Knowledge Engineering: Homework 4

Yifei Zuo
PB20061254

Yunqin Zhu
PB20061372

1 Probabilistic Logics

1.1 Review Kolmogrov's probability axioms and how they are related to probabilistic logic.

Solution

Probability space can be formalized as a measure space $(\Omega, \mathcal{F}, \mathrm{P})$ with $\mathrm{P}(E)$ being the probability of some event $E \in \mathcal{F}$ and $\mathrm{P}(\Omega) = 1$, eq uipped with following *Kolmogorov axioms*:

- Non-negativity: $\mathrm{P}(E) \in \mathbb{R}, \mathrm{P}(E) \geq 0$ for $\forall E \in \mathcal{F}$.
- Unitarity: $\mathrm{P}(\Omega) = 1$.
- σ-additivity: Any countable sequence of disjoint sets over mutually exclusive events $E_1, E_2, \ldots$ satisfies $\mathrm{P}\left(\bigvee_{i=1}^{\infty} E_i\right) = \sum_{i=1}^{\infty} \mathrm{P}(E_i)$.

Kolmogorov axioms serves as foundation to probabilistic logic's semantics and reasoning, in the following sense:

- We can formalize one statement as an event in probability space, where non-negativity and unitarity restricts the semantic of the probability of a statement being true is always between 0 and 1, inclusive.
- A probability of 0 means that the statement is considered false, while a probability of 1 means that the statement is considered true. Any value between 0 and 1 represents the degree of belief in the truth of the statement.
- The unitarity combined with additivity further constrains that for any statement, the sum for the statement being true and being false will always equal to 1, i.e. $\mathrm{P}(A) + \mathrm{P}(\neg A) = 1$, for any statement A.
- Additivity also contributes to the axiom of reasoning in probabilistic logic, where for any A, B being two mutually exclusive statement, $\mathrm{P}(A \vee B) = \mathrm{P}(A) + \mathrm{P}(B)$ which represent the probability for either A or B being true.

1.2 Follow the examples given in the slides, and show how to use Bayesian network and Markov logic network for problem solving.

Solution

- **Baysian network:**
  Consider the following knowledge base in Bayesian network:

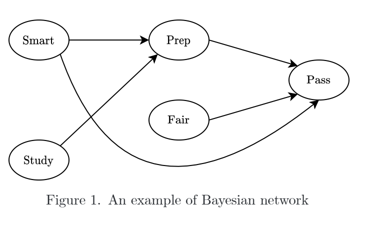

Figure 1. An example of Bayesian network

Suppose that our goal is to query the posterior $\mathrm{P}(\mathsf{study} \mid \mathsf{pass})$ based on $\mathrm{P}(\mathsf{smart}) = 0.8$, $\mathrm{P}(\mathsf{study}) = 0.6$, $\mathrm{P}(\mathsf{fair}) = 0.9$ and the following conditional probability tables:

Table 1. Conditional probability table of Prep

$\mathrm{P}(\mathsf{prep} \mid \cdot)$	smart	¬smart
study	0.9	0.7
¬study	0.5	0.1

Table 2. Conditional probability table of Pass

$\mathrm{P}(\mathsf{pass} \mid \cdot)$	smart ∧ prep	smart ∧ ¬prep	¬smart ∧ prep	¬smart ∧ ¬prep
fair	0.9	0.7	0.7	0.2
¬fair	0.1	0.1	0.1	0.1

We write the target posterior as:

$$
\begin{aligned}
&\mathbf{P}(\mathsf{Study} \mid \mathsf{pass}) = \frac{\mathbf{P}(\mathsf{pass}, \mathsf{Study})}{\mathrm{P}(\mathsf{pass})} = \alpha \mathbf{P}(\mathsf{pass}, \mathsf{Study}) \\
&= \alpha \sum_{\mathsf{smart}} \sum_{\mathsf{prep}} \sum_{\mathsf{fair}} \mathbf{P}(\mathsf{pass}, \mathsf{Study}, \mathsf{smart}, \mathsf{prep}, \mathsf{fair}) \\
&= \alpha \sum_{\mathsf{smart}} \sum_{\mathsf{prep}} \sum_{\mathsf{fair}} \mathrm{P}(\mathsf{pass} \mid \mathsf{smart}, \mathsf{prep}, \mathsf{fair}) \mathbf{P}(\mathsf{prep} \mid \mathsf{smart}, \mathsf{Study}) \mathrm{P}(\mathsf{smart}) \mathbf{P}(\mathsf{Study}) \mathrm{P}(\mathsf{fair})
\end{aligned}
$$

