CSCI-699

Structured Probabilistic Models for Deep Learning

Lecturer: Prof. Xiang Ren *Scribe: Negar Mokhberian*

A structured probabilistic model is a way of describing a probability distribution, using a graph (consisting of nodes and edges) to describe which random variables in the probability distribution interact with each other directly. These models are often also referred to as graphical models.
In this chapter, we provide basic background on some of the most central ideas of graphical models, with an emphasis on the concepts that have proven most useful to the deep learning research community

1 The Challenge of Unstructured Modeling

The goal of deep learning is to become able to understand high imensional data with rich structure. We would like AI algorithms to be able to understand natural images,1 audio waveforms representing speech, and documents containing multiple words and punctuation characters.
Classification algorithms can take an input from such a rich high-dimensional distribution and summarize it with a categorical labelwhat object is in a photo, what word is spoken in a recording, what topic a document is about. The process of classification discards most of the information in the input and produces a single output (or a probability distribution over values of that single output). It is possible to ask probabilistic models to do many other tasks. These tasks are often more expensive than classification. These tasks include the following:

- **Density estimatin:** given an input x, the machine learning system returns an estimate of the true density p(x) under the data generating distribution. This requires only a single output, but it does require a complete understand- ing of the entire input. If even one element of the vector is unusual, the system must assign it a low probability.

- **Denoising:** given a damaged or incorrectly observed input x , the machine learning system returns an estimate of the original or correct x. For example, the machine learning system might be asked to remove dust or scratches from an old photograph. This requires multiple outputs (every element of the estimated clean example x) and an understanding of the entire input (since even one damaged area will still reveal the final estimate as being damaged).

- **Missing value imputation:** given the observations of some elements of x, the model is asked to return estimates of or a probability distribution over some or all of the unobserved elements of x. This requires multiple outputs. Because the model could be asked to restore any of the elements of x, it must understand the entire input.

- **Sampling:** the model generates new samples from the distribution p(x). Applications include speech synthesis, i.e. producing new waveforms that sound like natural human speech. This requires multiple output values and a good model of the entire input. If the samples have even one element drawn from the wrong distribution, then the sampling process is wrong.

If we wish to model a distribution over a random vector $\mathbf{x}$ containing n discrete variables capable of taking on k values each, then the naive approach of representing $p(x)$ by storing a lookup table with one probability value per possible outcome requires k^n parameters. This is not feasible for several reasons:

- **Memory:** The cost of storing the representation: Unless n and k are very small, representing the distribution as a table will require too many values to store.
- **Statisticall Efficiency:** As the number of parameters in a model increases, because the table-based model has an astronomical number of parameters, it will require an astronomically large training set to fit accurately.
- **Runtime (he cost of inference):** For performing inference tasks like computing the marginal distribution $P(x_1)$ or the conditional distribution $P(x_2|x_1)$ using our model of the joint distribution $P(\mathbf{x})$, we need to sum up across the entire table. The runtime of these operations is as high as the intractable memory cost of storing the model.
- **Runtime (the cost of sampling):** Suppose we want to draw a sample from the model. The naive way to do this is to sample some value $u \sim U(0, 1)$, then iterate through the table, adding up the probability values until they exceed u and return the outcome corresponding to that position in the table. This requires reading through the whole table in the worst case, which has an exponential cost.

The problem with the table-based approach is that we are explicitly modeling every possible kind of interaction between every possible subset of variables. The probability distributions we encounter in real tasks are much simpler than this. Usually, most variables influence each other only indirectly.

Consider modeling the finishing times of a team in a relay race. Suppose the team consists of three runners: Alice, Bob and Carol. At the start of the race, Alice begins running around a track and after completing her lap around the track, Bob runs his own lap and then Carol runs the final lap. We can model each of their finishing times as a continuous random variable. Alices finishing time does not depend on anyone elses, since she goes first. Bobs finishing time depends on Alices, because Bob does not have the opportunity to start his lap until Alice has completed hers. Carols finishing time depends on both her teammates. However, Carols finishing time depends only indirectly on Alices finishing time via Bobs. If we already know Bobs finishing time, we will not be able to estimate Carols finishing time better by finding out what Alices finishing time was. This means we can model the relay race using only two interactions: Alices effect on Bob and Bobs effect on Carol. We can omit the third, indirect interaction between Alice and Carol from our model.

Structured probabilistic models provide a formal framework for modeling only direct interactions between random variables. This allows the models to have significantly fewer parameters and therefore be estimated reliably from less data.

