CS7015 (Deep Learning) : Lecture 19

Using joint distributions for classification and sampling, Latent Variables, Restricted Boltzmann Machines, Unsupervised Learning, Motivation for Sampling

Mitesh M. Khapra

Department of Computer Science and Engineering
Indian Institute of Technology Madras

Acknowledgments

- Probabilistic Graphical models: Principles and Techniques, Daphne Koller and Nir Friedman
- An Introduction to Restricted Boltzmann Machines, Asja Fischer and Christian Igel

Module 19.1: Using joint distributions for classification and sampling

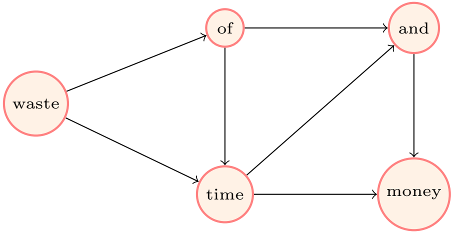
Now that we have some understanding of joint probability distributions and efficient ways of representing them, let us see some more practical examples where we can use these joint distributions

- **M1:** An unexpected and necessary masterpiece
- **M2:** Delightfully merged information and comedy
- **M3:** Director's first true masterpiece
- **M4:** Sci-fi perfection,truly mesmerizing film.
- **M5:** Waste of time and money
- **M6:** Best Lame Historical Movie Ever

- Consider a movie critic who writes reviews for movies
- For simplicity let us assume that he always writes reviews containing a maximum of 5 words
- Further, let us assume that there are a total of 50 words in his vocabulary
- Each of the 5 words in his review can be treated as a random variable which takes one of the 50 values
- Given many such reviews written by the reviewer we could learn the joint probability distribution

$$P(X_1, X_2, \ldots, X_5)$$

- **M1:** An unexpected and necessary masterpiece
- **M2:** Delightfully merged information and comedy
- **M3:** Director's first true masterpiece
- **M4:** Sci-fi perfection,truly mesmerizing film.
- **M5:** Waste of time and money
- **M6:** Best Lame Historical Movie Ever



- In fact, we can even think of a very simple factorization for this model

$$P(X_1, X_2, \ldots, X_5) = \prod P(X_i|X_{i-1}, X_{i-2})$$

- In other words, we are assuming that the i-th word only depends on the previous 2 words and not anything before that
- Let us consider one such factor $P(X_i = time|X_{i-2} = waste, X_{i-1} = of)$
- We can estimate this as

$$\frac{count(\text{waste of time})}{count(\text{waste of})}$$

- And the two counts mentioned above can be computed by going over all the reviews
- We could similarly compute the probabilities of all such factors

M7: More realistic than real life

w	$P(X_i = w \mid X_{i-2} = more, X_{i-1} = realistic)$	$P(X_i = w \mid X_{i-2} = realistic, X_{i-1} = than)$	$P(X_i = w \mid X_{i-2} = than, X_{i-1} = real)$	...
than	0.61	0.01	0.20	...
as	0.12	0.10	0.16	...
for	0.14	0.09	0.05	...
real	0.01	0.50	0.01	...
the	0.02	0.12	0.12	...
life	0.05	0.11	0.33	...

$$P(M7) = P(X_1 = more).P(X_2 = realistic|X_1 = more).$$
$$P(X_3 = than|X_1 = more, X_2 = realistic).$$
$$P(X_4 = real|X_2 = realistic, X_3 = than).$$
$$P(X_5 = life|X_3 = than, X_4 = real)$$
$$= 0.2 \times 0.25 \times 0.61 \times 0.50 \times 0.33 = 0.005$$

- Okay, so now what can we do with this joint distribution?
- Given a review, *classify* if this was written by the reviewer
- *Generate* new reviews which would look like reviews written by this reviewer
- How would you do this? By sampling from this distribution! What does that mean? Let us see!

w	$P(X_1 = w)$	$P(X_2 = w \| , X_1 = the)$	$P(X_i = w \| , X_{i-2} = the, X_{i-1} = movie)$	$\dots$
the	0.62	0.01	0.01	$\dots$
movie	0.10	0.40	0.01	$\dots$
amazing	0.01	0.22	0.01	$\dots$
useless	0.01	0.20	0.03	$\dots$
was	0.01	0.00	0.60	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\dots$

The movie was really amazing

- How does the reviewer start his reviews (what is the first word that he chooses)?
- We could take the word which has the highest probability and put it as the first word in our review
- Having selected this what is the most likely second word that the reviewer uses?
- Having selected the first two words what is the most likely third word that the reviewer uses?
- and so on...

w	$P(X_1 = w)$	$P(X_2 = w \mid, X_1 = the)$	$P(X_i = w \mid, X_{i-2} = the, X_{i-1} = movie)$	$\dots$
the	0.62	0.01	0.01	$\dots$
movie	0.10	0.40	0.01	$\dots$
amazing	0.01	0.22	0.01	$\dots$
useless	0.01	0.20	0.03	$\dots$
was	0.01	0.00	0.60	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\dots$

The movie was really amazing

- But there is a catch here!
- Selecting the most likely word at each time step will only give us the same review again and again!
- But we would like to generate different reviews
- So instead of taking the max value we can sample from this distribution
- How? Let us see!

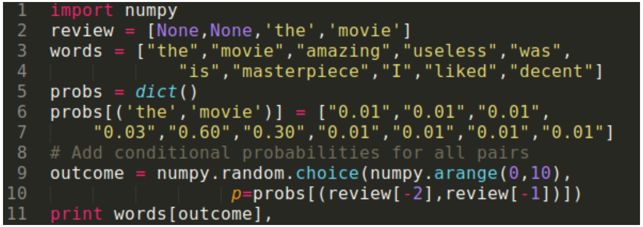
w	$P(X_1 = w)$	$P(X_2 = w \mid, X_1 = the)$	$P(X_i = w \mid, X_{i-2} = the, X_{i-1} = movie)$	...
the	0.62	0.01	0.01	...
movie	0.10	0.40	0.01	...
amazing	0.01	0.22	0.01	...
useless	0.01	0.20	0.03	...
was	0.01	0.00	0.60	...
is	0.01	0.00	0.30	...
masterpiece	0.01	0.11	0.01	...
I	0.21	0.00	0.01	...
liked	0.01	0.01	0.01	...
decent	0.01	0.02	0.01	...