The last line of the equation can be computed by enumerating all 2^3 possible combinations of **Smart**, **Prep** and **Fair**, or by a more efficient inference algorithm such as variable elimination as shown below:

$$
\begin{aligned}
&\mathbf{P}(\mathsf{Study} \mid \mathsf{pass}) \\
&= \alpha \mathbf{P}(\mathsf{Study}) \sum_{\mathsf{smart}} \mathrm{P}(\mathsf{smart}) \sum_{\mathsf{prep}} \mathbf{P}(\mathsf{prep} \mid \mathsf{smart}, \mathsf{Study}) \sum_{\mathsf{fair}} \mathrm{P}(\mathsf{pass} \mid \mathsf{smart}, \mathsf{prep}, \mathsf{fair}) \mathrm{P}(\mathsf{fair}) \\
&= \alpha \mathbf{P}(\mathsf{Study}) \sum_{\mathsf{smart}} \mathrm{P}(\mathsf{smart}) \sum_{\mathsf{prep}} \mathbf{P}(\mathsf{prep} \mid \mathsf{smart}, \mathsf{Study}) \sum \left\langle \begin{pmatrix} 0.9 & 0.7 \\ 0.7 & 0.2 \end{pmatrix}, \begin{pmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{pmatrix} \right\rangle \times \langle 0.9, 0.1 \rangle \\
&= \alpha \mathbf{P}(\mathsf{Study}) \sum_{\mathsf{smart}} \mathrm{P}(\mathsf{smart}) \sum \left\langle \begin{pmatrix} 0.9 & 0.7 \\ 0.5 & 0.1 \end{pmatrix}, \begin{pmatrix} 0.1 & 0.3 \\ 0.5 & 0.9 \end{pmatrix} \right\rangle \times \left\langle \begin{pmatrix} 0.82 & 0.64 \end{pmatrix}, \begin{pmatrix} 0.64 & 0.19 \end{pmatrix} \right\rangle \\
&= \alpha \mathbf{P}(\mathsf{Study}) \sum \langle 0.8, 0.2 \rangle \times \left\langle \begin{pmatrix} 0.802 & 0.73 \end{pmatrix}, \begin{pmatrix} 0.505 & 0.235 \end{pmatrix} \right\rangle \\
&= \alpha \begin{pmatrix} 0.6 & 0.4 \end{pmatrix} \circ \begin{pmatrix} 0.7426 & 0.631 \end{pmatrix} \\
&= \alpha \begin{pmatrix} 0.44556 & 0.2524 \end{pmatrix} \approx \begin{pmatrix} 0.6384 & 0.3616 \end{pmatrix}
\end{aligned}
$$

Therefore, the probability $\mathbf{P}(\mathsf{study} \mid \mathsf{pass}) = 0.6384$.

- **Markov logic network:**
  Consider a Markov logic network with the following formulas:

$$
\begin{gathered}
\forall x, \mathsf{Smokes}(x) \rightarrow \mathsf{Cancer}(x) \\
\forall x, y, \mathsf{Friends}(x, y) \rightarrow (\mathsf{Smokes}(x) \leftrightarrow \mathsf{Smokes}(y))
\end{gathered}
$$

For simplicity, we assign the same weight $w = 1$ to both formulas. Suppose that the domain contains only two constants A and B, representing two people Anna and Bob, respectively. Then, the ground Markov network is constructed as follows:

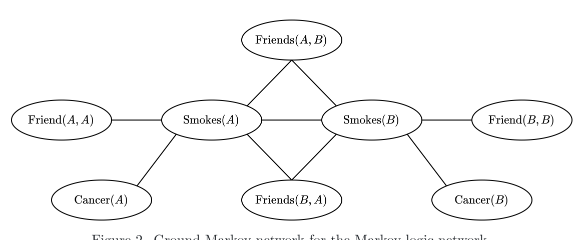

Figure 2. Ground Markov network for the Markov logic network.