2 Using Graphs to Describe Model Structure

Structured probabilistic models use graphs to represent interactions between random variables. Each node represents a random variable. Each edge represents a direct interaction. These direct interactions imply other, indirect interactions, but only the direct interactions need to be explicitly modeled.
In the following sections we describe two categories of graphical models: models based on directed acyclic graphs, and models based on undirected graphs.

2.1 Directed Models

One kind of structured probabilistic model is the directed graphical model, otherwise known as the belief network or Bayesian networ. that is, they point from one vertex to another. Drawing an arrow from a to b means the distribution over b depends on the value of a.

Continuing with the relay race example, suppose we name Alice's, Bob's and Carol's finishing times respectively t_0, t_1 and t_2. As we saw earlier, our estimate of t_1 depends on t_0. Our estimate of t_2 depends directly on t_1 but only indirectly on t_0. We can draw this relationship in a directed graphical model, illustrated in figure 1.
Formally, a directed graphical model defined on variables $\mathbf{x}$ is defined by a directed acyclic graph G whose vertices are the random variables in the model, and a set of local conditional probability distributions $p(x_i|Pa_G(x_i))$ where $Pa_G(x_i)$ gives the set of parents of x_i in G. The probability distribution over $\mathbf{x}$ is given by

$$P(\mathbf{x}) = \prod_i p(x_i|Pa_G(x_i)) \tag{1}$$

In our relay race example, this means that, using the graph drawn in figure 1,

$$p(t_0, t_1, t_2) = p(t_0)p(t_1|t_0)p(t_2|t_1). \tag{2}$$

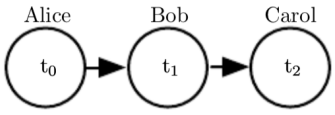

Figure 1: A directed graphical model depicting the relay race example.

We can examine the cost of using a structured probabilistic model over unstructured modeling. Suppose we represented time by discretizing time ranging from minutes 0 to 10 into 6 second chunks. This would give 100 possible values for each variable. If we attempted to represent $p(t_0, t_1, t_2)$ with a table, it would need to store 999,999 values.
If instead, we only make a table for each of the conditional probability distributions, then the

distribution over t_0 requires 99 values, the table defining t_1 given t_0 requires 9900 values, and so does the table defining t_2 given t_1. This comes to a total of 19,899 values. This means that using the directed graphical model reduced our number of parameters by a factor of more than 50!
In general, to model n discrete variables each having k values, the cost of the single table approach scales like $O(k^n)$, as we have observed before. Now suppose we build a directed graphical model over these variables. If m is the maximum number of variables appearing (on either side of the conditioning bar) in a single conditional probability distribution, then the cost of the tables for the directed model scales like $O(k^m)$. As long as we can design a model such that $m << n$, we get very dramatic savings. In other words, so long as each variable has few parents in the graph, the distribution can be represented with very few parameters.

2.2 Undirected Models

Another popular language is that of undirected models, otherwise known as Markov random fields (MRFs) or Markov networks [1].

Not all situations we might want to model have such a clear direction to their interactions. When the interactions seem to have no intrinsic direction, or to operate in both directions, it may be more appropriate to use an undirected model.

As an example of such a situation, suppose we want to model a distribution over three binary variables: whether or not you are sick, whether or not your coworker is sick, and whether or not your roommate is sick represented by h_y, h_c and h_r. Assuming that your coworker and your roommate do not know each other, it is very unlikely that one of them will give the other an infection directly. However, it is reasonably likely that either of them could give you a cold, and that you could pass it on to the other. We can model the indirect transmission of a cold from your coworker to your roommate by modeling the transmission of the cold from your coworker to you and the transmission of the cold from you to your roommate. See figure 2 for the drawing representing this scenario. Unlike directed models, the edge in an undirected model has no arrow, and is not associated with a conditional probability distribution.

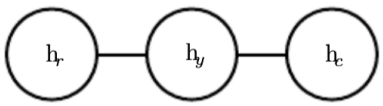

Figure 2: An undirected graph representing how your roommates health, your health, and your work colleague's health affect each other.

Formally, an undirected graphical model is a structured probabilistic model defined on an undirected graph G. For each clique C in the graph, a factor $\phi(C)$ (also called a clique potential) measures the affinity of the variables in that clique for being in each of their possible joint states. The factors are constrained to be non-negative. Together they define an unnormalized probability

distribution

$$\tilde{p}(\mathbf{x}) = \prod_{C \in G} \phi(C) \tag{3}$$

See figure 3 for an example of reading factorization information from an undirected graph. Our example of the cold spreading between you, your roommate, and your colleague contains two cliques. One clique contains h_y and h_c. The factor for this clique can be defined by a table, and might have values resembling these:

	$h_y = 0$	$h_y = 1$
$h_c = 0$	2	1
$h_c = 1$	1	10

To complete the model, we would need to also define a similar factor for the clique containing h_y and h_r.