- Suppose there are 10 words in the vocabulary
- We have computed the probability distribution $P(X_1 = word)$
- $P(X_1 = the)$ is the fraction of reviews having *the* as the first word
- Similarly, we have computed $P(X_2 = word_2 | X_1 = word_1)$ and $P(X_3 = word_3 | X_1 = word_1, X_2 = word_2)$

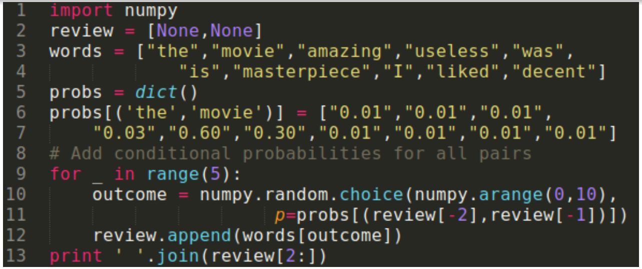
The movie …

Index	**Word**	$P(X_i = w \mid, X_{i-2} = the, X_{i-1} = movie)$	…
0	the	0.01	…
1	movie	0.01	…
2	amazing	0.01	…
3	useless	0.03	…
4	was	0.60	…
5	is	0.30	…
6	masterpiece	0.01	…
7	I	0.01	…
8	liked	0.01	…
9	decent	0.01	…

- Now consider that we want to generate the 3rd word in the review given the first 2 words of the review
- We can think of the 10 words as forming a 10 sided dice where each side corresponds to a word
- The probability of each side showing up is not uniform but as per the values given in the table
- We can select the next word by rolling this dice and picking up the word which shows up
- You can write a python program to roll such a biased dice



```
import numpy
review = [None,None,'the','movie']
words = ["the","movie","amazing","useless","was",
             "is","masterpiece","I","liked","decent"]
probs = dict()
probs[('the','movie')] = ["0.01","0.01","0.01",
    "0.03","0.60","0.30","0.01","0.01","0.01","0.01"]
# Add conditional probabilities for all pairs
outcome = numpy.random.choice(numpy.arange(0,10),
                  p=probs[(review[-2],review[-1])])
print words[outcome],
```



```python
import numpy
review = [None,None]
words = ["the","movie","amazing","useless","was",
            "is","masterpiece","I","liked","decent"]
probs = dict()
probs[('the','movie')] = ["0.01","0.01","0.01",
    "0.03","0.60","0.30","0.01","0.01","0.01","0.01"]
# Add conditional probabilities for all pairs
for _ in range(5):
    outcome = numpy.random.choice(numpy.arange(0,10),
                     p=probs[(review[-2],review[-1])])
    review.append(words[outcome])
print ' '.join(review[2:])
```

Generated Reviews

- the movie is liked decent
- I liked the amazing movie
- the movie is masterpiece
- the movie I liked useless

- Now, at each timestep we do not pick the most likely word but all words are possible depending on their probability (just as rolling a biased dice or tossing a biased coin)
- Every run will now give us a different review!

Returning back to our story....

M7: More realistic than real life

w	$P(X_i = w \mid X_{i-2} = more, X_{i-1} = realistic)$	$P(X_i = w \mid X_{i-2} = realistic, X_{i-1} = than)$	$P(X_i = w \mid X_{i-2} = than, X_{i-1} = real)$	...
than	0.61	0.01	0.20	...
as	0.12	0.10	0.16	...
for	0.14	0.09	0.05	...
real	0.01	0.50	0.01	...
the	0.02	0.12	0.12	...
life	0.05	0.11	0.33	...

$$P(M7) = P(X_1 = more).P(X_2 = realistic|X_1 = more). P(X_3 = than|X_1 = more, X_2 = realistic). P(X_4 = real|X_2 = realistic, X_3 = than). P(X_5 = life|X_3 = than, X_4 = real)$$

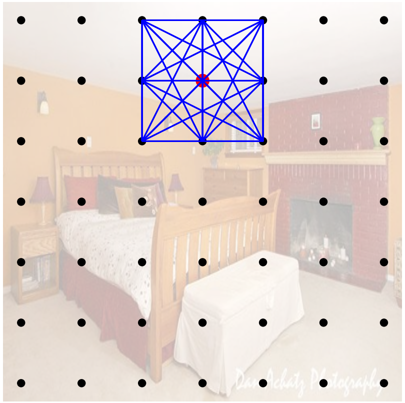
$$= 0.2 \times 0.25 \times 0.61 \times 0.50 \times 0.33 = 0.005$$

- Okay, so now what can we do with this joint distribution?
- Given a review, *classify* if this was written by the reviewer
- *Generate* new reviews which would look like reviews written by this reviewer
- *Correct noisy reviews* or help in completing incomplete reviews

$$\underset{X_5}{argmax}\ P(X_1 = the, X_2 = movie, X_3 = was, X_4 = amazingly, X_5 = ?)$$

Let us take an example from another domain

- Consider images which contain $m \times n$ pixels (say 32×32)
- Each pixel here is a random variable which can take values from 0 to 255 (colors)
- We thus have a total of $32 \times 32 = 1024$ random variables $(X_1, X_2, ..., X_{1024})$
- Together these pixels define the image and different combinations of pixel values lead to different images
- Given many such images we want to learn the joint distribution $P(X_1, X_2, ..., X_{1024})$



- We can assume each pixel is dependent only on its neighbors
- In this case we could factorize the distribution over a Markov network

$$\prod \phi(D_i)$$

where D_i is a set of variables which form a maximal clique (basically, groups of neighboring pixels)

- Again, what can we do with this joint distribution?
- Given a new image, *classify* if is indeed a bedroom
- *Generate new images* which would look like bedrooms (say, if you are an interior designer)
- *Correct noisy images* or help in completing incomplete images

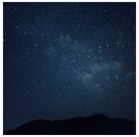
- Such models which try to estimate the probability $P(X)$ from a large number of samples are called generative models

Module 19.2: The concept of a latent variable

- We now introduce the concept of a latent variable
- Recall that earlier we mentioned that the neighboring pixels in an image are dependent on each other
- Why is it so? (intuitively, because we expect them to have the same color, texture, etc.?)
- Let us probe this intuition a bit more and try to formalize it

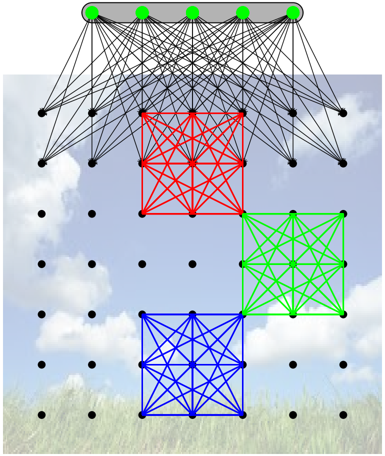
- Suppose we asked a friend to send us a good wallpaper and he/she thinks a bit about it and sends us this image
- Why are all the pixels in the top portion of the image blue? (because our friend decided to show us an image of the sky as opposed to mountains or green fields)
- But then why blue why not black? (because our friend decided to show us an image which depicts daytime as opposed to night time)
- Okay, But why is it not cloudy (gray)?(because our friend decided to show us an image which depicts a sunny day)
- These decisions made by our friend (sky, sunny, daytime, etc) are not explicitly known to us (they are hidden from us)
- We only observe the images but what we observe depends on these latent (hidden) decisions