A general problem solved by Markov logic network is to compute the probability of a query y given some evidence x:

$$
\mathrm{P}(y \mid x) = \frac{\mathrm{P}(x, y)}{\mathrm{P}(x)} = \frac{\sum_h \exp\left(\sum_i w_i n_i(x, y, h)\right)}{\sum_{y,h} \exp\left(\sum_i w_i n_i(x, y, h)\right)}
$$

where both x and y are first-order logic formulas, ideally conjunctions of ground atoms, and Z_x is the so-called partition function. In practice, we often use approximate inference methods such as Markov Chain Monte Carlo (MCMC) with Gibbs sampling to compute such probability. Here, we give an example of exact inference. Suppose that we want to compute the probability that Bob has cancer given that Anna smokes and they are friends. Then

$$
\begin{aligned}
&\mathrm{P}(\mathsf{Cancer}(B) \mid \mathsf{Smokes}(A) \wedge \mathsf{Friends}(A, B) \wedge \mathsf{Friends}(B, A)) \\
&= \frac{\exp(1+2) + \exp(1+0)}{\exp(1+2) + \exp(1+0) + \exp(0+2) + \exp(1+0)} \approx 0.6929
\end{aligned}
$$

Another typical problem is to find the most possible world, which can be solved by MaxWalkSAT, etc. We omit the details here.

1.3 Review how to do inference and learning for Bayesian network and Markov logic network.

Solution

- **Bayesian network:**
  Bayesian network inference includes exact inference and approximate inference. Former methods compute the exact probabilities for the queries. Some examples include the Variable Elimination algorithm, the Clique Tree algorithm, and the Hugin algorithm, while the latter one provide approximations of the probabilities when exact computation is intractable. Some examples include Monte Carlo methods, such as the Markov Chain Monte Carlo (MCMC) and the Gibbs sampling algorithm, and the Loopy Belief Propagation algorithm.

  We here present enumeration for BN inference, which is a brute-force fashion method with worse time and space complexity. The worst-case time complexity of the enumeration inference method is $O(nd^k)$, where:

  - n is the number of variables in the Bayesian network
  - d is the maximum number of values a variable can take (i.e., the domain size)
  - k is the maximum number of parents a variable can have (i.e., the maximum in-degree in the graph)

  When performing actual inference, we basically categorize the random variables into three groups: observed, unobserved and target variables. We denotes the collection of unobserved variables as $\mathbf{Y}$ and observed variables as $\mathbf{Y}$, then the target distribution give evidence would be:

$$
\mathbf{P}(X|e) = \alpha P(X, \mathbf{E}) = \alpha \sum \mathbf{P}(X, \mathbf{E}, \mathbf{Y})
$$

If we want to calculate the probability for X being true given e, we have to iterate all variables in the collection $\mathbf{E}$ s.t.

$$
\mathbf{P}(x|e) = \alpha \sum_{(y_1, \ldots, y_n) \in \mathbf{Y}} \mathbf{P}(x, e, y_1, \ldots, y_n)
$$

The learning task in Bayesian network can also be divided into two parts: structure learning and parameter learning. The goal of *structure learning* is to discover the best network structure (DAG) given a dataset. This can be done using search-and-score methods (e.g., hill climbing, simulated annealing) or constraint-based methods (e.g., PC algorithm, FCI algorithm). Once the structure is known or assumed, the conditional probability tables (CPTs) can be learned from the data in *parameter learning*. This is typically done using Maximum Likelihood Estimation (MLE) or Bayesian Estimation (BE). In course slides, we basically mentioned parameter learning with the view under specifying for each node X the probability distribution for X conditional upon X's parents.

- **Markov logic network:**
  The goal of inference in a Markov Logic Network is to find stationary distribution of this system or an approximation. A stationary distribution assigns the most likely assignment of probability of the vertices in the graph, which could be seen as the ground atoms of an interpretation. Inference in MLNs can be performed using standard Markov network inference techniques over the minimal subset of the relevant Markov network required for answering the query. A MAE/MPE inference task intends to find the most likely state of world given evidence:

$$
\arg\max_y \sum_i w_i n_i(x, y)
$$

which could be further formalized as a weighted MaxSAT problem. Therefore, we could use weighted SAT solver to solve the inference problem in reasonable time complexity if we handle it in a lazy evaluation fashion. MCMC could be utilized to compute the query's probability as well, which leverages Gibbs sampling:

Algorithm 1: MCMC with Gibbs sampling

```
state ← random truth assignment
for i ← 1 to num-samples do
    for each variable x
        sample x according to P(x|MB(x))
        state ← state with new value of x
P(F) ← fraction of states in which F is true
```

