A structured probabilistic model describes probability distributions using a graph (graph theory) to describe which random variables interact with each other directly; these are also called graphical models

When using a probabilistic model, you cannot ignore most of the inputs (as opposed to traditional classification where most inputs are useless)

Some examples of tasks like these include:

* Density estimation - given an input x, the machine learning system returns an estimate of the true weight / density p(x) under the data generating distribution; this requires complete knowledge of the input
* Denoising - a corrupted image x~ must be completely cleaned, at all input locations
* Missing value imputation: given some of the elements of x, the model must propose a probability distribution for the unobserved values of x
* Sampling - the model generates new samples from the distribution p(x)

If we have an input with n discrete variables which can each take on k different values, then modeling this using a look-up table is infeasible since it requires storing kn parameters

* There is not enough memory
* Failure of statistical efficiency, since there will probably never be enough training data
* High runtime / cost of inference if we use our model of the joint distribution P(x) to infer marginal distributions such as P(x1) or P(x2 | x1)
* High cost of sampling - if we want to sample some particular u ~ U(0, 1), we must sum through the table adding up the probabilities until they exceed u and then return the corresponding output

The table based approach looks at all possible approaches, which is not effective since probability distributions we encounter tend to be simpler than this, with most variables only influencing each other indirectly

Structured probablistic models provide a framework for modeling only direct interactions, which is good when reducing unnecessary parameters (which would otherwise add redundancy)

**Using Graphs to Describe Model Structure**

Direct interactions imply other indirect interactions, but only the direct ones need to be modeled

Graphical models can be based on directed acyclic graphs, or models based on undirected graphs

*Directed Models*

One kind of structured probabilstic model is called the directed graphical model, belief network, or Bayesian network

Drawing an arrow from a to b means we define the probability distribution of b in terms of a conditional distribution given a, p(b | a)

A directed graphical model on variables x is defined by a directed acyclic graph G with the random variables as the vertices and a set of local conditional probability distributions

The probability distribution over x is given by 

Suppose we represented time by discretizing time ranging from minute 0 to minute 10 in 6 second chunks; t0, t1, and t2

t0 t1 and t2 each is a discrete variable with 100 values, which would require 999,999 values to store all possible configurations of

If t2 is dependent on t1 and t1 is dependent on t0, then we can make the table representing t0 99 values, the table representing t1 9,999 values, and the table representing t2 9,999 values, reducing memory requirements by a factor of 50

To model n discrete variables, each with k values, the cost of a table scales like O(kn)

With a graphical directed model, it scales in O(km) where m is the maximum number of variables between the input and output

If m << n, we get dramatic savings

Few parents in the graph causes fewer parameters for the distribution

Restrictions on the graph, such as being a tree, allows the conditional distribution to be calculated efficiently

Suppose Bob takes the same time to run a lap, regardless of Alice's starting time; this requires only O(k) parameters to encode, but Bob's time to run is still directly dependent on Alice (since Bob's absolute finishing time = t0 + dt) => our graph must contain an arrow from t0 to t1

We cannot encode the fact that Bob's personal running time (dt) is independent from others in the graph; instead, we must encode this in the conditional distribution itself

The conditional distribution is no longer a k x k-1 element table, but now is a slightly more complicated formula with only k - 1 parameters

The directed graphical model places no constraint on how we define our conditional distributions

*Undirected Models*

Another popular way to describe structured probabliistic models is through undirected graphs, Markov Random Fields (MRFs), or Markov Networks

We use directed models when there is a clear cause/effect between two random variables

If there is nothing like that, or it is unclear, then we use an undirected model

An undirected graphical model is a structural probabilistic model defined on an undirected graph

For each clique (subset of connected vertices) C in the graph, a factor 𝜱(C) - the clique potential - measures the joint probability of the edges in the clique being in their possible states

The factors are non-negative, and together define an unnormalized probability distribution