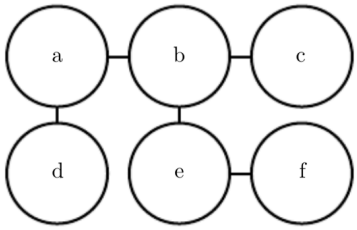

Figure 3: This graph implies that $p(a, b, c, d, e, f)$ can be written as $\frac{1}{Z}\phi_{(a,b)}(a, b)\phi_{(b,c)}(b, c)\phi_{(a,d)}(a, d)\phi_{(b,e)}(b, e)\phi_{(e,f)}(e, f)$ for an appropriate choice of the ϕ functions.

2.3 The Partition Function

While the unnormalized probability distribution is guaranteed to be non-negative everywhere, it is not guaranteed to sum or integrate to 1. To obtain a valid probability distribution, we must use the corresponding normalized probability distribution:

$$p(\mathbf{x}) = \frac{1}{Z}\tilde{p}(\mathbf{x}) \tag{4}$$

where Z is the value that results in the probability distribution summing or integrating to 1:

$$Z = \int \tilde{p}(\mathbf{x}) d\mathbf{x} \tag{5}$$

You can think of Z as a constant when the ϕ functions are held constant. The normalizing constant Z is known as the partition function, a term borrowed from statistical physics.

Since Z is an integral or sum over all possible joint assignments of the state x it is often intractable to compute. Hence, we must resort to approximations.
When designing undirected models, it is possible to specify the factors in such a way that Z does not exist. This happens if some of the variables in the model are continuous and the integral of $\tilde{p}$ over their domain diverges. For example, suppose we want to model a single scalar variable $x \in \mathbb{R}$ with a single clique potential $\phi(x) = x^2$. In this case

$$Z = \int x^2 dx \tag{6}$$

2.4 Energy-Based Models

Many interesting theoretical results about undirected models depend on the assumption that $\forall \mathbf{x}, \tilde{p}(\mathbf{x}) > 0$. A convenient way to enforce this condition is to use an energy-based model (EBM) where

$$\tilde{p}(\mathbf{x}) = exp(-E(\mathbf{x})) \tag{7}$$

and E(x) is known as the energy function. Being completely free to choose the energy function makes learning simpler. If we learned the clique potentials directly, we would need to use constrained optimization to arbitrarily impose some specific minimal probability value. By learning the energy function, we can use unconstrained optimization. The probabilities in an energy-based model can approach arbitrarily close to zero but never reach it.

Any distribution of the form given by equation 16.7 is an example of a Boltz- mann distribution. For this reason, many energy-based models are called Boltzmann machines.

Cliques in an undirected graph correspond to factors of the unnormalized probability function. Because $exp(a)exp(b) = exp(a + b)$, this means that different cliques in the undirected graph correspond to the different terms of the energy function.See figure 4 for an example of how to read the form of the energy function from an undirected graph structure. One can view an energy-based model with multiple terms in its energy function as being a product of experts.

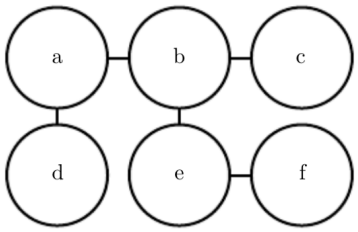

Figure 4: This graph implies that $E(a, b, c, d, e, f)$ can be written as $E_{a,b}(a, b) + E_{b,c}(b, c) + E_{a,d}(a, d) + E_{b,e}(b, e) + E_{e,f}(e, f)$ for an appropriate choice of the per-clique energy functions.

2.5 Separation and D-Separation

In the case of undirected models, conditional independence implied by the graph is called separation. We say that a set of variables A is separated from another set of variables B given a third set of variables S if the graph structure implies that A is independent from B given S. If two variables a and b are connected by a path involving only unobserved variables, then those variables are not separated. If no path exists between them, or all paths contain an observed variable, then they are separated. We refer to paths involving only unobserved variables as "active" and paths including an observed variable as "inactive".

