S=}\cdots\texttt{, A1=}\cdots\texttt{, A2=}\cdots\texttt{, viterbiAnnot(m(S,A,A1,A2,}\cdots\texttt{))},$$

will instantiate `A` to the annotation that provides the highest probability of the goal `m(S,A,A1,A2,` $\cdots$`)`, thus implementing the argmax in equation (3) above.

The scripts in the LoSt environment implement the correct ordering of sub-model processing as prescribed by our incremental prediction and training algorithms described in sections 4 and 5 above, however, avoiding computations that have been made before and whose results are available on files.

[2] To be exact, a switch introduced by a declaration `values(name, [`$\cdots$ *outcomes* $\cdots$`])` defines a family of random variables, one for each execution of `msw(`*name*`,` $\cdots$`)` in a program run.

6.2 Example: Gene-finding in DNA

We illustrate our methodology showing experiments with BANs that represent gene-finders for DNA sequences. A piece of DNA is a sequence of letters $\{a, c, g, t\}$; it can be viewed as a sequence of triplets, each called a codon. Codons are separated into specific start-codons, stop-codons and other codons; a gene is a specific subsequence matching the codon structure such that it must begin with a start codon and it will definitely end at the next stop-codon; such a syntactic pattern is called an open reading frame (ORF). Our BAN models are designed to annotate ORFs, where the annotation task is to find out whether an ORF contains a gene and, if so, where in the sequence the gene starts.

We define sub-models for different signals — codon preference, gene length and conservation — which are are expected to have influence on whether a sequence is a gene or not. The resulting annotation from such models is a sequence that for each position in the original sequence contains a `1` if the position is predicted as part of a gene and a `0` if it is not.

All our probabilistic models are output HMMs with a gene-state and a non-gene state, which can emit symbols of the annotations of the parent nodes. The transitions between the states reflect the described ORF pattern.

The *codon preference* model **m1** reflects preferential codon usage in the gene and non-gene state. The states can each emit one of the 64 possible codons.

The *gene length* annotation is obtained by using a deterministic model **m2** that annotates each potential start codon with a symbol representing the distance to the upstream stop codon.

Conservation describes a degree to which the codons of a DNA sequence are conserved across species. To detect conservation, each ORF matched to a database of genome sequences of distantly related organisms [3] using the tblastn tool, which produce a gapped alignment of the matches. Only statistically significant matches (evalue $< 10^{-34}$) and only one match per organism are reported. The conservation model **m3** emits identity positions of reported matches to ORFs.

In the following we discuss and assess a number of BAN topologies built using these three signals as basic building blocks. The considered models are **m1**, **m3**, **m1** conditioned on **m2** – **m1(m2)**, **m1** conditioned on **m3** – **m1(m3)**, and **m1** conditioned on both **m2** and **m3** – **m1(m2,m3)**.

We train and predict on the well-annotated *Escherischia Coli* genome and its curated gene annotations from refseq (NC_000913). We have randomly divided the ORFs of the genome into a training and a test set. Supervised training is done using only the former and the method for supervised training algorithm described in section 5. We report prediction accuracy results for both sets. Accuracy is measured as $\mathit{Sensitivity}(SN) = \frac{TP}{TP+FN}$ and $\mathit{Specificity}(SP) = \frac{TP}{TP+FP}$, with respect to annotation of start and stop codons. The results are summarized in table 1.

It can be observed from table 1 that all our models have good generalization capabilities, since the performance is very similar on both the training and test set.

The best model seems to be **m1(m2)**, which achieve a significant increase in specificity with only slightly degraded sensitivity, e.g. it predicts fewer genes but its predictions are more reliable. By them selves, both **m1** and **m3** have reasonable stop specificity, but **m3** has consistent tendency to predict too long genes, leading to severely decreased start specificity.

[3] The sequences are from refseq: NC_004547, NC_008800, NC_009436, NC_009792, NC_010067, NC_010694 and NC_011283.

