GRD Journal for Engineering | Volume 6 | Issue 2 | January 2021
ISSN- 2455-5703

Elements of Probabilistic Graphical Models for Machine Learning

Advait Pravin Savant
Department of Information Technology
Sardar Patel Institute of Technology

Abstract

A model is a representation of a real world phenomena. Probability theory gives us a framework to quantify uncertainty in a mathematically rigorous manner [4]. A random variable is a function on the elements in the sample space. A random variable takes on values and the act of a random variable taking on values can be described by a probability distribution. Conceptualizing entities in terms of random variables and representing the joint probability distribution over these entities is essential to statistical modelling and supervised machine learning. A graph is a data structure which consists of a collection of nodes and edges. Graphs as mathematical structures are studied in graph theory [9]. Using graphs to describe probability distributions over random variables gives us a potent way to map the flow of influence, interdependencies and independencies between the random variables. Graph based manipulations give us enhanced computational efficiency and add to the descriptive power, performance of our model. In this paper, we attempt to understand the essential concepts concerning probabilistic graphical models. We will see representation using Bayesian belief networks, Markov networks, techniques for inference and learning in probabilistic graphical models.
Keywords- Bayesian Belief Networks, Markov Networks, Clique Trees, Belief Propagation, Expectation Maximization, Computational Complexity, Relative Entropy

I. Representation

A. *Bayesian Belief Networks*

Consider the case where we have n random variables each of which takes on K discrete values. We see that we need (K^n – 1) parameters to represent the joint probability distribution of these variables. We shall see that the number of parameters needed to denote a joint distribution depends on how we factorize it and subsequently the independencies that we ascertain in our distribution. A graph provides a diagrammatic way to represent these concepts. From a standard valid factorization which does not make any independence assumptions, we get,

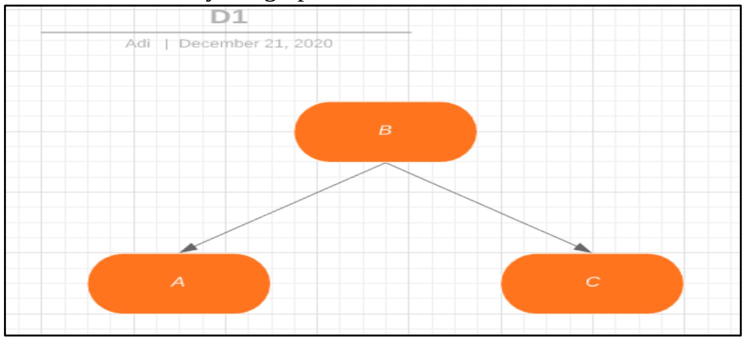
$$p(x_1, \dots, x_n) = p\left({}^{x_n}/_{x_{n-1}}, \dots, x_1\right) p\left({}^{x_{n-1}}/_{x_{n-2}}, \dots, x_1\right) \dots p(x_1)$$

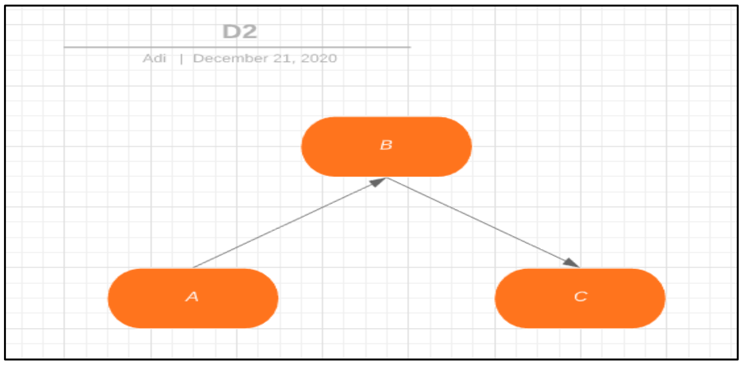
Conditioning x_n on x_{n-1} up-to x_1, we have K-1 values of x_n (normalization of the probability distribution) for each of the K^{n-1} different combinations of the variables it is conditioned on. Proceeding in this manner for x_{n-1} and ahead, we get a sum of a geometric progression which again gives us (K^n – 1) parameters. On the other hand, in the case when the n random variables are independent, the joint probability distribution will be given by the product of the individual probability distributions and with (K-1) parameters for each random variable, we will require n(K-1) parameters. To further show how the conditional independencies in our distribution affect our model size, consider standard multivariate linear regression [1] in which we take data points to be conditionally independent given the weights as we calculate the joint probability distribution and find the maximum likelihood estimate for the weights to obtain a compact parameterization for the Gaussian conditional distribution governing the input output relationship.

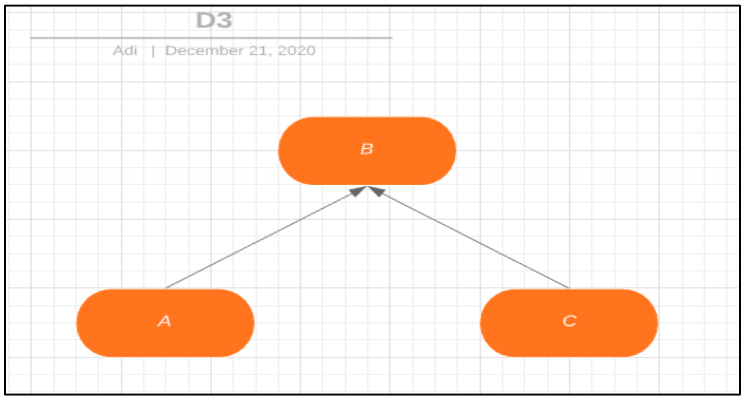
A graph data structure provides us a skeleton to represent the factorization of a joint probability distribution, model the conditional independencies and depict the flow of influence among variables represented as nodes in the graph. It enables us to perform probabilistic reasoning and inference in a computationally efficient manner. A Bayesian belief network also known as a Bayesian network is represented by a directed acyclic graph(DAG) with the nodes representing the variables and the corresponding joint probability distribution is represented as a product of conditional probability distributions of each variable given its parents in the graph.

$$p(X_1, \dots, X_n) = \prod_{i=1}^{n} p\left({}^{X_i}/_{Pa(X_i)}\right)$$

The directionality of the edges in the graph denotes how one variable influences the other and can intuitively be thought of as depicting the causal process [2]. We will talk about causality later. As we shall see, such a factorization based on a particular directed acyclic graph implies certain independencies and conversely, the independencies in a joint probability distribution determine whether or not it can be represented as a particular directed acyclic graph. Consider the Bayesian network for 3 variables as depicted in diagrams D1, D2 and D3. The edges are A-B and B-C, we consider all possible orientations of the edges and the corresponding factorizations from the directed acyclic graph.