Cliques don't have a structure, so they probably are not valid probability distributions

*The Partition Function*

We must normalize the probability distribution in order to use it



Z results in the probability distribution summing to 1:



If the clique potential is constant, then so is Z

Z is defined as the partition function

Z is usually intractable to compute exactly, so we approximate its value

It is possible to specify the factors in such a way that Z des not exist

This happens if the variables are continuous and the integral over their domain of p~ diverges

For example, suppose we want to model a single scalar variable with clique potential x2

Z = ∫x2dx ← For each state x, get its probability using its clique potential x2

Since this integral diverges, there is no probability distribution corresponding to 𝜱(x) and it cannot be normalized

Sometimes, the choice of the parameter of 𝜱 determines where it diverges

𝜱 can only be normalized if 𝜷 is positive and the following case is used:



In undirected graphs, the clique function determines the probability distribution

One key idea to keep in mind is that the domain of each of the variables has a big effect on which probability distributions the clique functions correspond to

For example, 𝜱(xi) = exp(bixi) can be a sigmoid, softmax, etc. depending on the domain of x

Many interesting theoretical results depend on the assumption 

To enforce this condition, we can use an energy-based model where p~(x) = exp(-E(x)), where E(x) is the energy function

Since exp(z) is always positive, for any value of z, this allows for unconstrained optimization (we do not have any arbitrary constraints)

Any distribution given by the form exp(-E(x)) is called a Boltzmann distribution, while EBMs are referred to as Boltzmann machines

Boltzmann machines usually refer to EBMs with latent variables

Since exp(a)exp(b) = exp(a + b), different cliques correspond to different terms of the energy function; multiplying the probability of two cliques a and b is the same as adding them in E

Therefore, an EBM can be thought of as a special kind of Markov network

It is basically a product of experts, where each term is a factor in the probability distribution determing whether the corresponding soft constraint has been satisfied

When all the factors are combined through a multiplication of probabilities, this imposes a hard, high-dimensional, complicated constraint

Fig 16.5

Shows a graph indicating how the energy function can be written as a function of per-clique energy functions, since E(a,b,c,d,e,f) = Eb,c (b,c) + … + Ee,f(e, f)

Each 𝜱 can be obtained by setting the 𝜱 of the exponential to the negative of the corresponding energy function: 𝜱(a, b) = exp(-E(a, b))

Note that the negative sign in an EBM does not have a functional purpose

Many algorithms operating on probabilistic models do not need pmodel(x) but only log pmodel(x)

For EBMs with latent variables h, these algorithms are phrased in terms of the negative of this quantity, the free energy



**Main idea of cliques: they can represent probability distributions p(x), and using the structure of an EBM imposes the logical constraint that p(x) > 0; cliques can be combined**

*Separation and D-Separation*

Sometimes, we need to know which variables indirectly interact; in other words, we would like to know which variables are conditionally independent given another set of variables

Using undirected models / graphs, it is easy to see conditional dependencies

Conditional independence implied by the graph is known as separation

A set of variables is separated from a set of variables B if A is independent given S

If a is related to b through only a path with unobserved variables, then they are not separated and this is called an active path

Fig 16.7

Shaded in variables indicated observed variables

We can see an active path betwen two edges in the model, a and d

The only path from a to c is blocked by a shaded node b, so a and c are separated from each other given c

For a directed model, we say two variables are d-separated, where d stands for dependence

The conditions are the same as an undirected model; two variables are d-separated if no active path exists between them (only observed variables between them)

A graph cannot represent all independencies

Some independencies are context-specific, meaning they depend on the values of the variables (e.g how memory can capture information only when a certain bit is turned on)

Fig 16.8

V-structure / collider case is when two variables in a directed model a and b both point to s

This is known as the explaining away effect: in this case, the path is active if s or its descendants are observed