Latent Variable = daytime


Latent Variable = night

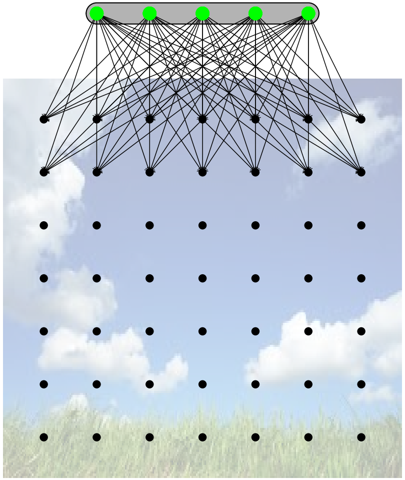
Latent Variable = cloudy

- So what exactly are we trying to say here?
- We are saying that there are certain underlying hidden (latent) characteristics which are determining the pixels and their interactions
- We could think of these as additional (latent) random variables in our distribution
- These are latent because we do not observe them unlike the pixels which are observable random variables
- The pixels depend on the choice of these latent variables



- More formally we now have visible (observed) variables or pixels ($V = \{V_1, V_2, V_3, \ldots, V_{1024}\}$) and hidden variables ($H = \{H_1, H_2, ..., H_n\}$)
- Can you now think of a Markov network to represent the joint distribution $P(V, H)$?
- Our original Markov Network suggested that the pixels were dependent on neighboring pixels (forming a clique)
- But now we could have a better Markov Network involving these latent variables
- This Markov Network suggests that the pixels (observed variables) are dependent on the latent variables (which is exactly the intuition that we were trying to build in the previous slides)
- The interactions between the pixels are captured through the latent variables

- Before we move on to more formal definitions and equations, let us probe the idea of using latent variables a bit more
- We will talk about two concepts: *abstraction* and *generation*



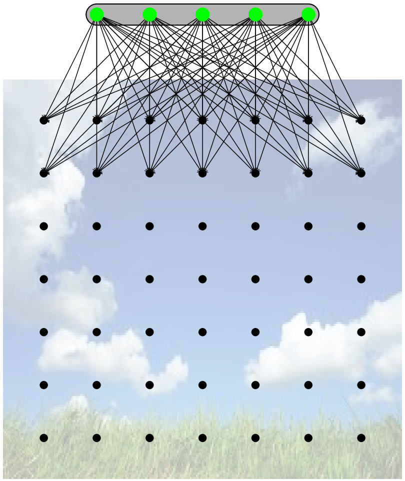
- First let us talk about *abstraction*
- Suppose, we are able to learn the joint distribution $P(V, H)$
- Using this distribution we can find

$$P(H|V) = \frac{P(V,H)}{\sum_H P(V,H)}$$

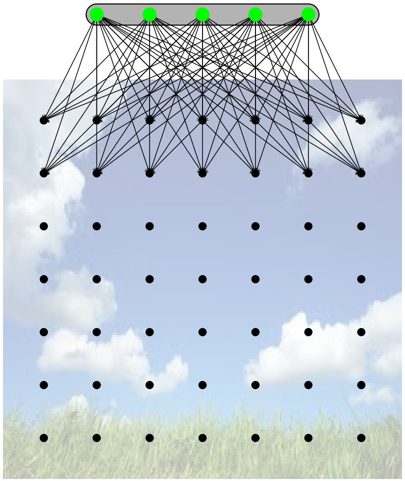
- In other words, given an image, we can find the most likely latent configuration $(H = h)$ that generated this image (of course, keeping the computational cost aside for now)
- What does this h capture? It captures a latent representation or abstraction of the image!

- In other words, it captures the most important properties of the image
- For example, if you were to describe the adjacent image you wouldn't say "I am looking at an image where pixel 1 is blue, pixel 2 is blue, ..., pixel 1024 is beige"
- Instead you would just say "I am looking at an image of a sunny beach with an ocean in the background and beige sand"
- This is exactly the abstraction captured by the vector h

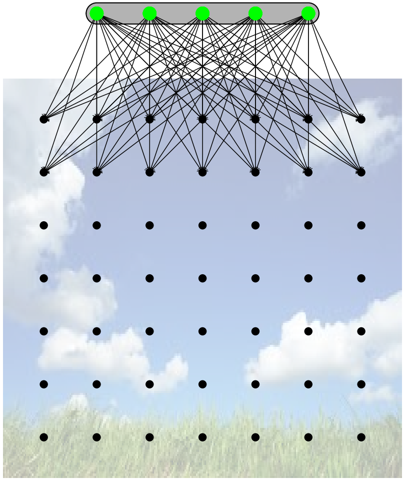
- Under this abstraction all these images would look very similar (i.e., they would have very similar latent configurations h)
- Even though in the original feature space (pixels) there is a significant difference between these images, in the latent space they would be very close to each other
- This is very similar to the idea behind PCA and autoencoders



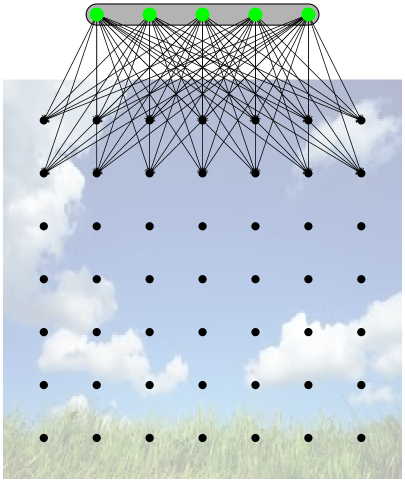
- Of course, we still need to figure out a way of computing $P(H|V)$
- In the case of PCA, learning such latent representations boiled down to learning the eigen vectors of $X^\top X$ (using linear algebra)
- In the case of Autoencoders, this boiled down to learning the parameters of the feedforward network (W_{end}, W_{dec}) (using gradient descent)
- We still haven't seen how to learn the parameters of $P(H, V)$ (we are far from it but we will get there soon!)