Based on our definition of Bayesian networks, the factorization for D1 is given as,

$$p(A,B,C) = p(A/B)p(C/B)p(B)$$

Marginalizing over B,

$$p(A,C) = \sum\nolimits_{B} p(A/B)p(C/B)p(B)$$

This joint probability of A, C does not necessarily boil down as the product of individual probability distributions and hence A, C are dependent random variables wherein the knowledge of the state of one variable can influence the distribution over the other variable.
Conditioning on B,

$$p(A,C/B) = p(A,B,C)/p(B) = p(A/B)p(C/B)$$

Thus given B, A and C are conditionally independent. We can show a similar proof with the same results for D2, and reversing the edges of the graph in D2 will give us the same result.
Now consider diagram D3,

$$p(A,B,C) = p(A)p(C)p(B/A,C)$$

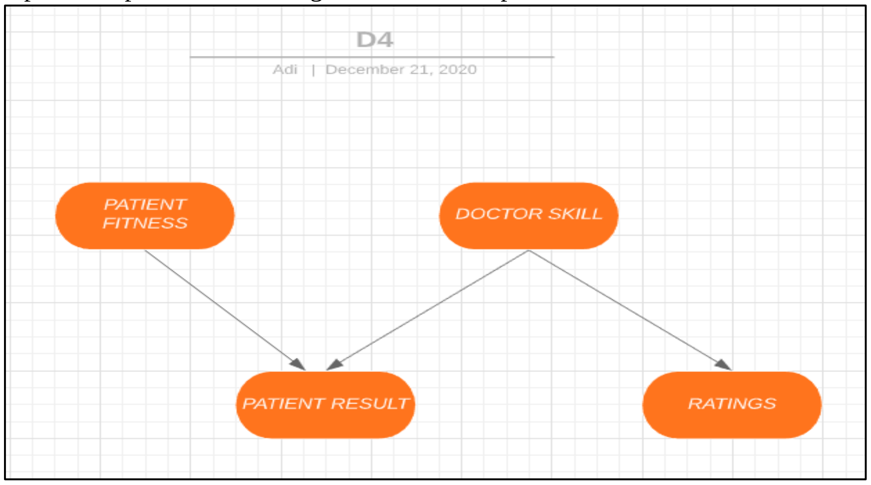
Marginalizing over B,

$$p(A,C) = p(A)p(C)$$

A and C are thus independent. But if we condition on B, we get,

$$p(A,C/B=b) = \frac{p(A)p(C)p(B=b/A,C)}{\sum_A \sum_c p(A)p(C)p(B=b/A,C)}$$

This expression does not boil down to any conditional independence, hence A and C are dependent given B. The directed acyclic graph depicted in diagram D3 where the two edges from A and C are incident on B is known as a v-structure and such sub-structures in a larger graph are essential to determine the conditional independencies implied by the directed acyclic graph. We develop intuition for the concept of d-separation from diagram D4 where a patient visits a doctor,



Patient fitness, doctor skill, patient result and doctor ratings are modeled as random variables with 3 discrete values each denoting qualitative gradation. Patient result is a random variable denoting the degree success. Given the doctor skill, the patient result and ratings are conditionally independent. Notice that this follows from our discussion of diagram D1. Knowledge of the patient fitness does not affect our probability distribution and hence beliefs about the doctor skill. But if the patient result is given and it is a high success, knowing that the patient was very unfit enhances the chances of the doctor being skilled. Conditioned on patient result, random variables doctor skill and patient fitness are dependent. Our intuitions follow from our discussion about diagram D3. We see that the presence of the v-structure blocks the flow of influence and hence dependence among the variables in the graph, however, if the conditioned on the variable at the head of edges in the v-structure, the influence flows through.
For a path X1-X2-…. XK in a DAG which represents a distribution conditioned on the set of variables Z, we call the path an active trail if in the corresponding path, there are no v-structures, or if there is a v-structure, any of the descendants of the concerned variable are present in the conditioning set Z [2].

Two variables in a graph are said to be d-separated if there is no active trail between them. d-separation of two variables implies the conditional independence of the variables [1]. d-separation of variables is a property of the graph and thus associated with each directed acyclic graph used to factorize a probability distribution are a set of conditional independence assumptions. These assumptions implied by the graph must be satisfied by the probability distribution in order for it to be factorized based on the graph. Conversely, if the set of conditional independence assumptions implied by a graph are satisfied by a probability distribution, it can be factorized based on the graph [1][2].

A Markov Blanket for a variable is the set of its parents, its children and the parents of its children as shown by the DAG. We can see that every variable is independent of all other variables in the distribution given its Markov Blanket.
From a formal perspective, the factorization and independence properties are the concepts defined in a Bayesian network. The directionality of the edge between two variables need not strictly imply causality. For more about Causality and related concepts such as the Simpsons paradox, Causal models and do-calculus, refer [2][3].

B. Markov Networks

We now move our discussion to Markov networks. Markov networks are defined over undirected graphs where there is no directionality in the flow of influence between two edge connected variables. We first need to define the concept of a factor. A factor over a set of random variables is a function which maps every configuration of states for the set of variables to a real number. We can define a corresponding factor table for each mapping in the discrete case. A factor over variables need not represent a probability distribution. A potential is a non-negative factor. When we take a product over two factors, we take a union of the variables and assign it to the multiplication of the corresponding two real numbers. In a Markov network, we have an undirected graph which is used to represent the joint probability distribution over a set of random variables. As with Bayesian networks, there are certain independence properties implied from the graph and the probability distributions are defined in such a way to satisfy these properties. Let C be a set of variables in the conditioning set. For two variables A and B in an undirected graph G, if there is no path from A to B which does not pass through variables in C, C separates A and B, random variables A and B are conditionally independent given C. This is the global Markov property. In light of the Hammersley-Clifford theorem [3][1], we define the joint distribution based on maximal cliques in a graph for the global Markov property to hold. A clique in an undirected graph is a set of nodes (here, random variables) all of which are connected to each other by an edge. A maximal clique is a clique in a graph to which no other node can be added. We can see that the based on the independence properties, the Markov Blanket for a variable in this network is the set of its neighbors.