However, it suffers from efficiency drawback in non-deterministic problems while a deterministic dependences break the execution of MCMC foundation. One possible solution is to combine MCMC and WalkSAT into an integrated method, leading to the MC-SAT algorithm. We give the pseudo-code of MaxWalkSAT algorithm as follows:

Algorithm 2: MaxWalkSAT

```
for i ← 1 to max-tries do
    solution = random truth assignment
    for j ← 1 to max-flips do
        if the sum of weights of satisfied clauses is greater than threshold
            return solution
        c ← random unsatisfied clause
        with probability p
            flip a random variable in c
        else
            flip variable in c that maximizes the sum of weights of satisfied clauses
return failure, best solution found
```

Learning in MLN also combines the *structural learning* and *parameter learning*. A step to step weight learning algorithm is *Generative Weight Learning*, which tries to maximize the likelihood using gradient ascend fashion:

$$
\frac{\partial}{\partial w_i} \log \mathrm{P}_w(x) = n_i(x) - \mathbb{E}_w[n_i(x)]
$$

Since this method yields gradient computation in each step of inference, the computation expense is unacceptable. One possible way to avoid step-wise calculation inference would be *pseudo-likelihood* method, which considers the value as product of each variable's conditional probability given it neighbors (Markov blanket):

$$
\mathrm{PL}(x) := \prod_i \mathrm{P}(x_i|\mathrm{MB}(x_i))
$$

The problem with this method is the inaccuracy in long range inference chains. Several other method like *Discriminative Weight Learning* and *Voted Perceptron* were proposed later to further address this problem. In structural learning, the actual bottleneck is the efficiency of counting clause groundings, therefore, subsampling techniques are applied to compromise.

1.4 Again, pick up one of the application areas mentioned in the slides. Go deeper and deeper to explain how probabilistic logics are used in these areas.

Solution

- **Area:** Link Prediction
  Link prediction is the problem of predicting the existence of a link between two entities in a network. It can be used to predict friendship links among users in a social network, co-authorship links in a citation network, and interactions between genes and proteins in a biological network, etc.

  Consider a network $G = (V, E)$, where V represents the entity nodes in the network and $E \subseteq |V| \times |V|$ represents the set of true links across entities in the network. We are given the set of entities V and a subset of true links which are referred to as *observed* links. The goal of link prediction is to identify the *unobserved* true links in a set E' of *potential* links.

- **Logical formulation:**
  In real-world networks, each entity nodes may have multiple attributes. We can formulate these attributes as predicates of the form $A(x, v)$, where A is an attribute, $x \in V$ is an entity node, and v is the value of A in x. Similarly, we can formulate relations between entities as predicates of the form $R(x, y)$, corresponding to a link $e \in E$ between two nodes $x, y \in V$. It is worth noting that heterogeneous networks can also be represented in this way, where different types of nodes and links are represented by different predicates.

  Accordingly, we can formulate the problem of link prediction as a logical inference problem. Given a knowledge base KB of observed links, node attributes and domain knowledge, our goal is to infer the existence of true links between nodes in E'. In the sense of classical logics, we infer the truth values of predicates $R(x, y)$ for all pairs of nodes x, y that are connected by a potential link in E'. Link prediction approaches for this setting learn a binary classifier f that maps links in E' to positive and negative labels, i.e. $f : E' \to \{0, 1\}$.

  In the sense of probabilistic logics, however, we infer the probabilities of predicates $R(x, y)$. Link prediction approaches for this setting learn a model f that maps links in E' to a probability, i.e. $f : E' \to [0, 1]$. Both Bayesian networks and Markov (logic) networks can be used to model and solve the problem of link prediction.

- **Probabilistic relational models:**
  Specifically, we often define a *probabilistic relational model* (PRM) as a template for the joint probability distribution over the attributes of a relational database. The template describes the relational schema for the domain, and the probabilistic dependencies between attributes in the domain. A PRM, together with a particular database of entities and unobserved links, defines a probability distribution over the unobserved links. Inference is often done by creating the complete ground network, which limits their scalability. PRM was originally designed for attribute prediction in relational data, and it later extended to link prediction tasks.