Table 1 Accuracy of predictions using different BAN topologies.

	Training set (114429 ORFs, 2075 genes)				Test set (114404 ORFs, 2065 genes)			
BAN	SN_{start}	SP_{start}	SN_{stop}	SP_{stop}	SN_{start}	SP_{start}	SN_{stop}	SP_{stop}
m1	0.7701	0.2935	0.9711	0.3701	0.7564	0.2920	0.9719	0.3751
m3	0.0636	0.0322	0.8255	0.4183	0.0140	0.0072	0.8412	0.4298
m1(m2)	0.6723	0.5011	0.9345	0.6965	0.6489	0.4896	0.9433	0.7117
m1(m3)	0.4405	0.2243	0.8255	0.4204	0.4315	0.2216	0.8416	0.4323
m1(m2,m3)	0.4361	0.2228	0.8255	0.4217	0.4174	0.2149	0.8416	0.4333

Interestingly, conditioning **m1** on the conservation additional signal **m3** does not improve prediction accuracy much. It does lead to slightly better stop specificity but it tends to degrade the start specificity. Additionally, conditioning on the length signal as done in **m1(m2,m3)** does not seem to help, even though the impact observed in **m1(m2)** was quite significant. It seems that the **m3** signal dominates decisions about which ORFs should be predicted as coding. This effect is apparent from model parameters and it is possible to get an intuition of the problem from inspection of the prediction accuracies.

The **m3** model has a (stop) false negative rate of $1 - SP_{stop} = 1 - 0.4183 \approx 0.58$. The vast majority of ORFs $\sim 98\%$ does not contain genes. The probability that an ORF contains a gene but **m3** classifies it as non-gene is thus relatively small, $1 - 0.98 \times 0.58 \approx 0.11$. In the conditional distribution defined by **m1(m3)** (given predictions of **m3**), it becomes virtually impossible for the viterbi algorithm to classify an ORF as a gene if **m3** has not, since the probability of the gene hypothesis is scaled by ~ 0.11 and the non-gene hypothesis by ~ 0.89.

Part of the explanation is that maximizing the likelihood of observed data (as we do in training) is not equivalent to maximizing prediction accuracy; it may have an adverse effect when selecting predictions as most probable explanations as done by the viterbi algorithm. An other part of the explanation is in our model assumptions; namely, **m1(m3)** is an output HMM that has joint emissions of both codon and the signal from **m3**, and these are dominated by **m3** as explained above. Alternative HMM structures with different constraints and independence tradeoffs might avoid the dominating effect of **m3**. We are still investigating how this is best done.

7 Related work

Our method is closely related to *Dynamic Bayesian Networks* (DBNs) of [10]. By our definition of a BAN, the detailed dependencies between individual models in the network are left abstract, but a concrete instantiation of a BAN may indeed be a DBN. However, as the nodes in a BAN may be arbitrary probabilistic models, for instance context-free grammars, not all BAN instantiations can be represented as DBNs. Oppositely, we only define BANs for discrete models but DBNs may include continuous-valued nodes.

In the realm of classification techniques, it is common to combine the results of different classifiers of the same phenomena in ways such that the combined classifications outperform the individual constituent classifiers. Such methods are generally known by the name *ensemble methods*, which covers a wide range of different ways to the combine classifiers [14]. Our method is related but quite different; this is not just because we consider sequence annotation rather than classification, but also because constituent models of a BAN may model very different phenomena.

In biological sequence analysis, the most successful genome annotation programs are

combiners [4]; programs which combines different sources of annotation evidence using some sort of weighting scheme. Evidence may come in diverse forms, including comparative analysis sources [12], but are typically predictions (e.g. annotations) from other annotation programs (e.g. gene finders). Brent [1] makes a distinction between combiners and joint models, where joint models are described as models which consider the full joint probability distributions evidence and combiners as probabilistic models of the the relative accuracy of evidence sources they are combining. Using our approximate inference algorithm we have a situation similar to combiners in that predictions of parents are combined by child nodes.