For a Markov network, we define the joint probability distribution as a product of potentials defined on maximal cliques of the graph.

$$p(X_1, \dots, X_n) = \frac{1}{Z}\prod_{c=1}^{C} ø(X_c)$$

Here Z is the partition function to normalize the probability distribution.

$$Z = \sum_{X}\prod_{c=1}^{C} ø(X_c)$$

Consider the process of converting a Bayesian network to a Markov network. For a Bayesian network, we have a term for each variable given its parents in the joint distribution. If we connect the parents of that variable with a non directional edge, we get a factor over the variable and its parents. We can use this factor in the distribution of our Markov network and remove the directionality of the edges from the graph. This process of marrying the parents is known as moralization and the corresponding undirected graph is called a moral graph. Going from a directed graph based representation to an undirected graph based representation makes us lose some conditional independencies [2]. However, such a process is useful for inference in probabilistic graphical models, which is our next section.

II. INFERENCE

A. Variable Elimination

Given a probabilistic graphical model for a set of variables X, our focus now is to answer queries of the form p(Y/E=e) wherein we find the marginal probability distribution over the variable Y given evidence E. Without loss of generality, we deal with potential factors and represent the conditional probability distributions in a Bayesian network in terms of factors. Let W denote the variables in X not in E and not Y. We start with the variable elimination algorithm and then move to cluster trees/graphs.

$$p(y/E = e) = p(y, e)/p(e)$$

$$p(y, e) = \frac{1}{Z}\sum_{W} p(y, W, e) = \frac{1}{Z}\sum_{W}\prod_{c=1}^{C} ø(X_c, E = e)$$

Here Z is the summation of the numerator for all y,e in order to normalize the distribution.

$$p(e) = \sum_{y} p(y, e)$$

We see that we can just normalize the joint distribution with respect to all y values and divide by p(e) to obtain the conditional distribution. This gives us an easier partition function relative to p(y,e) or p(e).

$$p(y/E = e) = \frac{1}{Z}\sum_{W}\prod_{c=1}^{C} ø(X_c, E = e)$$

Here Z is the summation of the numerator for all y values.

The evidence E makes us reduce the number of terms in the factors. Only those factors where the variables in concern match the evidence are retained in the factor table mapping. The ordering of the variables while performing the summation plays an important role in our variable elimination procedure and depends on the structure of the Bayesian network. After reducing factors by the evidence, based on a certain ordering we follow the following procedure. For each non query variable, we eliminate the variable from the set of factors and sum over the factors containing the variable in the factor product. A new factor is generated which is placed in the factor product in place of the old factors. We multiply the remaining factors and continue with the next variable until all concerned variables are eliminated.

Consider a simplistic scenario to determine the complexity of variable elimination. Let there be n variables. For the elimination of a particular variable k, if there are m_k different factors involving k and the maximum size of each factor (number of rows in the factor table) is N, we have $(m_k-1)*N$ multiplications for k. Summing over all variables, the maximum number of different factors is m*(m + n), where we have one factor produced per variable apart from the m initial factors, we have O(Nm*) operations in multiplication. As far as addition is concerned, we have O(N) additions for every variable and thus O(Nn) additions in the complete process. Overall we see that the computational complexity depends on N, the number of rows in a factor table, if there are r variables which go into a factor and each variable takes d values we have $N=O(d^r)$ which grows exponentially with the scope of the factor.

We now analyze variable elimination as a graph transformation. Consider an arbitrary moralized graph for a Bayesian network with a certain elimination ordering. During variable elimination, for each variable, we multiply the factors containing the variable that factor and we sum over the variable for that factor product. In this process, we generate a new factor with a scope that does not contain the eliminated variable. We connect all the variables that appear together in the scope of this new factor with an edge. What we get as we continue this process for all variables is an induced graph. The induced graph has useful properties. The scope of every factor produced in the variable elimination process is a clique in the induced graph. Every maximal clique in the induced graph is the scope of a factor in the process. For any maximal clique C in the induced graph, let X be the variable to be eliminated first based on our choice of ordering. Once X is eliminated, no new neighbors are added and it already has the variables in the clique as its neighbors. Based on how we define an induced graph, X must have participated in factors with these neighbors. When these factors are multiplied together, we get a factor over the variables in our maximal clique C. Another property of the induced graph is that it is a chordal graph which contains no cycle of length greater than three [2]. The induced graph and the size of the largest maximal clique depend strongly on the elimination ordering, the size of the largest clique determines the size of the largest factor and is an indicator of the computational complexity of the variable elimination process. In practice, various heuristic cost functions are used based on a greedy search to determine the variable elimination ordering. The cost of a vertex can be the number of neighbors it has in the current graph, the number of values of the factor product to be summed over for that vertex to be eliminated, the number of fill edges added in the process of vertex elimination etc. [2]

B. Message Passing, Belief Propagation and Cluster Graphs

Consider a Markov network of variables in the form of a linear chain X_1-X_2-…-X_N. The joint probability distribution is given by,

$$p(X) = \frac{1}{Z} \phi_{1,2}(X_1, X_2) \dots \phi_{n-1,n}(X_{n-1}, X_n)$$

If we want to find the marginal distribution of X_P between X_1, X_N, we have to sum the joint over all other variables. If each of the n variables take on K values, simple Combinatorics shows us that the naïve implementation would be exponential in the length of the chain, $O(K^N)$. We can make use of the structure of the network in order to perform local computations in a systematic manner which enables us to improve the computational efficiency and eliminate redundant summations. Consider the summation as follows,

$$p(X_p) = \frac{1}{Z}\left[\sum_{p-1} \phi_{p-1,p}(X_{n-1}, X_n) \dots \sum_{X_2} \sum_{X_1} \phi_{1,2}(X_1, X_2)\right]\left[\sum_{p+1} \phi_{p,p+1}(X_p, X_{p+1}) \dots \sum_{X_{n-1}} \sum_{n} \phi_{n-1,n}(X_{n-1}, X_n)\right]$$