- Ok, I am just going to drag this a bit more! (bear with me)
- Remember that in practice we have no clue what these hidden variables are!
- Even in PCA, once we are given the new dimensions we have no clue what these dimensions actually mean
- We cannot interpret them (for example, we cannot say dimension 1 corresponds to weight, dimension 2 corresponds to height and so on!)
- Even here, we just assume there are some latent variables which capture the essence of the data but we do not really know what these are (because no one ever tells us what these are)
- Only for illustration purpose we assumed that h_1 corresponds to sunny/cloudy, h_2 corresponds to beach and so on



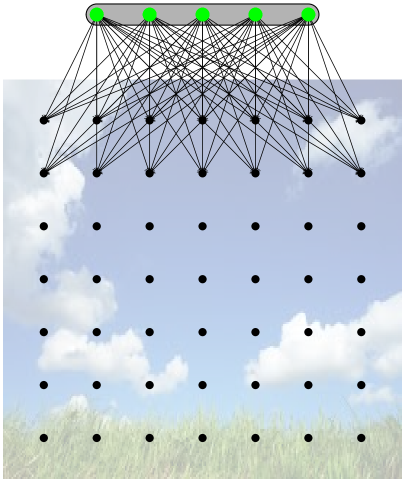
- Just to reiterate, remember that while sending us the wallpaper images our friend never told us what latent variables he/she considered
- Maybe our friend had the following latent variables in mind: $h_1 = cheerful$, $h_2 = romantic$, and so on
- In fact, it doesn't really matter what the interpretation of these latent variable is
- All we care about is that they should help us learn a good abstraction of the data
- How? (we will get there eventually)



- We will now talk about another interesting concept related to latent variables: *generation*
- Once again, assume that we are able to learn the joint distribution $P(V, H)$
- Using this distribution we can find

$$P(V|H) = \frac{P(V, H)}{\sum_V P(V, H)}$$

- Why is this interesting?



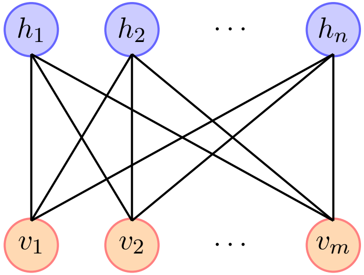
- Well, I can now say "Create an image which is cloudy, has a beach and depicts daytime"
- Or given $h = [....]$ find the corresponding V which maximizes $P(V|H)$
- In other words, I can now generate images given certain latent variables
- The hope is that I should be able to ask the model to generate very creative images given some latent configuration (we will come back to this later)

The story ahead...

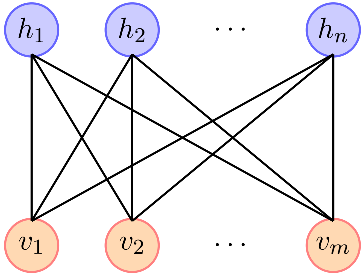
- We have tried to understand the intuition behind latent variables and how they could potenatially allow us to do abstraction and generation
- We will now concretize these intuitions by developings equations (models) and learning algoritms
- And of course, we will tie all this back to neural networks!

- For the remainder of this discussion we will assume that all our variables take only boolean values
- Thus, the vector V will be a boolean vector $\in \{0,1\}^m$ (there are a total of 2^m values that V can take)
- And the vector H will be a boolean vector $\in \{0,1\}^n$ (there are a total of 2^n values that H can take)

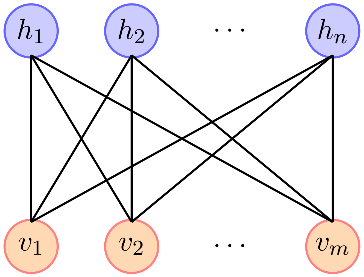
Module 19.3: Restricted Boltzmann Machines



- We return back to our Markov Network containing hidden variables and visible variables
- We will get rid of the image and just keep the hidden and latent variables
- We have edges between each pair of (hidden, visible) variables.
- We do not have edges between (hidden, hidden) and (visible, visible) variables



- Earlier, we saw that given such a Markov network the joint probability distribution can be written as a product of factors
- Can you tell how many factors are there in this case?
- Recall that factors correspond to maximal cliques
- What are the maximal cliques in this case? every pair of visible and hidden node forms a clique
- How many such cliques do we have? $(m \times n)$



- So we can write the joint pdf as a product of the following factors

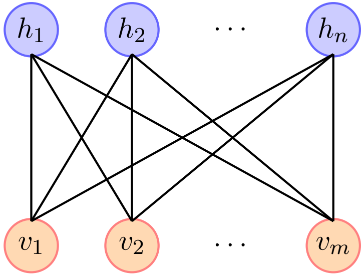
$$P(V,H) = \frac{1}{Z}\prod_i \prod_j \phi_{ij}(v_i, h_j)$$

- In fact, we can also add additional factors corresponding to the nodes and write

$$P(V,H) = \frac{1}{Z}\prod_i \prod_j \phi_{ij}(v_i, h_j)\prod_i \psi_i(v_i)\prod_j \xi_j(h_j)$$

- It is legal to do this (i.e., add factors for $\psi_i(v_i)\xi_j(h_j)$) as long as we ensure that Z is adjusted in a way that the resulting quantity is a probability distribution
- Z is the partition function and is given by

$$\sum_V \sum_H \prod_i \prod_j \phi_{ij}(v_i, h_j)\prod_i \psi_i(v_i)\prod_j \xi_j(h_j)$$

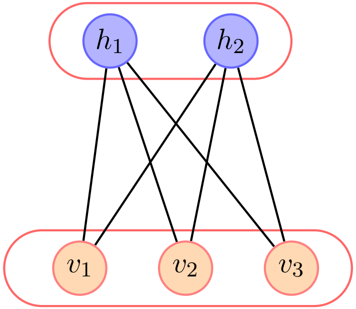


$\phi_{11}(v_1, h_1)$		
0	0	30
0	1	5
1	0	1
1	1	10

$\psi_1(v_1)$	
0	10
1	2

- Let us understand each of these factors in more detail
- For example, $\phi_{11}(v_1, h_1)$ is a factor which takes the values of $v_1 \in \{0, 1\}$ and $h_1 \in \{0, 1\}$ and returns a value indicating the affinity between these two variables
- The adjoining table shows one such possible instantiation of the ϕ_{11} function
- Similarly, $\psi_1(v_1)$ takes the value of $v_1 \in \{0, 1\}$ and gives us a number which roughly indicates the possibility of v_1 taking on the value 1 or 0
- The adjoining table shows one such possible instantiation of the ψ_{11} function
- A similar interpretation can be made for $\xi_1(h_1)$

Just to be sure that we understand this correctly let us take a small example where $|V| = 3$ (i.e., $V \in \{0, 1\}^3$) and $|H| = 2$ (i.e., $H \in \{0, 1\}^2$)



$\phi_{11}(v_1,h_1)$			$\phi_{12}(v_1,h_2)$			$\phi_{21}(v_2,h_1)$			$\phi_{22}(v_2,h_2)$			$\phi_{31}(v_3,h_1)$			$\phi_{32}(v_3,h_2)$		
0	0	20	0	0	6	0	0	3	0	0	2	0	0	6	0	0	3
0	1	3	0	1	20	0	1	3	0	1	1	0	1	3	0	1	1
1	0	5	1	0	10	1	0	2	1	0	10	1	0	5	1	0	10
1	1	10	1	1	2	1	1	10	1	1	10	1	1	10	1	1	10

