Restricted Boltzmann Machines

Data Science @ Jellyfish - Journal Club

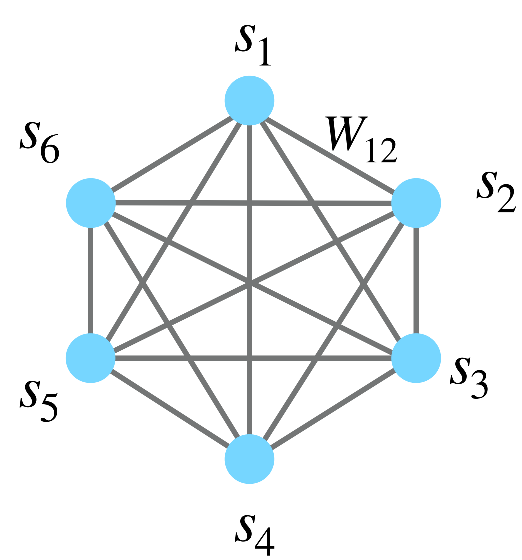
Mobolaji Williams, October 6, 2020

Boltzmann Machines

Boltzmann Machines (def): Class of models that allow you to find patterns in binary data by representing the data as state vectors in a statistical physics problem.

	1	2	3	4
A	+	−	−	+
B	−	+	−	−
C	+	−	+	−
D	−	+	−	+

Data Set



$$E(\{s_i\}) = -\sum_{i,j} s_i W_{ij} s_j - \sum_i b_i s_i$$

Boltzmann Machine Model

W_{ij} : Roughly how correlated column i is with column j

b_i : Roughly how biased column i is to positive values

Inference

Boltzmann Machines - Example

Setup: Say we have a table that lists movies watched and whether a viewer liked them or not

	LotR	Harry Potter	Alien	Moana	Get Out	MIB
Alice	Yes	No	No	Yes	Yes	No
Bob	Yes	Yes	Yes	No	No	No
Carl	Yes	No	No	Yes	No	Yes
David	?	?	Yes	Yes	?	?

Question:
If we have a new viewer David and we know that he likes "Alien" and "Moana", (but we don't know any other movies he's seen), how can we recommend movies to him?

Answer (From Boltzmann Machines):

1. Represent current user preferences as vectors with binary values
2. Assume the values have a particular interaction weight with each other and a particular bias in a certain direction; Use to define probability
3. Find weights and biases most likely to produce data
4. Use the learned weights and biases to predict probability for that viewer likes unseen movies

Boltzmann Machines - Example

Setup: Say we have a table that lists movies watched and whether a viewer liked them or not

Question:
If we have a new viewer David and we know that he likes "Alien" and "Moana", (but we don't know any other movies he's seen), how can we recommend movies to him?

1. Represent current user preferences as vectors with binary values

	LotR	Harry Potter	Alien	Moana	Get Out	MIB
Alice	Yes	No	No	Yes	Yes	No
Bob	Yes	Yes	Yes	No	No	No
Carl	Yes	No	No	Yes	No	Yes

$\vec{s}_A = [1,0,0,1,1,0]$

$\vec{s}_B = [1,1,1,0,0,0]$

$\vec{s}_C = [1,0,0,1,0,1]$

$$\vec{s} = [s_1, s_2, s_3, s_4, s_5, s_6]$$

2. Assume the values have a <u>particular interaction weight</u> with each other and a <u>particular bias</u> in a certain direction; Use to define probability

$$E(\vec{s}) = -\sum_{i,j=1}^{N} s_i W_{ij} s_j - \sum_{i=1}^{N} b_i s_i$$

Weight: Defines tendency for both i and j to both be "on"

Bias: Defines tendency for i to be "on" independent of other values

Probability to see data row $\vec{s}$:

$$P(\vec{s}) = \exp(-E(\vec{s}))/Z \quad \text{where} \quad Z = \sum_{s_1=0}^{1} \cdots \sum_{s_N=0}^{1} \exp(-E(\{s_i\}))$$