We have distributed the summations in an algebraically consistent manner that has some useful properties. It is easy to determine that the computational complexity of this summation is $O(NK^2)$. There is a recursive pattern in the summation as represented above. Consider the innermost term in the first bracket which is a summation over X_1 for a factor of X_1 and X_2. The result will be a factor of X_2. That result will be multiplied by the factor over X_2 and X_3 and the composite factor will be summed over X_2. We can think of that result from the first summation as a message passed along the chain from the first node to the second. Continuing forward, each node receives a message from its previous node, multiplies it by the next factor and sums over the variable in order to generate the next message to pass forward. In this manner, we will get a factor over X_P from as a message from the nodes in the summation in the first bracket. For the second bracket, we look at the variables ahead of X_P, we can think of a backward pass of messages in an analogous manner starting with X_N up-to X_{P+1} and finally a factor over X_P as a message from X_{P+1} from the backward pass. We see that the product of these two messages (factors over X_P from the forward and backward pass) divided by the partition function enables us to calculate the marginal probability distribution over X_P. The importance of conceptualizing messages is in the following insight. For any node X_M we can think of messages in the forward pass and the backward pass relative to it in order to find the marginal over X_M. For any generic forward pass starting with X_1, the messages between two consecutive nodes will remain the same regardless of the variable we are finding the marginal over. Similarly, for the backward pass starting with X_N, the message passed between some X_R and X_{R-1} will remain the same in the process of calculating marginal probabilities over X_P, X_M or any node behind them. So we can do one forward pass over the chain, one backward pass over the chain and we find marginal over all the variables in $O(NK^2)$ time complexity rather than repeating the message passing procedure for every variable. Now that we have the intuition for message passing, we will define clusters graphs, clique trees and their operations in order to describe the belief propagation procedure for inference in a probabilistic graphical model and we will see how this is related to variable elimination and inference as optimization.

A cluster is a node which is associated with a set of variables and a factor Ψ over a subset of these variables. Ψ can be a composite product of other factors over the variables in the domain of the cluster. Between two connected clusters is an edge, associated with each such edge is a sepset which is set of variables common between the two clusters. Two clusters talk about variables in their sepset. A message is sent between two clusters which is a factor over the variables in their sepset. Consider the message passed between neighbors' X and Y. Cluster X receives messages from its neighbors except neighbor Y and then based on this information and its intrinsic factor, it passes a message to Y. Let σ(i,j) represent the message passed between cluster C_i and C_j.

$$\sigma_{i\to j} = \sum_{C_i - S_{i,j}} \{\, \Psi_i \prod_{k \epsilon Nb(i)-j} \sigma_{k\to i} \,\}$$

We take the factor product of the messages(factors) received to C_i from its neighbors apart from C_j and multiply it with the factor associated with C_i, we than sum over all variables in C_i not in the sepset to send the corresponding message to C_j. As we shall see, clusters give us a way of re-parametrization of the joint probability distribution of a Markov network in terms of beliefs. We describe the belief propagation algorithm for cluster graphs. We construct the initial potentials associated with each cluster and initialize the message factors as unity. We select an edge based on some ordering about which we will discuss later. We than propagate the message along that edge and we compute the beliefs at the cluster node where the message is sent. Beliefs are given by,

$$B_i(C_i) = \Psi_i \prod_{k \epsilon Nb(i)} \sigma_{k\to i}$$

Such a process is continued iteratively until some convergence criterion which we will discuss. Cluster graphs have what is the running intersection property. For each pair of clusters C_i and C_j if the variable Y belongs to both C_i and C_j then there exists a unique path between C_i and C_j for which all clusters and sepsets contain Y. A cluster graph is said to have achieved convergence if the message passed between any two clusters remains the same across successive iterations. A cluster graph is said to be calibrated if the any two clusters agree about variables in their sepset. Meaning,

$$\sum_{C_i - S_{i,j}} B_i(C_i) = \sum_{C_j - S_{i,j}} B_j(C_j)$$

We will now show that convergence implies calibration. Let σ’(i->j) be the next message that cluster C_i will send to cluster C_j. Let σ(i->j) be the message sent previously. Consider the following manipulations,

$$\sigma'_{i\to j} = \sum_{C_i - S_{i,j}} \Psi_i \prod_{k\epsilon Nb(i)-j} \sigma_{k\to i}$$

$$\sigma'_{i\to j} = \sum_{C_i - S_{i,j}} \Psi_i \prod_{k\epsilon Nb(i)-j} \sigma_{k\to i} * \frac{\sigma_{j\to i}}{\sigma_{j\to i}}$$

$$\sigma'_{i\to j} = \frac{1}{\sigma_{j\to i}} \sum_{C_i - S_{i,j}} \Psi_i \prod_{k\epsilon Nb(i)} \sigma_{k\to i}$$