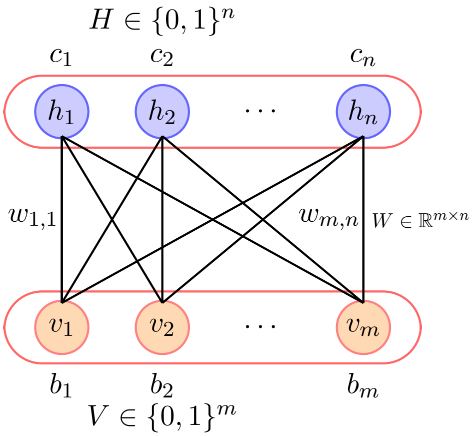
$\psi_1(v_1)$		$\psi_2(v_2)$		$\psi_3(v_3)$		$\xi_1(h_1)$		$\xi_2(h_2)$	
0	30	0	100	0	1	0	100	0	10
1	1	1	1	1	100	1	1	1	10

- Suppose we are now interested in $P(V =< 0,0,0 >, H =< 1,1 >)$
- We can compute this using the following function

$$\begin{aligned} &P(V =< 0,0,0 >, H =< 1,1 >) \\ &= \frac{1}{Z}\phi_{11}(0,1)\phi_{12}(0,1)\phi_{21}(0,1) \\ &\quad \phi_{22}(0,1)\phi_{31}(0,1)\phi_{32}(0,1) \\ &\quad \psi_1(0)\psi_2(0)\psi_3(0)\xi_1(1)\xi_2(1) \end{aligned}$$

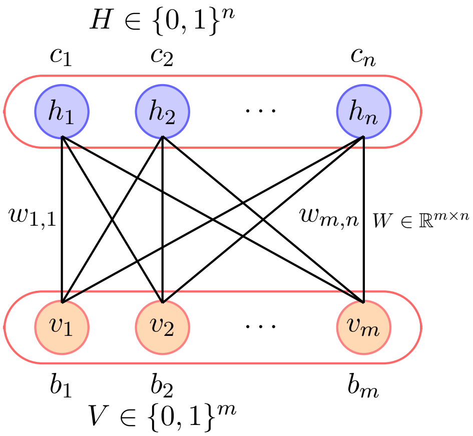
- and the partition function will be given by

$$\sum_{v_1=0}^{1}\sum_{v_2=0}^{1}\sum_{v_3=0}^{1}\sum_{h_1=0}^{1}\sum_{h_2=1}^{1} P(V =< v_1, v_2, v_3 >, H =< h_1, h_2 >)$$



- How do we learn these clique potentials: $\phi_{ij}(v_i, h_j), \psi_i(v_i), \xi_j(h_j)$?
- Whenever we want to learn something what do we introduce? (parameters)
- So we will introduce a parametric form for these clique potentials and then learn these parameters
- The specific parametric form chosen by RBMs is

$$\phi_{ij}(v_i, h_j) = e^{w_{ij} v_i h_j}$$
$$\psi_i(v_i) = e^{b_i v_i}$$
$$\xi_j(h_j) = e^{c_j h_j}$$



- With this parametric form, let us see what the joint distribution looks like

$$P(V,H) = \frac{1}{Z}\prod_i\prod_j \phi_{ij}(v_i,h_j)\prod_i \psi_i(v_i)\prod_j \xi_j(h_j)$$

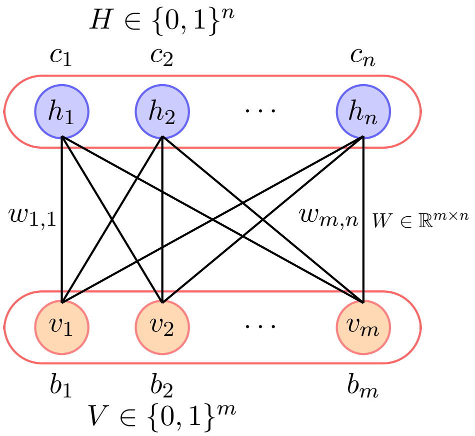
$$= \frac{1}{Z}\prod_i\prod_j e^{w_{ij}v_ih_j}\prod_i e^{b_iv_i}\prod_j e^{c_jh_j}$$

$$= \frac{1}{Z}e^{\sum_i\sum_j w_{ij}v_ih_j}e^{\sum_i b_iv_i}e^{\sum_j c_jh_j}$$

$$= \frac{1}{Z}e^{\sum_i\sum_j w_{ij}v_ih_j+\sum_i b_iv_i+\sum_j c_jh_j}$$

$$= \frac{1}{Z}e^{-E(V,H)} \text{ where,}$$

$$E(V,H) = -\sum_i\sum_j w_{ij}v_ih_j - \sum_i b_iv_i - \sum_j c_jh_j$$



$$E(V,H) = -\sum_i \sum_j w_{ij} v_i h_j - \sum_i b_i v_i - \sum_j c_j h_j$$

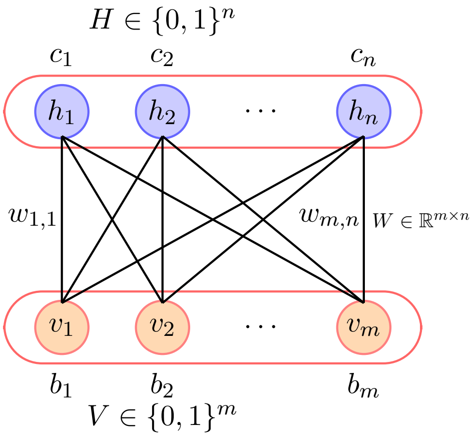
- Because of the above form, we refer to these networks as (restricted) Boltzmann machines
- The term comes from statistical mechanics where the distribution of particles in a system over various possible states is given by

$$F(state) \propto e^{-\frac{E}{kt}}$$

which is called the Boltzmann distribution or the Gibbs distribution

Module 19.4: RBMs as Stochastic Neural Networks

- But what is the connection between this and deep neural networks?
- We will get to it over the next few slides!