  We divide PRMs into two group based on representation: *relational Bayesian network* (RBN) and *relational Markov network* (RMN):

  - **Relational Bayesian networks:**
    The knowledge base represented by a RBN is a directed acyclic graph, with a set $\mathcal{P}$ of CPDs to represent a joint distribution over attributes and relations of entities. The set $\mathcal{P}$ contains a conditional probability distribution for each ground atom given its parents $\mathrm{P}(x \mid \mathrm{Pa}_x)$. The need to avoid cycles in RBN leads to significant representational and computational difficulties.
  - **Relational Markov networks:**
    A RMN uses an undirected graph and a set of potential functions Φ to represent the joint distribution over attributes and relations of entities. Denote by C the set of cliques in the ground network and each clique $c \in C$ is associated with a set of variables X_c (i.e. the nodes in this clique) and a clique potential $\Phi_c(x_c)$ which is a non-negative function over the possible values for X_c, then the joint probability over x is calculated with the formula $\mathrm{P}(x) = \frac{1}{Z} \prod_{c \in C} \Phi_c(x_c)$, where Z is a normalizing constant, which sums over all possible instantiations. RMN has strong relation with Markov logic networks, which is a first-order logic based template for Markov networks.

- **Other models:**
  Apart from PRMs, many pioneer works in link prediction also use probabilistic logics to model the problem. Listed below are some of them (references are omitted for brevity):

Table 3. Probabilistic graphical models for link prediction

Model	**Network types**	**Characteristics**
Hierarchical structure model	Hierarchical networks	High accuracy for HSM and low for non-HSM structure
Stochastic block model	Noisy networks	Good at predicting spurious and missing links
Parametric model	Dynamic networks	Extracts only topological features and performs better than structural methods
Non-parametric model	Dynamic networks	Explicitly clusters links instead of nodes
Local Markov random fields	Coauthorship networks	Combines co-occurrence features with topological and semantic features
Factor graph	Heterogeneous social networks	Link prediction with aggregate statistics problem

1.5 Go deeper to discuss the similarities and differences among probabilistic logics.

Solution

We consider propositional probabilistic logic, Bayesian networks and Markov logic networks.

- **Similarities:**
  (1) Probabilistic logics combine probability theory and logic by adopting degree of belief as epistemological commitment.
  (2) Inference in probabilistic logics involves the calculation of probabilities and is based on the Kolmogrov's axiom system.
  (3) Inference can either be exact or approximate. Exact inference is usually NP-hard. Efficient inference methods are powered by propagation-based and particle-based algorithms, such as belief propagation and Markov chain Monte Carlo.
  (4) Logical formulas (structure of Bayesian/Markov network) and their corresponding probability distributions (model parameters) can either be specified by experts or learned from data. Learning can be discriminative or generative, supervised or unsupervised, heuristic or principled.
- **Differences:**
  (1) Propositional probabilistic logic and Bayesian networks are both based on propositional logic, while Markov logic networks are based on first-order logic thus more expressive.
  (2) Propositional probabilistic logic represents knowledge with plain logical formulas and their probabilities, while Bayesian networks and Markov logic networks are built on probabilistic graphical models, enabling compact representation and efficient inference.
  (3) Propositional probabilistic logic does not specify dependencies between atoms explicitly; Bayesian networks represent conditional dependencies with directed edges induced by Horn clauses, while not allowing cycles; Markov logic networks represent conditional dependencies by undirected edges induced by first-order logic formulas.
  (4) The learnable parameters of propositional probabilistic logic and Bayesian networks are conditional probabilities, while the parameters of Markov logic networks are weights of formulas.
  (5) The semantics of propositional probabilistic logic and Bayesian networks are defined by direct calculation of probabilities using law of total probability or Bayes' rule, while Markov logic networks serve as a template for Markov networks, where the probability is dependent on the weights of features and the potential functions.
  (6) Inference tasks of Bayesian networks include simple queries, conjunctive queries, optimal decisions, value of information, sensitivity analysis and explanation; inference tasks of Markov logic networks include finding the most probable explanation and computing marginal or conditional probabilities. The log-linear models in Markov logic networks allows for the use of Weighted Max-SAT solvers to perform inference, which cannot directly be done in the other two probabilistic logics.
  (7) Parameter learning of propositional probabilistic logic and Bayesian networks is usually done generatively by estimating probability distributions, while weight learning of Markov logic networks can either be generative or discriminative.

1.6 Go deeper to discuss the similarities and differences between probabilistic logics, rule based logics and classical logics.