$$\sigma'_{i\to j} = \frac{1}{\sigma_{j\to i}} \sum_{C_i - S_{i,j}} B_i(C_i)$$

$$\sigma'_{i\to j} * \sigma_{j\to i} = \sum_{C_i - S_{i,j}} B_i(C_i)$$

At convergence,

$$\sigma'_{i\to j} = \sigma_{j\to i}$$

Therefore,

$$\sigma_{i\to j} * \sigma_{j\to i} = \sum_{C_i - S_{i,j}} B_i(C_i)$$

Similarly, we can show for cluster C_j,

$$\sigma_{j\to i} * \sigma_{i\to j} = \sum_{C_j - S_{i,j}} B_j(C_j)$$

Hence,

$$\sum_{C_i - S_{i,j}} B_i(C_i) = \sum_{C_j - S_{i,j}} B_j(C_j)$$

Let,

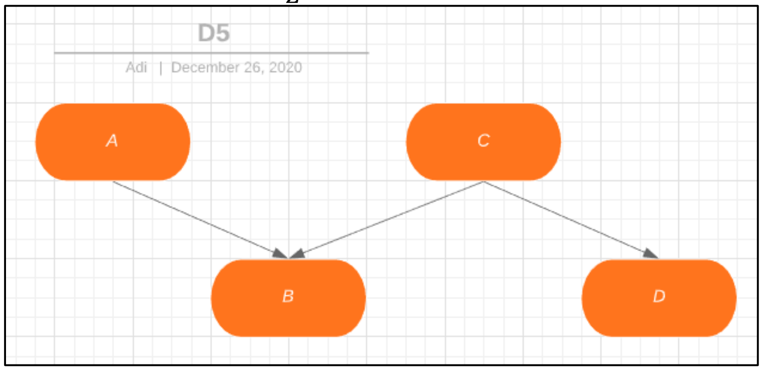
$$u_{i,j} = \sigma_{j\to i} * \sigma_{i\to j}$$

Based on our definition of beliefs, factors and the joint probability distribution, the un-normalized joint can be written as a re-parametrization of the form,

$$P'(X) = \frac{\prod_i B_i(C_i)}{\prod_{i,j} u_{i,j}}$$

As we shall see, an advantage of such an approach is in the efficient computation of marginal probabilities. Cluster trees arise naturally from variable elimination. Consider the following Bayesian network in diagram D5 and the corresponding joint probability distribution in terms of factors.

$$p(A,B,C,D) = \frac{1}{Z} \text{ø}_1(A)\text{ø}_2(C)\text{ø}_3(A,B,C)\text{ø}_4(C,D)$$



What happens is that each of the factors that goes into one step of the variable elimination process corresponds to a cluster and the factor produced after the elimination of the variable in that step corresponds to a message produced by that cluster. If the message produced by cluster X is used in the elimination process as a part of a composite factor in another step corresponding to cluster Y, there is an edge between clusters X and Y. This will be clear with an example, consider finding a marginal over D with respect to diagram D5.
First we sum over and eliminate A.

$$p(B,C,D) = \frac{1}{Z}ø_2(C)ø_4(C,D)\sum_A ø_1(A)ø_3(A,B,C)$$

Here the two factors inside the summation will go to the first cluster C_1, their product will be the composite factor for C_1. λ will be used to denote the messages.

$$p(B,C,D) = \frac{1}{Z}ø_2(C)ø_4(C,D)\lambda_1(B,C)$$

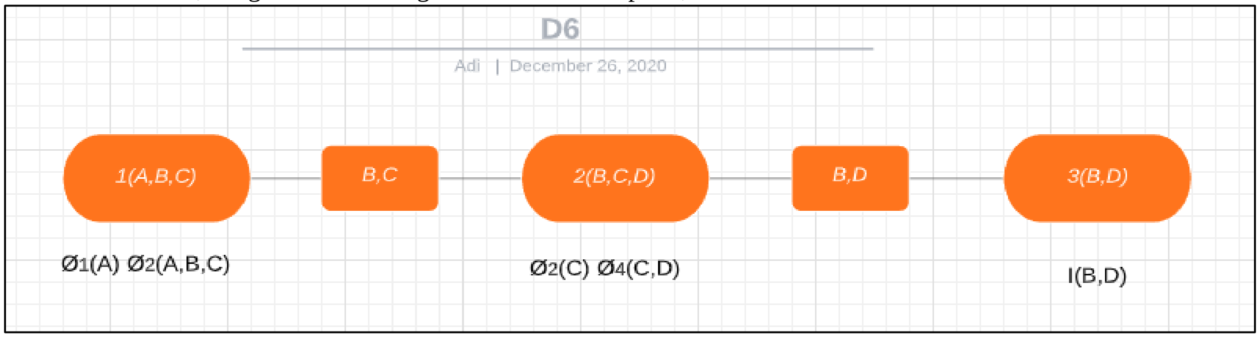
Next we eliminate C, Factors ø2, ø3 will go into cluster 2 and since the message generated by cluster 1 is used in the elimination of factors associated with cluster 2, we have an edge between clusters 1 and 2.

$$p(B,D) = \frac{1}{Z}\sum_C ø_2(C)ø_4(C,D)\lambda_1(B,C)$$

Consequently, λ_2 is the message produced by cluster 2 and for ease of explanation, we use it with an identity factor associated with cluster 3.

$$p(D) = \frac{1}{Z}\sum_D I(B,D)\lambda_2(B,D)$$

Based on our discussion, we get the following cluster tree and sepsets,



We see that we have factors associated with each clusters the composite product of which gives the Ψ_i values for each cluster i. We can observe that the variable elimination process we performed earlier to build the cluster tree is analogous to message passing and belief propagation from cluster 1 to 2 and then 2 to 3. This gives us p(B,D) at cluster 3 which can be marginalized to get p(D). The advantage of cluster trees is that if we pass a message from 1 to 2 and 3 to 2, then the beliefs will give us the marginal probabilities for variables B,C,D in cluster 2. This can be easily verified from the joint probability distribution and our definition of beliefs, messages passed. To generalize, with a cluster tree, we pick any root and we pass the messages starting from the leaves towards the root. When we compute the beliefs at the root, we get the un-normalized marginal probabilities of the variables in the root cluster factors. When we start from the leaves, we know that the message the leaves will pass is the final message and hence remains the same, satisfies convergence properties. The message passed by the penultimate cluster upstream after receiving the messages will be final. We have seen that convergence implies calibration. We proceed in this manner to calculate the final calibrated beliefs at the root. Once the root has received all the information from its downstream neighbors, the messages that we pass downstream from the root will also be final. In this manner, we can find the calibrated beliefs at each cluster in one pass towards the root from the leaves and one pass starting from the roots towards the leaves. We now provide intuition that these calibrated beliefs in a tree correspond to un-normalized marginal probabilities over the subset of cluster variables. If we compute the marginal probabilities, we need to sum the joint over all other variables. The joint is a product of the factors associated with each cluster. In a cluster tree, if a variable is eliminated at an edge, it cannot appear on the other side of the tree based on the running intersection property. The summation of variables over the joint can be spilt based on the factors in a way that represents message passing. A neat proof by induction is given in [2]. Since each factor generated in a variable elimination step is used once only, the cluster graph produced by variable elimination is necessarily a tree. Each factor in the variable elimination process also corresponds to a maximal clique in the induced graph, hence cluster trees are also known as clique trees.