While many combiners use non-probabilistic combinations methods, several are explicitly based on principles of (dynamic) Bayesian networks [11, 8]. A main difference is that our framework allows multi-layered and branching topologies where the combiners are usually just single layered probabilistic models.

Our approach also has analogies to *annotation pipelines* [13, 2] where a complex sequence of analysis steps are performed in a possibly branching topology and perhaps synthesized (e.g. by a combiner) in a final annotation as the last step. Opposed to combiners, such pipelines usually allows complex topologies like our framework. However, such pipelines are usually just practical and pragmatic ways of combining existing tools and incorporate probabilistic modeling only to a very limited degree.

There are other declarative approaches to combining evidence in biological sequence analysis. In GAZE [5], a configurable XML-based specification describes a particular composition of evidence sources. However, GAZE integrates existing tools, where our PRISM based approach allows for much more modelling flexibility and have clear and well-defined semantics.

8 Conclusions

We have proposed a Bayesian framework, *Bayesian Annotation Networks*, which allows the representation and composition of models for complex sequence analysis. In a modular way, it supports experimentation with and evaluation of models and signals and it is a practically useful tool for modeling and analyzing sequences. In particular, its applicability to biological sequence analysis has been motivated. We have shown that reasonable complexity can be achieved by the use of tractable, incremental algorithms for inference and training, which can be implemented by successive calls to PRISM, and shown that these algorithms may produce useful annotations.

In general, we have no good analytical or sampling-based principles for analyzing the quality of the approximated annotations compared with the ideal ones. By assumption, the ideal annotations provided by a BAN for a given sequence is too complex to be evaluated, so we need to rely on standard validation techniques based on authoritative test data. However, we will list a few observations which may be used as guidelines.

The crux in our approximate inference algorithm is, in each iteration step, to select a most probable annotation $approx_i$ for each annotation node a_i and take it as a representative for the distribution of all possible a_i values. In the detailed calculations, this means that we use $P(a_j \mid s, \cdots approx_i \cdots)$, for some a_j with $a_i \in parents_j$, as a replacement of a weighted sum over all possible a_i values of $P(a_j \mid s, \cdots a_i \cdots)$.

In the trivial case, where all freedom of choice is implemented in the top node of the Bayesian Network, the approximate algorithm coincides to the ideal. Beyond the trivial case, however, it is difficult (impossible in general) to give sufficient conditions for which the approximate inference method will yield good results.

The relation between the quality of an annotation and its probability is assumed implied by the purpose of a probabilistic annotation model; e.g. it should assign high probability to good annotations. In our definitions of BANs, we define a child model to be dependent on its parent model. In a concrete BAN, however, individual models typically have more fine grained interdependencies, e.g. enforce their inherent independence assumptions. If these assumptions are not faithful to the actual data dependencies, a discordance between annotation quality and probability may arise. Similarly, in the case of approximate inference, we are concerned in particular with the degree of validity of the assumption about independence of parent distributions given the most probable individual elements of those distributions.

Perhaps surprisingly, the annotation quality achieved by the approximate method may be positively affected by correlation between assumed independent nodes of the network. Redundant (correlated) signals does not generally result in better annotations, if the ideal inference method is used. However, such overlapping signals may indeed compensate for the information lost due to the (possibly) unjustified independence assumptions imposed by the approximation method or inherent in constituent models. For instance, information contained in the distribution of a particular parent node, but not reflected by the best annotation from that distribution, may be reflected through the best annotation of some other (correlated) parent.

In practice, we are satisfied with the approximation if the annotations are judged as good using an external measure of quality (e.g. sensitivity/specificity) and we have used cross-validation to build confidence about generality, as demonstrated in section 6.2. Obviously, this may require a considerable amount of, possibly unavailable, labelled training data. A second consequence, also observed in section 6.2, is that the measure optimized by the training algorithm does not necessarily coincide with the external measure of quality. Model constraints and independence assumptions play a key role affecting the correlation between these measures.