Similar concepts apply to directed models, except that in the context of directed models, these concepts are referred to as d-separation. The "d" stands for "dependence." D-separation for directed graphs is defined the same as separation for undirected graphs.

It is important to remember that separation and d-separation tell us only about those conditional independences that are implied by the graph.

There is no requirement that the graph imply all independences that are present. For example, consider a model of three binary variables: a, b and c. Suppose that when a is 0, b and c are independent, but when a is 1, b is deterministically equal to c. Encoding the behavior when $a = 1$ requires an edge connecting b and c. The graph then fails to indicate that b and c are independent when $a = 0$.

2.6 Converting between Undirected and Directed Graphs

Directed models and undirected models both have their advantages and disad- vantages.We may choose to use either directed modeling or undirected modeling based on which approach can capture the most independences in the probability distribution or which approach uses the fewest edges to describe the distribution. We may sometimes switch between different modeling languages. Sometimes a different language becomes more appropriate if we observe a certain subset of variables, or if we wish to perform a different computational task.

How to show a probability distribution using complete graph

- Undirected: a graph containing a single clique encompassing all of the variables.
- Directed: any directed acyclic graph where we impose some ordering on the random variables.

To convert a directed model with graph D into an undirected model, we need to create a new graph U. For every pair of variables x and y, we add an undirected edge connecting x and y to U if there is a directed edge (in either direction) connecting x and y in D or if x and y are both parents in D of a third variable z. The resulting U is known as a moralized graph.

To convert a undirected model with graph U into a directed model, if we have a loop with length of four or more, we need to add an edge between any connection between any two non-consecutive variables in the sequence defining the loop (this is the definition of chord). To finish the conversion process, we must assign a direction to each edge. When doing so, we must not create any directed

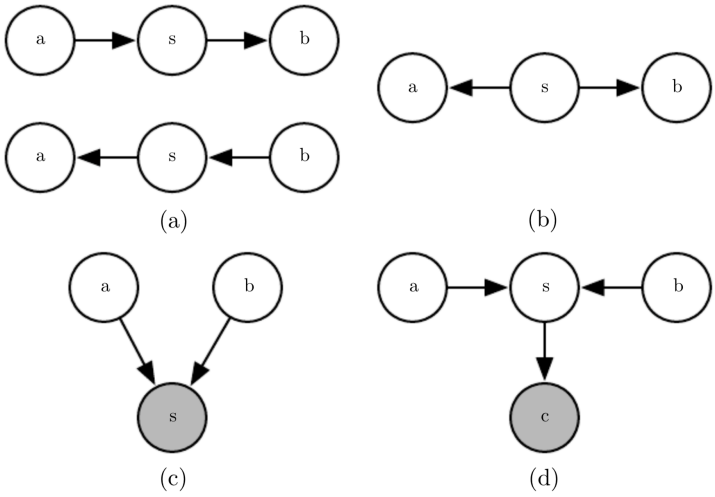

Figure 5: All of the kinds of active paths of length two that can exist between random variables a and b. (a)Any path with arrows proceeding directly from a to b or vice versa. This kind of path becomes blocked if s is observed. We have already seen this kind of path in the relay race example. (b)a and b are connected by a common cause s. For example, suppose s is a variable indicating whether or not there is a hurricane and a and b measure the wind speed at two different nearby weather monitoring outposts. If we observe very high winds at station a, we might expect to also see high winds at b. This kind of path can be blocked by observing s. If we already know there is a hurricane, we expect to see high winds at b, regardless of what is observed at a. A lower than expected wind at a (for a hurricane) would not change our expectation of winds at b (knowing there is a hurricane). However, if s is not observed, then a and b are dependent, i.e., the path is active. (c)a and b are both parents of s. This is called a V-structure or the collider case. The V-structure causes a and b to be related by the explaining away effect. In this case, the path is actually active when s is observed. For example, suppose s is a variable indicating that your colleague is not at work. The variable a represents her being sick, while b represents her being on vacation. If you observe that she is not at work, you can presume she is probably sick or on vacation, but it is not especially likely that both have happened at the same time. If you find out that she is on vacation, this fact is sufficient to explain her absence. You can infer that she is probably not also sick. (d)The explaining away effect happens even if any descendant of s is observed! For example, suppose that c is a variable representing whether you have received a report from your colleague. If you notice that you have not received the report, this increases your estimate of the probability that she is not at work today, which in turn makes it more likely that she is either sick or on vacation. The only way to block a path through a V-structure is to observe none of the descendants of the shared child.

cycles. One way to avoid directed cycles is to impose an ordering over the nodes, and always point each edge from the node that comes earlier in the ordering to the node that comes later in the ordering.