Fig 16.9

a and b are d-separated given the empty set, meaning there is no active path between them

a and b are not d-separated given c, which means there is an active path

**Main idea of separation: If two variables are connected by observed variables in a graph, they are separated because the observed variables are treated as conditionals making the two variables conditionally independent**

*Converting Between Directed and Undirected Graphs*

A model is not inherently undirected or directed; it can be represented by both graphs

The better type of graph may be the one which uses less edges and captures more independencies

There are other factors which play a role, such as computation

Directed models, for example, are more computationally efficient when drawing samples from models

In the worst case, any probability distribution can be represented using a complete graph

A complete graph is any directed acyclic graph where we impose an ordering on the random variables such that each variables has all other variables which precede it as ancestors in the graph

In the undirected case, a complete graph is a graph which contains a single clique encompassing all the variables

Fig 16.10

Shows example of complete graphs

In the directed case, the graph is not unique since the order of variables can take many forms

The direction in the directed graph is given by a node from one variable to the variable which follows it

There is a factorial number of complete graphs

Complete graphs however capture no independencies, so they are not very useful since the point of a graph is to capture indirect interactions between variables

Directed models are better than undirected models for capturing independencies between some distributions, and vice versa

Directed models can capture immoralities, which occurs when a and b are parents of a random variable c, with no edge connecting a or b in any direction

To convert a directed model w/ graph D into an undirected model, create a new graph U

Connect x and y in U if there is an edge in any direction between them, or if they both point to the same child z (even if they are not directly connected)

The resulting U is a moralized graph

Likewise, an undirected model can capture independencies a directed model cannot

D cannot capture the same independencies as U if U contains a loop, unless that loop also contains a chord

A loop is undirected with a connection between the last variable back to the first variable

A chord is a connection between two non-consecutive variables; chords are required to make undirected graphs w/ loops into directed graphs

Adding chords discards some implied independencies in U

Fig 16.11

Shows how an undirected model must have a clique encompassing a, b, and c if both a and b share the same child c; this means a and b must be connected in order to capture the dependence

We can also see how moralizing the graph can result in a quadratic number of new direct dependencies / edges, which loses implied independencies

When chords are added to the undirected graphs, the resulting graph is chordal

When assigning direction in the chordal graph, we must make sure there is no directed cycle or the resulting distribution will not be valid

Fig 16.12

Triangulating the graph by adding a connection to loops greater than a length of 3 allows an undirected model to be converted to directed

To avoid directed cycles, we randomly assign each node an ordering and always point in the direction from the earlier node to the later node

One way to assign direction in D is to assign an order to the random variables and make connections from nodes which come earlier in the ordering to nodes which come later

*Factor Graphs*

Factor graphs are a way of resolving ambiguity in undirected models

The scope of every 𝜱 function (density for joint probability of a clique) must be part of some subset of some clique C in the graph

Ambiguity arises for whether or not the factor encompasses C - a clique containing 3 nodes may correspond to a factor over all 3 nodes, or 3 factors each only containing a pair of the nodes

Factor graphs explicitly represent the scope of each 𝜱

A factor graph is an undirected model which contains a bipartite (2 part) undirected graph

In a factor graph, circular nodes represent random variables and square nodes represent factors; circular and square nodes are connected if the random variables are arguments to 𝜱

Random vars cannot be connected to random vars, and same with factors

Fig 16.13

Shows how factors are used in undirected model

The left figure does not specify information about the factors

The middle figure has one factor which takes in all 3 random variables as arguments

The right figure shows the most efficient implementation of factors in an undirected model, with multiple factors each taking in a unique set of 2 nodes

**Main idea: directed and undirected models each have their own areas which they work well in, in terms of capturing dependencie; different graphs capture different information, and efficiency is based on the situation**

**Sampling from Graphical Models**

Directed models can use a simple and efficient procedure called ancestral sampling to produce samples from a model's joint distribution

It creates a topological ordering where xi is the parent of xj in the graph if j is greater than xi for all i and j