References

1 Michael R Brent. Steady progress and recent breakthroughs in the accuracy of automated genome annotation. *Nature Reviews Genetics*, 9(1):62–73, 2008.

2 Brandi L Cantarel, Ian Korf, Sofia M C Robb, Genis Parra, Eric Ross, Barry Moore, Carson Holt, Alejandro Sánchez Alvarado, and Mark Yandell. MAKER: an easy-to-use annotation pipeline designed for emerging model organism genomes. *Genome Research*, 18(1):188–196, 2008.

3 Henning Christiansen and John Gallagher. Non-discriminating Arguments and their Uses. In *Logic Programming, 25th International Conference, ICLP 2009, Lecture Notes in Computer Science 5649*, pages 55–69. Springer, 2009.

4 Roderic Guigó, Paul Flicek, Josep F Abril, Alexandre Reymond, Julien Lagarde, France Denoeud, Stylianos Antonarakis, Michael Ashburner, Vladimir B Bajic, Ewan Birney, Robert Castelo, Eduardo Eyras, Catherine Ucla, Thomas R Gingeras, Jennifer Harrow, Tim Hubbard, Suzanna E Lewis, and Martin G Reese. EGASP: the human ENCODE Genome Annotation Assessment Project. *Genome Biology*, 7(Suppl 1):S2, 2006.

5 Kevin L Howe, Tom Chothia, and Richard Durbin. GAZE: A Generic Framework for the Integration of Gene-Prediction Data by Dynamic Programming. *Genome Research*, 12(9):1418–1427, 2002.

6 Finn V. Jensen and Thomas D. Nielsen. *Bayesian Networks and Decision Graphs*. Springer, 2 edition, 2007.

7 Yoshitaka Kameya and Taisuke Sato. Efficient em learning with tabulation for parameterized logic programs. In John W. Lloyd, Verónica Dahl, Ulrich Furbach, Manfred Kerber, Kung-Kiu Lau, Catuscia Palamidessi, Luís Moniz Pereira, Yehoshua Sagiv, and Peter J. Stuckey, editors, *Computational Logic*, volume 1861 of *Lecture Notes in Computer Science*, pages 269–284. Springer, 2000.

8 Qian Liu, Aaron J Mackey, David S Roos, and Fernando C N Pereira. Evigan: a hidden variable model for integrating gene evidence for eukaryotic gene prediction. *Bioinformatics*, 24(5):597–605, 2008.

9 LoSt. The lost project, 2007. http://lost.ruc.dk.

10 K P Murphy. Dynamic bayesian networks. *Probabilistic graphical models*, 41(November):515–29, 2003.

11 Vladimir Pavlović, Ashutosh Garg, and Simon Kasif. A Bayesian framework for combining gene predictions. *Bioinformatics*, 18(1):19–27, 2002.

12 Maria S. Poptsova and J. Peter Gogarten. Using comparative genome analysis to identify problems in annotated microbial genomes. *Microbiology*, 156(7):1909–1917, 2010.

13 Simon C Potter, Laura Clarke, Val Curwen, Stephen Keenan, Emmanuel Mongin, Stephen M J Searle, Arne Stabenau, Roy Storey, and Michele Clamp. The Ensembl Analysis Pipeline. *Genome Research*, 14(5):934–941, 2004.

14 Lior Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39, 2009.

15 Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series In Artificial Intelligence. Prentice Hall, 2003.

16 Taisuke Sato. Generative Modeling by PRISM. *Proceedings of the International Conference on Logic Programming*, LNCS 5649:24–35, 2009.

17 Taisuke Sato, Yoshitaka Kameya, and Kenichi Kurihara. Variational Bayes via propositionalized probability computation in PRISM. *Annals of Mathematics and Artificial Intelligence*, 54(1-3):135–158, 2009.