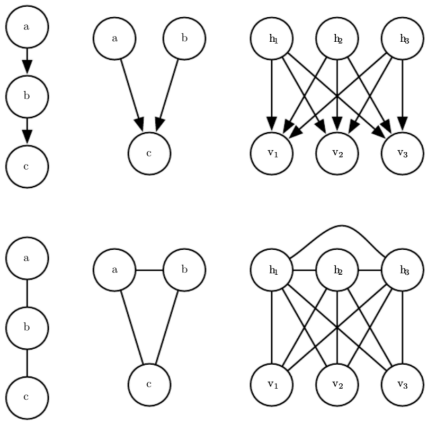

Figure 6: Converting a directed model to undirected

2.7 Factor Graphs

Factor graphs are another way of drawing undirected models that resolve an ambiguity in the graphical representation of standard undirected model syntax. In an undirected model, the scope of every ϕ function must be a subset of some clique in the graph. Ambiguity arises because it is not clear if each clique actually has a corresponding factor whose scope encompasses the entire clique. Factor graphs resolve this ambiguity by explicitly representing the scope of each factor ϕ of the unnormalized probability function drawn as squares. Other nodes correspond to random variables and are drawin with as circles. A variable and a factor are connected in the graph if and only if the variable is one of the arguments to the factor in the unnormalized probability distribution. No factor may be connected to another factor in the graph, nor can a variable be connected to a variable. See figure 7 for an example of how factor graphs can resolve ambiguity in the interpretation of undirected networks.

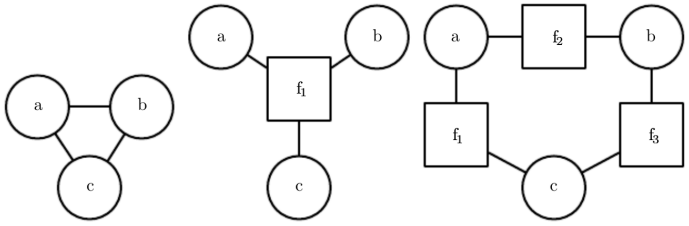

Figure 7: An example of how a factor graph can resolve ambiguity in the interpretation of undirected networks. (Left)An undirected network with a clique involving three variables: a, b and c. (Center)A factor graph corresponding to the same undirected model. This factor graph has one factor over all three variables. (Right)Another valid factor graph for the same undirected model. This factor graph has three factors, each over only two variables.

3 Sampling from Graphical Models

One advantage of directed graphical models is that a simple and efficient proce- dure called ancestral sampling can produce a sample from the joint distribution represented by the model.
The basic idea is to sort the variables x_i in the graph into a topological ordering, so that for all i and j, j is greater than i if x_i is a parent of x_j. The variables can then be sampled in this order.
We first sample $x_1 \sim P(x_1)$, then sample $P(x_2|Pa_G(x2))$, and so on, until finally we sample $P(x_n|Pa_G(x_n))$. The topological sorting operation guarantees that we can read the conditional distributions in equation 16.1 and sample from them in order.
Ancestral sampling is generally very fast and convenient.

drawbacks of ancestral sampling:

- It only applies to directed graphical models.
- It does not support every conditional sampling operation
- It is applicable only to directed models. Sampling from an undirected model without first converting it to a directed model requires resolving cyclical dependencies (The simplest approach is Gibbs sampling).

4 Advantages of Structured Modeling

The primary advantage of using structured probabilistic models is that they allow us to dramatically reduce the cost of representing probability distributions as well as learning and inference. Sampling is also accelerated in the case of directed models, while the situation can be complicated with undirected models.
A less quantifiable benefit of using structured probabilistic models is that they allow us to explicitly separate representation of knowledge from learning of knowledge or inference given existing knowledge. This makes our models easier to develop and debug.

5 Learning about Dependencies

A good generative model needs to accurately capture the distribution over the observed or "visible" variables $\mathbf{v}$. Often the different elements of $\mathbf{v}$ are highly dependent on each other. In the context of deep learning, the approach most commonly used to model these dependencies is to introduce several latent or "hidden" variables, $\mathbf{h}$. The model can then capture dependencies between any pair of variables v_i and v_j indirectly, via direct dependencies between v_i and $\mathbf{h}$, and direct dependencies between $\mathbf{h}$ and v_j.

A good model of $\mathbf{v}$ which did not contain any latent variables would need to have very large numbers of parents per node in a Bayesian network or very large cliques in a Markov network.