In other words, first we sample x1 ~ P(x1) and then sample P(x2 | PaG (x2)) until xn

If each conditional distribution is easy to sample from, then the whole model is

The topological sorting operation guarantees that we can read the conditional distribution and sample from them in the correct order

Ancestral sampling only works in directed graphs

Furthermore, ancestral sampling does not support every conditional sampling operation

If we want to sample a subset of variables given some other variables we require the conditioning variables come before the variables to be sampled

In this case, we sample from the local conditional distribution defined by the model

Otherwise, the conditional distributions we need to sample from are the posterior distributions given the observed variables which may not be specified or which may be hard to derive

Ancestral sampling only works in a directed model

Converting undirected models to directed models may make computing the marginal distribution over the resulting graph intractable or introduce so many edges in the new graph that sampling becomes intractable

Ancestral samping with undirected models requires resolving cyclical independencies, which connect basically every variable

This is an expensive multipass operation

The simplest way to do this with an undirected model is to use Gibb's sampling where p(xi | x-i) is conditioned on all other variables x-i

Calculating a fair sample p(x) cannot be done in one pass , so we must keep resampling and making passes until the model converges to a correct distribution

**Main idea of ancestral sampling: it works with directed models, and draws conditional samples based on all the variables before the one being sampled. Conditioning variables must come before the sampled variables, or we must use observed variables from the posterior.**

**Advantages of Structured Modeling**

With SPMs, learning and inference are quicker, due to computational gains from not modeling indirect interactions

SPMs also allow us to separate representation of knowledge from given knowledge

This allows us to create learning algorithms which are applicable to broad classes of graphs, each with their own features

Furthermore, we can capture our own beliefs by choosing what relationships to capture

We can also combine the algorithms to get a Cartesian product, to get many perpsectives

**Learning About Dependencies**

Dependencies can be captured in deep learning through latent variables h

Indirect connections between vi and vj are captured through direct interactions between vi and h and between vj and h

Without latent variables, you would need to use a Markov model with a large clique

The number of parameters increases exponentially with the size of the clique, and the exponential number of parameters requires much more data, so it is computationally and statistically expensive

When the model is intended to capture dependencies between visible variables w/ direct connections, it is infeasible to connect all variables so the graph must only connect variables which are tightly coupled

This general problem is called structure learning

In structure learning, the model is penalized based on its complexity and training accuracy

A greedy search is performed such that the expected result will improve the model's score

Using latent variables h instead of adaptive features can avoid the need to perform searches/iteratively learn structures

Furthermore, it allows us to capture indirect interactions with a fixed structure

Using parameter learning techniques, we can learn a model with a fixed structure which imputes/takes form of the right structure on the marginal p(v) where v is a vector of variables

v can be represented by the latent variable h

In a Gaussian mixture, the latent variable corresponds to the category the input example was drawn from, allowing h to be used for classification

Simple probabilistic models can learn latent variables which are inputs to classifiers (e.g sparse coding)

Other models can use latent variables in this same way to perform feature learning, but can capture richer descriptions of the input with model depth and interactions

Often given some model of v and h, experiments show E[h | v] or argmaxh p(h, v) is a good feature mapping from v

**Main idea of dependencies: we can capture dependencies between a set of variables through latent variables h, which may then be used for feature extraction with fewer parameters than the alternative**

**Inference and Approximate Inference**

Sometimes, we need to learn E[h | v] from a set of variables v to solve other tasks; we often train our model using maximum likelihood, and it simplies to the following in terms of h:



We want to compute p(h | v) in order to implement the learning rule

In inference problems, we must predict the probability distribution given values of variables

In deep learning, solving these inference problems is intractable even using structured probabilistic models since the graphs are not restrictive enough for efficient inference

Computing the marginal probability of a general graphical model is a #P hard problem

#P is a generalization of NP which includes not just finding a solution and solving, but also counting the number of solutions