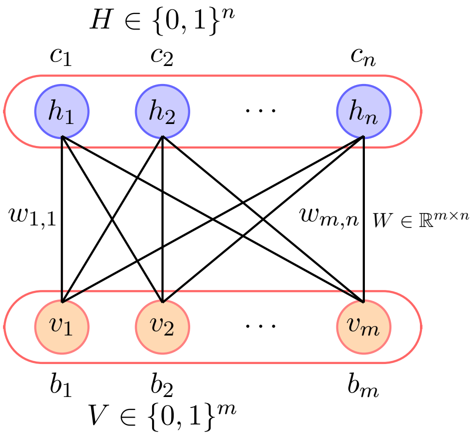
- We will start by deriving a formula for $P(V|H)$ and $P(H|V)$
- In particular, let us take the l-th visible unit and derive a formula for $P(v_l = 1|H)$
- We will first define V_{-l} as the state of all the visible units except the l-th unit
- We now define the following quantities

$$\alpha_l(H) = -\sum_{i=1}^{n} w_{il} h_i - b_l$$

$$\beta(V_{-l}, H) = -\sum_{i=1}^{n} \sum_{j=1, j \neq l}^{m} w_{ij} h_i v_j - \sum_{j=1, j \neq l}^{m} b_i v_i - \sum_{i=1}^{n} c_i h_i$$

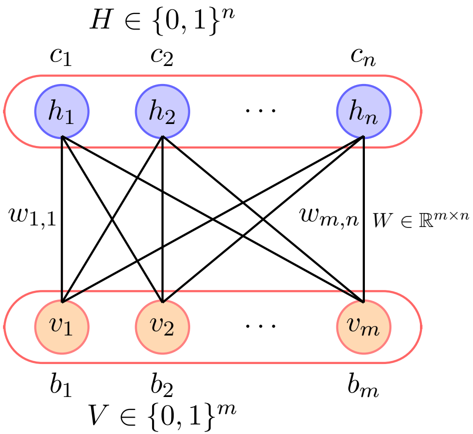
- Notice that

$$E(V, H) = v_l \alpha(H) + \beta(V_{-l}, H)$$



- We can now write $P(v_l = 1|H)$ as

$$\begin{aligned}
p(v_l = 1|H) &= P(v_l = 1|V_{-l}, H) \\
&= \frac{p(v_l = 1, V_{-l}, H)}{p(V_{-l}, H)} \\
&= \frac{e^{-E(v_l=1,V_{-l},H)}}{e^{-E(v_l=1,V_{-l},H)} + e^{-E(v_l=0,V_{-l},H)}} \\
&= \frac{e^{-\beta(V_{-l},H)-1\cdot\alpha_l(H)}}{e^{-\beta(V_{-l},H)-1\cdot\alpha_l(H)} + e^{-\beta(V_{-l},H)-0\cdot\alpha_l(H)}} \\
&= \frac{e^{-\beta(V_{-l},H)} \cdot e^{-\alpha_l(H)}}{e^{-\beta(V_{-l},H)} \cdot e^{-\alpha_l(H)} + e^{-\beta(V_{-l},H)}} \\
&= \frac{e^{-\alpha_l(H)}}{e^{-\alpha_l(H)} + 1} = \frac{1}{1 + e^{\alpha_l(H)}} \\
&= \sigma(-\alpha_l(H)) = \sigma(\sum_{i=1}^{n} w_{il}h_i + b_l)
\end{aligned}$$



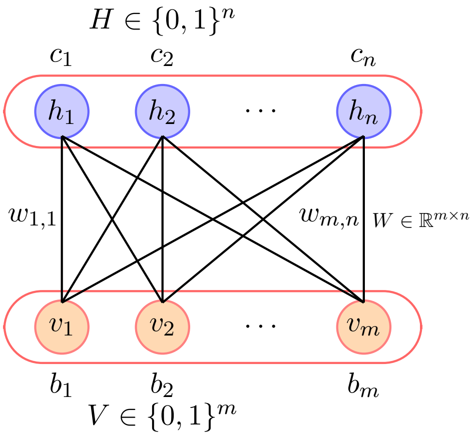
- Okay, so we arrived at

$$p(v_l = 1|H) = \sigma(\sum_{i=1}^{n} w_{il}h_i + b_l)$$

- Similarly, we can show that

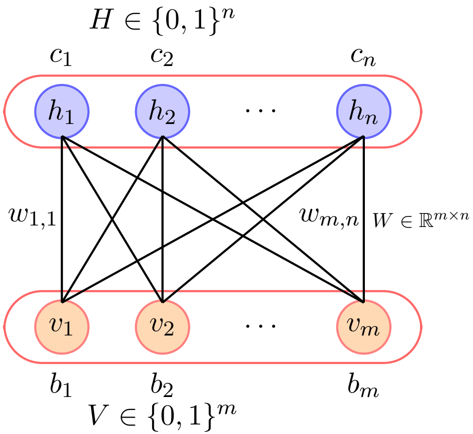
$$p(h_l = 1|V) = \sigma(\sum_{i=1}^{m} w_{il}v_i + c_l)$$

- The RBM can thus be interpreted as a stochastic neural network, where the nodes and edges correspond to neurons and synaptic connections, respectively.
- The conditional probability of a single (hidden or visible) variable being 1 can be interpreted as the firing rate of a (stochastic) neuron with sigmoid activation function

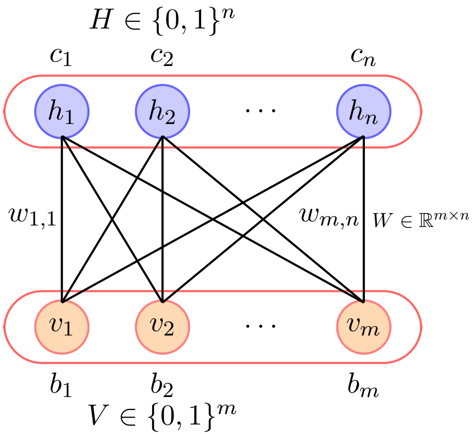


- Given this neural network view of RBMs, can you say something about what h is trying to learn?
- It is learning an abstract representation of V
- This looks similar to autoencoders but how do we train such an RBM? What is the objective function?
- We will see this in the next lecture!

Module 19.5: Unsupervised Learning with RBMs

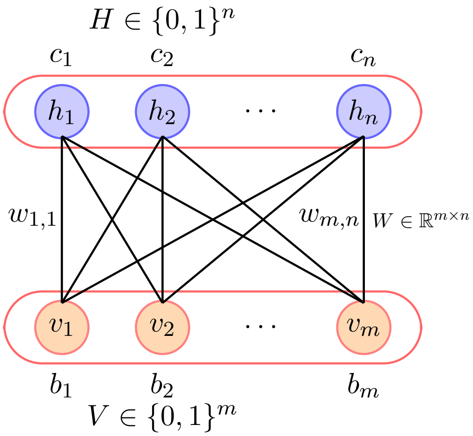


- So far, we have mainly dealt with supervised learning where we are given $\{x_i, y_i\}_{i=1}^{n}$ for training
- In other words, for every training example we are given a label (or class) associated with it
- Our job was then to learn a model which predicts $\hat{y}$ such that the difference between y and $\hat{y}$ is minimized



- But in the case of RBMs, our training data only contains x (for example, images)
- There is no explicit label (y) associated with the input
- Of course, in addition to x we have the latent variable h but we don't know what these h's are
- We are interested in learning $P(x, h)$ which we have parameterized as

$$P(V, H) = \frac{1}{Z} e^{-(-\sum_i \sum_j w_{ij} v_i h_j - \sum_i b_i v_i - \sum_j c_j h_j)}$$