An entire field of machine learning called structure learning is devoted to this problem For a good reference on structure learning, see [2]. Most structure learning techniques are a form of greedy search.

Using latent variables instead of adaptive structure avoids the need to perform discrete searches and multiple rounds of training. A fixed structure over visible and hidden variables can use direct interactions between visible and hidden units to impose indirect interactions between visible units. Using simple parameter learning techniques we can learn a model with a fixed structure that imputes the right structure on the marginal $p(\mathbf{v})$.

6 Inference and Approximate Inference

One of the main ways we can use a probabilistic model is to ask inference problems in which we must predict the value of some variables given other variables, or predict the probability distribution over some variables given the value of other variables. For example given a set of medical tests, we can ask what disease a patient might have. In a latent variable model, we might want to extract features $E[h|\mathbf{v}]$ describing the observed variables.
We often train our models using the principle of maximum likelihood. Because

$$\log p(\mathbf{v}) = E_{h\sim p(h|v)}[\log(p(h,v) - log(p(h|v))] \tag{8}$$

We often want to compute $p(\mathbf{h}|v)$ in order to implement a learning rule. All of these are examples of inference problems.
Unfortunately, for most interesting deep models, these inference problems are intractable, even when we use a structured graphical model to simplify them. This motivates the use of approximate inference. In the context of deep learning, this usually refers to variational inference, in which we approximate the true distribution $p(\mathbf{h}|v)$ by seeking an approximate distribution $q(\mathbf{h}|v)$ that is as close to the true one as possible.

7 The Deep Learning Approach to Structured Probabilistic Models

We can think of a latent variable h_i as being at depth j if the shortest path from h_i to an observed variable is j steps. We usually describe the depth of the graphical model as being the greatest depth of any such h_i.

The differences between deep learning and traditional graphical models are as follows:

- Deep learning models typically have more latent variables than observed variables. Complicated nonlinear interactions between variables are accomplished via indirect connections that flow through multiple latent variables. By contrast, traditional graphical models usually contain mostly variables that are at least occasionally observed.

- Deep learning practitioner typically does not intend for the latent variables to take on any specific semantics ahead of time, that's why the laten variables are usually not very easy for a human to interpret, however When latent variables are used in the context of traditional graphical models, they are often designed with some specific semantics in mind.

- Deep graphical models typically have large groups of units that are all connected to other groups of units and the interactions between these two groups is described using a single matrix. On the other hand, traditional graphical models have very few connections and the choice of connections for each variable may be individually designed which depends on choice of inference algorithm. Traditional approaches to graphical models typically aim to maintain the tractability of exact inference. When this constraint is too limiting, a popular approximate inference algorithm is an algorithm called loopy belief propagation.

- One of the most striking differences between the larger graphical models community and the deep graphical models community is that loopy belief propagation is almost never used for deep learning, because in deep learning we mostly use distributed representation which has the disadvantage of usually yielding graphs that are not sparse enough for the traditional techniques.

- The deep learning approach is often to figure out what the minimum amount of information we absolutely need is, and then to figure out how to get a reasonable approximation of that information as quickly as possible, rather than simplifying the model until all quantities we might want can be computed exactly.

The restricted Boltzmann machine The restricted Boltzmann machine (RBM) [3] or harmonium is the quintessential example of how graphical models are used for deep learning. The RBM is not itself a deep model. Instead, it has a single layer of latent variables that may be used to learn a representation for the input.

Its units are organized into large groups called layers and the connectivity is described as a single matrix. The model is designed to allow efficient Gibbs sampling, and learn latent variables whose semantics were not specified by the designer.

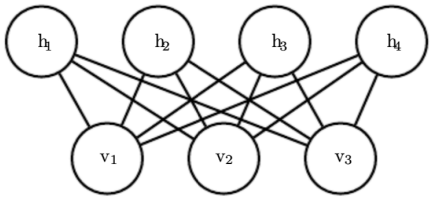

Figure 8: An RBM drawn as a Markov network

As this figure makes clear, an important aspect of this model is that there are no direct interactions between any two visible units or between any two hidden units (hence the restricted, a general Boltzmann machine may have arbitrary connections)

Overall, the RBM demonstrates the typical deep learning approach to graph- ical models: representation learning accomplished via layers of latent variables, combined with efficient interactions between layers parametrized by matrices.

References

[1] R. Kindermann. Markov random fields and their applications. *American mathematical society*, 1980.

[2] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[3] T. Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071. ACM, 2008.