NP hard graphs often arise in real world problems

3-SAT problem for example (look this up)

This motivates the use of approximate inference, or variation inference in deep learning

The true distribution p(h | v) is approximated using q(h | v) where q is as close to p as possible

**Main idea: our model has an ideal marginal probabliity distribution but it is too hard to learn since SPMs are not restrictive enough to also allow for efficient inference, so we must approximate the true distribution with an alternative q**

**Deep Learning Approach to Structured Probabilistic Models**

DL tools are the same as in traditional graphs, but they are combined in different ways when working with these kinds of graphs

The depth can be defined in terms of the graphical model instead of the computational graph (which is probably what a graphical model simplifies to)

The depth of a latent variable hi is j if hi is j steps from any observed variables

The depth of the graphical model is given by the greatest depth of any such hi

This kind of depth is different from the depth induced by the computational graph, where there may only be 1 layer of latent variables but a high computational depth

Deep learning always makes use of distribution representations

Deep learning models have more latent variables than observed variables, with indirect connections between non-connected layers corresponding to nonlinear connections

In a traditional graph structure (non-deep learning), variables are included even if they are only observed in a fraction of all examples

Higher order terms typically represented by a few latent variables capture complex relationships

In deep learning, the practictioner does not intend to capture certain things with latent variables since the model will learn these overarching concepts on its own

When using traditional graph structures, the practitioner usually designs the graphical models with concept in mind but are generally not as reusable or scalable to complex problems

Deep graphical models have large groups of units connected to other large groups of units so their interactions could be described by a single matrix

Traditional graphical models on the other hand have very few connections in order to keep exact inference tractable

When this is too limiting, we use loopy belief propagation, which also works well w/ sparsely connected graphs

In contrast, deep learning models connect each visible unit vi to all hidden units hj effectively making h a distributed representation of vi and several other observed variables

Distributed representations as used in deep learning yield graphs which are not sparse enough for traditional graph models or loopy belief propagation, so we typically do not use loopy belief propaagation with deep learning

Instead, we usually train the deep model to efficiently implement Gibbs' sampling

Since there are many latent variables, numerical code is essential

This provides a motivation for using matrix multiplication with efficient layer groupings - the individual steps of the algorithm are efficient matrix product operations or sparse generalizatons

Rather than simplifying the model until all quantities are known, we increase the power of the model until it is barely possible to use

*Example: Restricted Boltzmann Machine*

The restricted boltzmann machine or RBM is a good example of a deep learning graphical model (nontraditional)

It has a single layer of latent variable which can be used to learn a representation of the input

RBM has characteristics of many deep graphical models: it has layers with connectivity described by matrices, it has dense connectivity, and the model is designed to allow efficient Gibbs' Sampling

Furthermore, the latent variable features are not specified / encouraged by the model designer

An RBM is an energy-based model with an energy function given by  where b, c, and W are unconstrained learnable params

The model is divided into two groups of units v and h, with their interaction descirbed by W

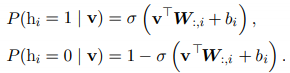
There is no direct interaction between v (visible units) with itself, or h (hidden) with itself

The RBM restrictions yields the properties of conditional independence:





For a binary RBM (where each hidden unit can take on two states):



Together, these properties allow for efficient block Gibbs Sampling where we alternate between sampling all of h simultaneously and all of v simultaneously

The energy function is just a linear function of its parameters, which makes it easy to take the derivative of



Efficient Gibbs and efficient derivatives make training easy

Training the model creates a representation h of the data v

An RBM demonstrates the typical approach to graphical models: representation learning through layers of latent variables combined with efficient interactions between layers parametrized by matrices

**Main idea of RBMs: An RBM is an EBM with a learnable energy function and 1 layer of latent variables, which allows the model to compute the conditional distribution p(v | h) and p(h | v) making Gibbs' Sampling easy**