Solution

- **Similarities:**
  (1) All these logics are formal languages, with specific syntax and semantics.
  (2) All these logics enable representation, reasoning and learning of knowledge, can be used to make predictions and support decisions, and share the same goal of building intelligent systems.
  (3) All these logics seek ways to model what exists in the world (e.g. facts, relations) and what an agent believes about facts (e.g. truth value, degree of belief).
  (4) All these logics have wide applications in the fields of machine learning, natural language processing, robotics, information retrieval, programming languages, etc.
- **Differences:**
  (1) Different logic has different ontological and epistemological commitments, as shown in the following table (modified from Russell et al., 2022):

Table 4. Formal languages and their ontological and epistemological commitments.

Language	**Ontological commitment**	**Epistemological commitment**
Propositional logic	facts	true/false(/unknown)
First-order logic	facts, objects, relations	true/false(/unknown)
Rule-based logic	facts (with rules), etc.	true/false(/unknown)
Propositional probabilistic logic	facts	degree of belief $\in [0, 1]$
First-order probabilistic logic	facts, objects, relations	degree of belief $\in [0, 1]$
Fuzzy logic	facts with degree of truth	known interval value

  (2) Each logic has its own syntax. In specific, classical logics use connectives such as $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$ to construct formulas; rule-based logics restrict the syntax to definite clauses and extend it for practical use; probabilistic logics assign probabilities to formulas and represent them with graphical models.
  (3) Each logic has its own semantics. In specific, classical logics use truth tables to define the semantics of formulas; rule-based logics may use Herbrand models to define the semantics of programs; probabilistic logics use probability distributions and dependencies among atoms as its semantics.
  (4) Different logics have different expressiveness (owing to their different syntax and semantics). For example, first-order logic is more expressive than propositional logic by allowing predicates, functions and quantifiers; some rule-based logics enhance their expressiveness by allowing negation as failure, aggregate functions or transitive closure; probabilistic logics can be considered as more expressive than classical logics by allowing real-valued degrees of belief.
  (5) Different logics have different inference tasks and methods. For example, model checking in classical logics can be done by exhaustive search or efficient SAT solvers; forward/backward chaining and resolution are used in rule-based logics to evaluate programs; probabilistic logics cover a wide range of inference tasks, including probability queries, most probable explanation, optimal decisions, value of information, sensitivity analysis, etc., which can be done either exactly or approximately.
  (6) Learning propositional or first-order formulas is usually done by inductive logic programming; parameter learning of probabilistic graphical models benefit from the rich literature of machine learning, including decision trees, neural networks, logistic regression, etc., besides traditional MLE or MAP-based estimation; structure learning of probabilistic graphical models involves heuristic or metaheuristic search algorithms.
  (7) Classical and rule-based logics can be used to find a optimal sequence of actions in a discrete, deterministic, static, fully observable environment, i.e. classical planning; probabilistic logics can be combined with Markov decision processes to solve sequential decision making problems in stochastic environments.
  (8) Classical and rule-based logics correspond to the definite clause grammar in natural language processing, while probabilistic logics correspond to the probabilistic context-free grammar.

References

Russell, S. J., Norvig, P., Chang, M., Devlin, J., Dragan, A., Forsyth, D., Goodfellow, I., Malik, J., Mansinghka, V., Pearl, J., & Wooldridge, M. J. (2022). *Artificial Intelligence: A Modern Approach* (Fourth edition, global edition). Pearson.

Pearl J (1985). Bayesian Networks: A Model of Self-Activated Memory for Evidential Reasoning (UCLA Technical Report CSD-850017). *Proceedings of the 7th Conference of the Cognitive Science Society*, University of California, Irvine, CA. pp. 329–334.

Wikipedia contributors. (2023). Bayesian network. In Wikipedia, The Free Encyclopedia. Retrieved 16:09, May 1, 2023, from https://en.wikipedia.org/w/index.php?title=Bayesian_network&oldid=1149465106

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine learning*, 62, 107-136.

Wikipedia contributors. (2022). Markov random field. In Wikipedia, The Free Encyclopedia. Retrieved 16:10, May 1, 2023, from https://en.wikipedia.org/w/index.php?title=Markov_random_field&oldid=1102260340

Kumar, A., Singh, S. S., Singh, K., & Biswas, B. (2020). Link prediction techniques, applications, and performance: A survey. *Physica A: Statistical Mechanics and Its Applications*, 553, 124289. https://doi.org/10.1016/j.physa.2020.124289

Wikipedia contributors. (2023). Link prediction. In Wikipedia, The Free Encyclopedia. Retrieved 16:09, May 1, 2023, from https://en.wikipedia.org/w/index.php?title=Link_prediction&oldid=1135736547