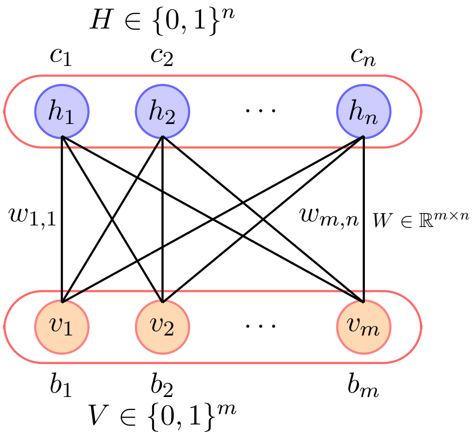
- What is the objective function that we should use?
- First note that if we have learnt $P(x, h)$ we can compute $P(x)$
- What would we want $P(X = x)$ to be for any x belonging to our training data?
- We would want it to be high
- So now can you think of an objective function

$$maximize \prod_{i=1}^{N} P(X = x_i)$$

- Or, log-likelihood

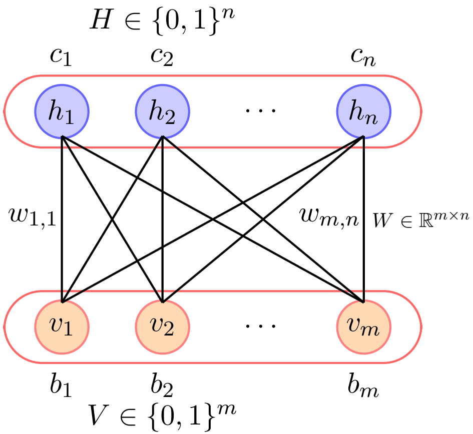
$$\ln \mathscr{L}(\theta) = \ln \prod_{i=1}^{l} p(x_i|\theta) = \sum_{i=1}^{l} \ln p(x_i|\theta)$$

where θ are the parameters



- Okay so we have the objective function now! What next?
- We need a learning algorithm
- We can just use gradient descent if we are able to compute the gradient of the loss function w.r.t. the parameters
- Let us see if we can do that

Module 19.6: Computing the gradient of the log likelihood



- We will just consider the loss for a single training example

$$\ln \mathscr{L}(\theta) = \ln p(V|\theta) = \ln \frac{1}{Z} \sum_H e^{-E(V,H)}$$

$$= \ln \sum_H e^{-E(V,H)} - \ln \sum_{V,H} e^{-E(V,H)}$$

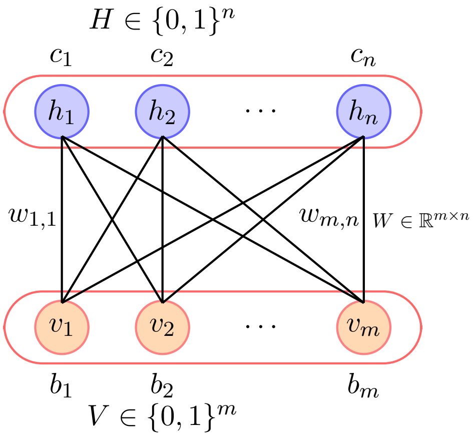
$$\frac{\partial \ln \mathscr{L}(\theta)}{\partial \theta} = \frac{\partial}{\partial \theta}\left(\ln \sum_H e^{-E(V,H)} - \ln \sum_{V,H} e^{-E(V,H)} \right)$$

$$= -\frac{1}{\sum_H e^{-E(V,H)}} \sum_H e^{-E(V,H)} \frac{\partial E(V,H)}{\partial \theta}$$

$$+ \frac{1}{\sum_{V,H} e^{-E(V,H)}} \sum_{V,H} e^{-E(V,H)} \frac{\partial E(V,H)}{\partial \theta}$$

$$= -\sum_H \frac{e^{-E(V,H)}}{\sum_H e^{-E(V,H)}} \frac{\partial E(V,H)}{\partial \theta}$$

$$+ \sum_{V,H} \frac{e^{-E(V,H)}}{\sum_{V,H} e^{-E(V,H)}} \frac{\partial E(V,H)}{\partial \theta}$$

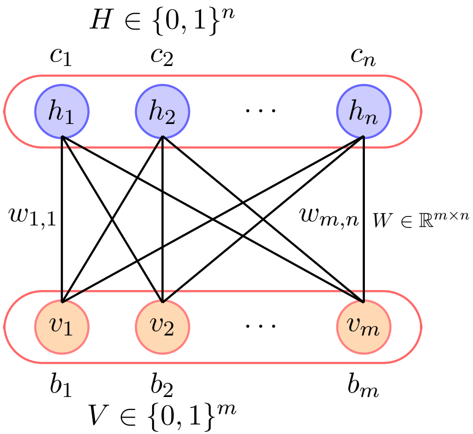


- Now,

$$\frac{e^{-E(V,H)}}{\sum_{V,H} e^{-E(V,H)}} = p(V,H)$$

$$\frac{e^{-E(V,H)}}{\sum_{H} e^{-E(V,H)}} = \frac{\frac{1}{Z}e^{-E(V,H)}}{\frac{1}{Z}\sum_{H} e^{-E(V,H)}}$$
$$= \frac{p(V,H)}{p(V)} = p(H|V)$$

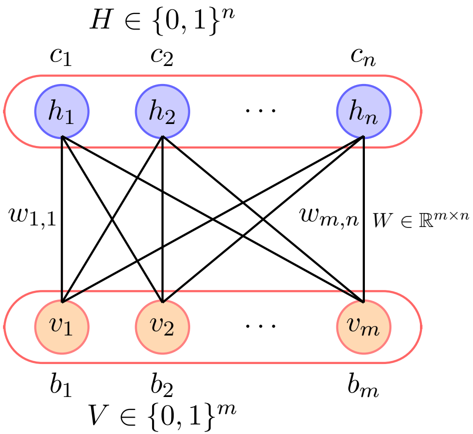
$$\frac{\partial \ln \mathscr{L}(\theta)}{\partial \theta} = -\sum_{H} \frac{e^{-E(V,H)}}{\sum_{H} e^{-E(V,H)}} \frac{\partial E(V,H)}{\partial \theta}$$
$$+ \sum_{V,H} \frac{e^{-E(V,H)}}{\sum_{V,H} e^{-E(V,H)}} \frac{\partial E(V,H)}{\partial \theta}$$
$$= -\sum_{H} p(H|V) \frac{\partial E(V,H)}{\partial \theta} + \sum_{V,H} p(V,H) \frac{\partial E(V,H)}{\partial \theta}$$