C. Inference as Optimization

In many cases, finding the exact inference results on P(X) is intractable. As an approximation, we consider a set of tractable distributions Q and find a distribution Q' in that set which is the best fit for P based on some optimization criterion. In information theory [5], the relative entropy or the KL divergence between two distributions P and Q is given by,

$$KL(P,Q) = D(P||Q) = E_P ln\frac{P(X)}{Q(X)}$$

If the true distribution over X is P, and we use the distribution Q to represent X, KL(P,Q) is the additional amount of information required in nats to describe X. Although relative entropy is a-symmetric, it can be thought of as a distance measure between two distributions. D(Q||P) is also a valid distance measure. We use D(Q ∥ P) because as P occurs in the logarithm only, we can represent it as a sum of factors which makes our calculations for Q easier, and we need not run inference on P. We represent Q in terms of the cluster beliefs B, sepset beliefs u subject to constraints as we have discussed. We then solve a constrained optimization problem for minimizing D(Q||P) using the method of Lagrange multipliers [8]. The expectation operation is distributive over the sum [4], for each cluster, the ln(Ψ)term is reduced to a sum of ln(ø) terms. For a term from a particular cluster, when we take the expectation with respect to Q, the summations of variables in probability distribution as we take the expectation, ensure that the expectation is done with respect to the beliefs in Q. The beliefs represent the marginal probabilities for the variables in the cluster. For that cluster, the term we are taking an expectation over is a function of a subset of those variables only. Based on such a formulation, the solution of the constrained optimization problem gives us the same results as belief propagation [2]. We have seen that exact inference of the marginals is possible in a cluster tree. If there are loops in the cluster graph, convergence is not guaranteed, the technique of loopy belief propagation and its analysis using the calculus of variations can be found in [1][2].

III. LEARNING

A. Bayesian Learning for Bayesian Networks

For a Bayesian network, we first look at the maximum likelihood estimate for the parameters ϴ which govern the conditional distributions coming from the network. We assume that the n individual data points in the training data D are independent and sampled from the same distribution. The likelihood L(ϴ,D) is given by,

$$L(\Theta, \mathrm{D}) = \prod_n p(x[n]/\Theta)$$

For all the variables i based on the factorization,

$$L(\Theta, \mathrm{D}) = \prod_n \prod_i \left(\left(p(x_i[n]) \middle/ Pa(p(x_i[n])\right)/\Theta\right)$$

We make the global independence assumption which entails that the parameters governing each of the conditional distributions in the joint are independent and the corresponding conditional distribution depends on those specific parameters only. We also make the local parameter assumption that given a set of parent states, the parameters governing the conditional distribution of a variable are independent from the parameters governing the conditional distribution of that variable for another set of parent states. we get,

$$L(\Theta, \mathrm{D}) = \prod_i \prod_n \left(\left(p(x_i[n]) \middle/ Pa(p(x_i[n])\right)/\Theta_i\right)$$

This enables us to find using optimization, the parameters for each of the conditional distribution separately. In the discrete case, for a variable state x, parent state u, in the likelihood product, $\Theta_{x/u}$ will be raised to the power of M[x,u] where M[x,u] represents the number of times x,u occur together in the training data set. We take the log of the likelihood for our optimization problem since the log is a monotonic function. This gives us a summation of parameters from the product. This summation has a lot of independent terms which makes differentiation easier. While performing optimization, we have to take into account that for each parental state configuration u, the conditional distribution of variable x should sum up to 1. We use Lagrange multipliers [8] to represent these constraints. We take the partial derivatives with respect to the Lagrange multipliers and the parameters and equate them to zero, solving for the parameters we get,

$$\Theta_{x/u} = \frac{M[x,u]}{\sum_{x'} M[x',u]} = \frac{M[x,u]}{M[u]}$$

We see that the maximum likelihood estimate essentially gives us the frequency counts. $\Theta_{x/u}$ is a multinomial distribution over the variable states based on the sufficient statistics M_1, M_2 etc which are the counts of each variable state and the parental state u. In Bayesian learning we introduce a prior over the parameters and treat the parameter as a random variable. We have,

$$p(\Theta/D) = p(D/\Theta)p(\Theta)$$

We introduce a Dirichlet prior [1] with hyperparameter set α for the parameter values which govern the conditional probability distributions. For a particular conditional distribution, for a particular parental state configuration we have,

$$p(\Theta) = \frac{1}{Z}\prod_i \Theta^{\alpha_i - 1}$$

Here, Z is given in terms of gamma functions of α. [1]

We know from Bayesian statistics [1][4] that for a multinomial distribution with sufficient statistics (M_1,M_2,..M_K) and a Dirichlet prior(α_1,α_2,..α_k), the posterior will be a Dirichlet distribution (M_1+ α_1, M_2+ α_1 ….M_k+ α_k). Dirichlet hyperparameters represent the strength of prior beliefs, their influence reduces with increasing training data. A wise choice of hyperparameters enables us to prevent overfitting on the training data and helps the model to generalize better. In a Markov network, the partition function and couples the various parameters and there is no closed form solution of maximum likelihood. However, we do have neat techniques for certain kinds of Markov network specifications. For learning the parameters of a Markov network, refer [2].