- Okay, so we have,

$$\frac{\partial \ln \mathscr{L}(\theta)}{\partial \theta} = -\sum_{H} p(H|V)\frac{\partial E(V,H)}{\partial \theta} + \sum_{V,H} p(V,H)\frac{\partial E(V,H)}{\partial \theta}$$

- Remember that θ is a collection of all the parameters in our model, i.e., $W_{ij}, b_i, c_j \forall i \in \{1,\ldots,m\}$ and $\forall j \in \{1,\ldots,n\}$
- We will follow our usual recipe of computing the partial derivative w.r.t. one weight w_{ij} and then generalize to the gradient w.r.t. the entire weight matrix W



$$\frac{\partial \mathscr{L}(\theta)}{\partial w_{ij}}$$

$$= -\sum_H p(H|V)\frac{\partial E(V,H)}{\partial w_{ij}} + \sum_{V,H} p(V,H)\frac{\partial E(V,H)}{\partial w_{ij}}$$

$$= \sum_H p(H|V)h_i v_j - \sum_{V,H} p(V,H)h_i v_j$$

$$= \mathbb{E}_{p(H|V)}[v_i h_j] - \mathbb{E}_{p(V,H)}[v_i h_j]$$

- We can write the above as a sum of two expectations

$$\frac{\partial \mathscr{L}(\theta)}{\partial w_{ij}} = \mathbb{E}_{p(H|V)}[v_i h_j] - \mathbb{E}_{p(V,H)}[v_i h_j]$$

- How do we compute these expectations?
- The first summation can actually be simplified (we will come back and simplify it later)
- However, the second summation contains an exponential number of terms and hence intractable in practice
- So how do we deal with this ?

Module 19.7: Motivation for Sampling

$$\frac{\partial \mathscr{L}(\theta)}{\partial w_{ij}} = \mathbb{E}_{p(H|V)}[v_i h_j] - \mathbb{E}_{p(V,H)}[v_i h_j]$$

- The trick is to approximate the sum by using a few samples instead of an exponential number of samples
- We will try to understand this with the help of an analogy

- Suppose you live in a city which has a population of 10M and you want to compute the average weight of this population
- You can think of X as a random variable which denotes a person
- The value assigned to this random variable can be any person from your population
- For each person you have an associated value denoted by $weight(X)$
- You are then interested in computing the expected value of $weight(X)$ as shown on the RHS

$$\mathbb{E}[weight(X)] = \sum_{(x \in P)} p(x)weight(x)$$

- Of course, it is going to be hard to get the weights of every person in the population and hence in practice we approximate the above sum by sampling only few subjects from the population (say 10000)

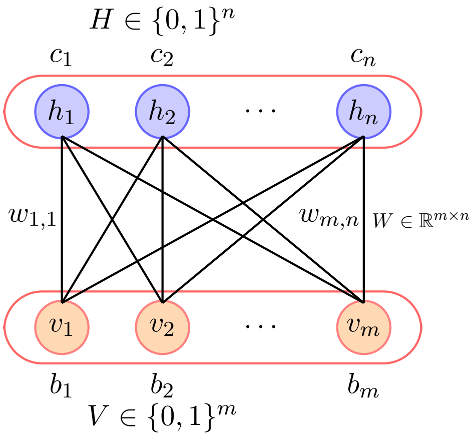
$$\mathbb{E}[weight(X)] \approx \frac{\sum_{x \in P[:10000]}[p(x)weight(x)]}{\sum_{x \in P[:10000]} p(x)}$$

- Further, you assume that $P(X) = \frac{1}{N} = \frac{1}{10K}$, i.e., every person in your population is equally likely

$$\mathbb{E}[weight(X)] \approx \frac{\sum_{x \in Persons[:10000]}[weight(x)]}{10^4}$$

$$\mathbb{E}[X] = \sum_{(x \in P)} xp(x)$$

- This looks easy, why can't we do the same for our task ?
- Why can't we simply approximate the sum by using some samples?
- What does that mean? It means that instead of considering all possible values of $\{v, h\} \in 2^{m+n}$ let us just consider some samples from this population
- Analogy: Earlier we had 10M samples in the population from which we drew $10K$ samples, now we have 2^{m+n} samples in the population from which we need to draw a reasonable number of samples
- Why is this not straightforward? Let us see!

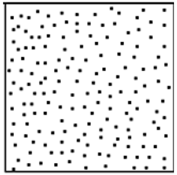


- For simplicity, first let us just focus on the visible variables ($V \in 2^m$) and let us see what it means to draw samples from $P(V)$
- Well, we know that $V = v_1, v_2, \ldots, v_m$ where each $v_i \in \{0, 1\}$
- Suppose we decide to approximate the sum by $10K$ samples instead of the full 2^m samples
- It is easy to create these samples by assigning values to each v_i
- For example, $V = 11111\ldots11111, V = 00000\ldots0000, V = 00110011\ldots00110011, \ldots V = 0101\ldots0101$ are all samples from this population
- So which samples do we consider ?

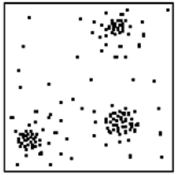
Likely

Unlikely

- Well, that's where the catch is!
- Unlike, our population analogy, here we cannot assume that every sample is equally likely
- Why? (Hint: consider the case that visible variables correspond to pixels from natural images)
- Clearly some images are more likely than the others!
- Hence, we cannot assume that all samples from the population ($V \in 2^m$) are equally likely



Uniform distribution



Multimodal distribution

- Let us see this in more detail
- In our analogy, every person was equally likely so we could just sample people uniformly randomly
- However, now if we sample people uniformly randomly then we will not get the true picture of the expected value
- We need to draw more samples from the high probability region and fewer samples from the low probability region
- In other words each sample needs to be drawn in proportion to its probability and not uniformly

$$\frac{\partial \mathscr{L}(\theta|V)}{\partial w_{ij}} = \mathbb{E}_{p(H|V)}[v_i h_j] - \mathbb{E}_{p(V,H)}[v_i h_j]$$

$$Z = \sum_V \sum_H \Big(\prod_i \prod_j \phi_{ij}(v_i, h_j) \prod_i \psi_i(v_i) \prod_j \xi_j(h_j) \Big)$$

- That is where the problem lies!
- To draw a sample (V, H), we need to know its probability $P(V, H)$
- And of course, we also need this $P(V, H)$to compute the expectation
- But, unfortunately computing $P(V, H)$ is intractable because of the partition function Z
- Hence, approximating the summation by using a few samples is not straightforward! (or rather drawing a few samples from the distribution is hard!)

The story so far

- Conclusion: Okay, I get it that drawing samples from this distribution P is hard.
- Question: Is it possible to draw samples from an easier distribution (say, Q) as long as I am sure that if I keep drawing samples from Q eventually my samples will start looking as if they were drawn from P!
- Answer: Well if you can actually prove this then why not? (and that's what we do in Gibbs Sampling)