B. Learning with Hidden Variables

For reasons ranging from model sparsity, better parameterization and non-availability of data, the hidden variables in a Bayesian belief network are not observed. Let a Bayesian network with a probability distribution p be parameterized by Ɵ. Consider a set of variables x where o is the set of observed variables and h is the set of hidden variables. Let p(o,h/ Ɵ) represent the joint probability distribution. Consider the relative entropy [5] between an arbitrary distribution q(h/o) and p(h/o, Ɵ).

$$KL(\ q(h/o)||\ p(h/o,\Theta)\ ) \geq 0$$

$$E_{q(h/o)}[ln(q(h/o) - ln(p(h/o,\Theta)] \geq 0$$

$$E_{q(h/o)} \ln q(h/o) - E_{q(h/o)} ln \frac{p(h,o/\Theta)}{p(o/\Theta)} \geq 0$$

$$E_{q(h/o)}\ ln\, q(h/o) - E_{q(h/o)}\ ln\, p(h,o/\Theta) - \ E_{q(h/o)} ln\, p(o/\Theta) \geq 0$$

Notice that the ln p(o/ Ɵ) does not depend on h and q(h/o) is a distribution on h, hence,

$$E_{q(h/o)} ln\, p(o/\Theta) = ln\, p(o/\Theta)$$

We thus obtain a lower bound on the log likelihood,

$$ln\, p(o/\Theta) \geq -\ E_{q(h/o)}\ ln\, q(h/o) + E_{q(h/o)}\ ln\, p(h,o/\Theta)$$

On the right hand side, the first term is the entropy of q(h/o). It depends on our choice of conditional distribution q. The second term is known as the energy term where the terminology is borrowed from statistical physics [2], it is computationally tractable to maximize for some q(h/o) as ln(o,p/ Ɵ) is essentially the entire joint distribution. In the KL divergence, equality holds when the two distributions are the same, in our case,

$$q(h/o) = p(h/o,\Theta)$$

For n=N data points,

$$\ln(O/\Theta) = \sum_n \ln p(o^n/\Theta)$$

$$ln(O/\ \Theta) \geq \sum_n -E_{q(h^n/o^n)}\ ln\, q(h^n/o^n) + \sum_n E_{q(h^n/o^n)}\ ln\, p(h^n,o^n/\Theta)$$

What we do in the expectation maximization algorithm [1] is we follow an iterative procedure based on these propositions since the log likelihood by itself has no closed form solution due to the sum over hidden variables inside the logarithm. We initialize the parameters Ɵ. At each iteration, we find the conditional distribution q(h/o)=p(h/o,Ɵ). Using this distribution and keeping it fixed, we maximize the energy term with respect to the parameters to find a set of parameters. We then use this new set of parameters in the next iteration to find q(h/o)=p(h/o,Ɵ). Based on this new q, we again maximize the energy function with respect to to parameters to find a new set of parameters. We repeat the procedure until convergence [1][3].

For a Bayesian network, we define the distribution q(h/o) as a distribution over x using the following manipulation,

$$q(x) = q(h/o)\delta(o(x),o)$$

$$Energy\ term = E_{q(x)} \log(p(x/\Theta))$$

$$E_{q(x)} \log(x/\Theta) = \sum_{all\ states\ of\ vector\ x} q(x) * \log(p(x/\Theta))$$

Using the factorization of the Bayesian network inside the log,

$$E_{q(x)} \log(x/\Theta) = \sum_{all\ states\ of\ vector\ x} q(x) * \left(\sum_i \log p(x^i\ /pa(x^i))\right)$$

We are essentially summing over all variables in x in the outer sum. The key here is in manipulating the joint distribution q(x) inside the outer summation to suit the inner summation. For each i, we take the q(x^i/pa(x^i)) and multiply it with log(p(x^i/pa(x^i)). In this context, we will use the term ‘influence’ to demonstrate edge directionality from parent to child. The terms consisting of the non-descendants of x^i which influence its parents are placed before this term. Let A be the set of non-descendants apart from parents, variables which influence the parents. The terms consisting of the children of x^i and their descendants, terms which are influenced by x^i, terms which have variables that do not influence the parents are placed after this term. Let C be the set of such variables. We use the independencies in the joint summation to perform local operations. Note that if a variable is a child of x^i or its descendants, since the Bayesian network is a directed acyclic graph, it cannot influence the parents of x^i. This will make it clear,

$$E_{q(x)} log(x/\Theta) = \sum_{pa(x^i)} \sum_A f_1(pa(x^i),A) \sum_{all\ states\ of\ x^i} q(x^i/pa(x^i)) * \log p(x^i\ /pa(x^i)) \sum_C f_2(C,x^i)$$

We partition the terms in q into f_1, f_2 and q(x^i/pa(x^i)). The sum over C will be one by properties of variable elimination and probability axioms. The sum over all states of x^i will give us an expectation, we can also see that the summation over A will give us p(pa(x^i)).

$$E_{q(x)} log(x/\Theta) = \sum_{pa(x^i)} q\left(pa(x^i)\right) * E_{q(x^i/pa(x^i))} \log p(x^i\ /pa(x^i))$$

$$Energy\ term = E_{q(pa(x^i)}[E_{q(x^i/pa(x^i))} \log p(x^i\ /pa(x^i))]$$

We have to maximize the energy term with respect to the parameters as in the expectation maximization steps discussed earlier. It is the distribution p which is dependent on the parameters, we add a term based on distribution q for the same optimization and rewrite the optimization problem as a minimization of L as follows,

$$L = E_{q(pa(x^i)}[E_{q(x^i/pa(x^i))} (\log q(x^i/pa(x^i) - \log p(x^i\ /pa(x^i))]$$

The inner expectation is a KL divergence which is always greater than or equal to zero. The outer expectation is probability weighted sum of independent KL divergences. In order to minimize this, we take the KL divergences to be zero and hence, the new values for the probability distribution p for x given its parents will be the current values from the distribution q. Based on the new p values, we find the distribution q and repeat the expectation maximization process as we discussed earlier.

C. Structure Learning

We first discuss the concept of mutual information [5]. The mutual information between two random variables X and Y is defined as the relative entropy of the joint distribution p(x,y) and the product distribution p(x)p(y).

$$I(X;Y) = E_{p(x,y)} \log \frac{p(x,y)}{p(x)p(y)}$$

H(X) denotes the entropy of random variable X. It can be shown that,

$$I(X;Y) = H(X) - H(X/Y) = H(Y) - H(Y/X)$$

Information is the resolution of uncertainty. The entropy of a random variable denotes the measure of uncertainty inherent in the act of the random variable taking on its possible outcomes. The mutual information X between and Y is the reduction in the uncertainty of X given the knowledge of Y. If Y contains no information about X, there is no reduction in uncertainty of X based on Y and I(X;Y) is zero. If given Y, we get a lot of information about X, the mutual information value is very high.

A lot of times, we have prior domain knowledge about the structure of the Bayesian network. However, in many cases, we need to develop mechanisms to produce a good structure of a graph for the Bayesian networks from the data set. For each graph, we can calculate the maximum likelihood estimate of the parameters and then we can establish a score based on the maximum likelihood estimate. We can search for the graph with the highest score. The dataset D has M data points and there are N variables in the network. Based on this dataset, we have a maximum likelihood estimate Ɵ‘ for parameters which define the constructed distribution p’. Consider score L(G,D) from the likelihood estimate for graph G. This is the value of the maximum likelihood. This follows from our discussion in section 3.1.

$$l(G,D) = \sum_{i=1}^{N} \left(\sum_{u_i \epsilon Pa(x_i)} \sum_{x_i} M[u_i, x_i]\ log\Theta'_{x_i/u_i} \right)$$

$$l(G,D) = M \sum_{i=1}^{N} \left(\sum_{u_i \epsilon Pa(x_i)} \sum_{x_i} \frac{M[u_i, x_i]}{M} \log \mathrm{p}'(x_i/u_i) \right)$$

$$l(G,D) = M \sum_{i=1}^{N} \left(\sum_{u_i \epsilon Pa(x_i)} \sum_{x_i} p'(x_i, u_i) \log \mathrm{p}'(x_i/u_i) \right)$$

$$l(G,D) = M \sum_{i=1}^{N} \left(\sum_{u_i \epsilon Pa(x_i)} \sum_{x_i} p'(x_i, u_i) \log \mathrm{p}'(x_i/u_i) - \sum_{x^i} p'(x_i) logp'(x_i) + \sum_{x^i} p'(x_i) logp'(x_i) \right)$$

Using the properties of the summation operator,

$$l(G,D) = M \sum_{i=1}^{N} \left(\sum_{u_i \epsilon Pa(x_i)} \sum_{x_i} p'(x_i, u_i) \log \mathrm{p}'(x_i/u_i) - \sum_{u_i \epsilon Pa(x_i)} \sum_{x^i} p'(x_i, u_i) logp'(x_i) - H(x_i) \right)$$

$$l(G,D) = M \sum_{i=1}^{N} \left(\sum_{u_i \epsilon Pa(x_i)} \sum_{x_i} p'(x_i, u_i) \log \frac{\mathrm{p}'(x_i/u_i)}{p'(x_i)} - H(x_i) \right)$$

$$l(G,D) = M \sum_{i=1}^{N} \left(\sum_{u_i \epsilon Pa(x_i)} \sum_{x_i} p'(x_i, u_i) \log \frac{\mathrm{p}'(x_i, u_i)}{p'(x_i)p'(u_i)} - H(x_i) \right)$$

$$l(G,D) = M \sum_{i=1}^{N} \{ I(x_i; Pa(x_i)) - H(x_i) \}$$

Where I(x_i, pa(x_i)) is the mutual information between x_i and pa(x_i). This makes sense based on our discussion of mutual information, the score is higher if a variable is correlated with its parents. To this score function L(G,D) , we add terms which penalize model complexity in congruence with the minimum description length principle in order to prevent overfitting and help generalization [2]. Based on this cost function, we search through the space of the graphs with techniques such as greedy hill climbing and simulated annealing [7] based on atomic operations for edge deletion, edge reversal and edge addition. Computational efficiency is achieved by calculating the change in the value of the score functions for two graphs adjacent in the search space rather than calculating the entire score for each graph [2].

IV. CONCLUSION

We have seen how probabilistic graphical model provides a framework to efficiently describe the joint probability distribution of a set of random variables in a way that captures the flow of influence and independencies between them. Graphs provide a natural and a mathematically rigorous way to represent the interaction between random variables and parameterize their distributions. We have seen how efficient inference can be performed in these models by conceptualizing message passing and belief propagation. We have seen the process of learning the parameters of a probabilistic graphical model in a Bayesian setting. We reviewed concepts from information theory as required and seen the process of learning with hidden variables, learning the structure of the network. Markov networks and Bayesian networks can be used to model a wide variety of phenomena and have been used successfully in computational genetics [1], computer vision tasks such as image segmentation etc [1][2] and natural language processing tasks [3]. With the increase in popularity of deep learning, probabilistic graphical models in deep learning are an active area of research [6]. As a framework for learning and intelligence, connectionism is in harmony with the concept of a graph and it is good to see computational graph theoretic approaches work in machine learning. In this line of inquiry, I find promise in avenues such as graph dynamical systems, graph neural networks to represent and simulate a variety of real world phenomena.

REFERENCES

[1] Christopher Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
[2] Daphne Koller and Nir Friedman, Probabilistic Graphical Models: Principles and Techniques, MIT Press, 2009.
[3] David Barber, Bayesian Reasoning and Machine Learning, Cambridge University Press, 2012.
[4] Vijay Rohatgi and Md. Ehsanes Saleh, An introduction to probability and statistics, Wiley, 2017.
[5] Thomas Cover and Joy Thomas, Elements of information theory, Wiley, 2016.
[6] Yoshua Bengio, Ian Goodfellow and Aaron Courville, Deep Learning, MIT Press, 2016.
[7] Stuart Russell and Peter Norvig, Artificial Intelligence, Pearson, 2015.
[8] Edwin Chang and Stanislaw Zak, An introduction to optimization, Wiley, 2013.
[9] Narsingh Deo, Graph Theory with applications to engineering and computer science, PHI, 2019.
[10] P.Bini Palas. "Machine Learning Intelligence based Guiding System." Global Research and Development Journal For Engineering [ICIET'16], K. L. N. College of Engineering Anna University, (July 2016) : pp. 573 - 577.
[11] A. Surya, S. Suma, M. U. Surya Prakash, Yudhishtran. "Medicine Prescription using Machine Learning." Global Research and Development Journal For Engineering 3.5 (2018): 1 - 6.
[12] Snehal Mastud, Shweta Pandit, Ankita Mungekar, Sachin Desai. "Comparative Study of Classification of Cancerous Profile using Deep Learning and Classification Algorithms." Global Research and Development Journal For Engineering 5.5 (2020): 18 - 23.