In Statistical Physics, this is the "Boltzmann Distribution"

Boltzmann Machines - Example

Setup: Say we have a table that lists movies watched and whether a viewer liked them or not

1. Represent current user preferences as vectors with binary values

$$\vec{s}_A = [1, 0, 0, 1, 1, 0]$$

$$\vec{s}_B = [1, 1, 1, 0, 0, 0]$$

$$\vec{s}_C = [1, 0, 0, 1, 0, 1]$$

Question:
If we have a new viewer David and we know that he likes "Alien" and "Moana", (but we don't know any other movies he's seen), how can we recommend movies to him?

2. Assume the values have a <u>particular interaction weight</u> with each other and a <u>particular bias</u> in a certain direction; Use to define probability

$$E(\vec{s}) = -\sum_{i,j=1}^{N} s_i W_{ij} s_j - \sum_{i=1}^{N} b_i s_i$$

$$P(\vec{s}) = \exp(-E(\vec{s}))/Z$$

$$Z = \sum_{s_1=0}^{1} \cdots \sum_{s_N=0}^{1} \exp(-E(\{s_i\}))$$

3. Find weights and biases most likely to produce data

Likelihood of data is

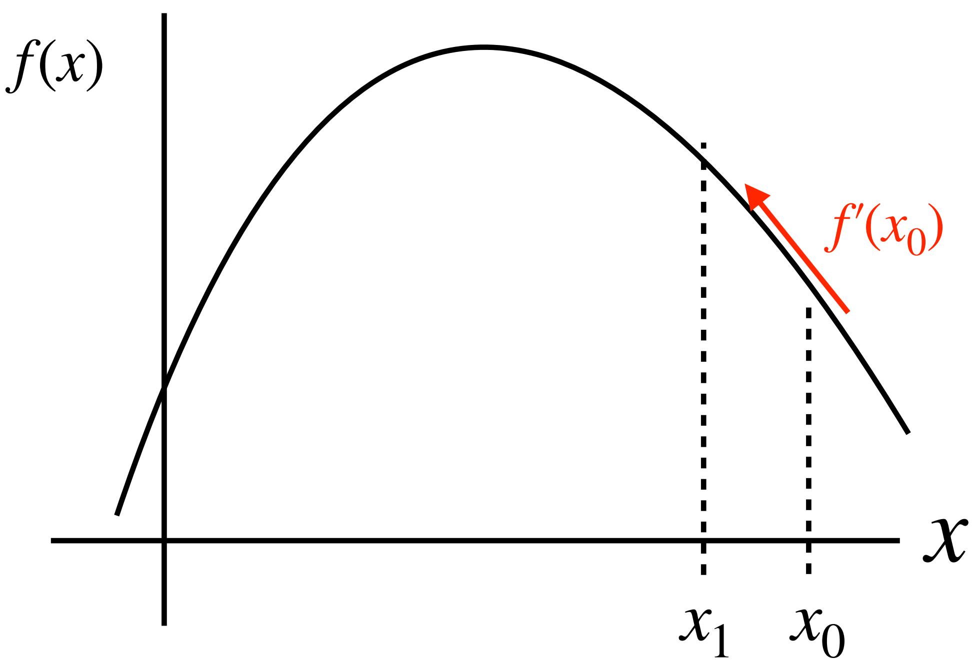
$$\prod_{\alpha=1}^{M} P(\vec{s}^{\alpha})$$

α : denotes data row

(M is number of rows)

What W_{ij} and b_i maximize this likelihood? → Find the answer with gradient ascent!

Gradient Descent Review



For a function $f(x)$, how can we numerically find the $\bar{x}$ that maximizes $f(x)$?

General Gradient Ascent Algorithm

1. Choose random x_0
2. Compute $f'(x_0)$
3. Compute new value $x_1 = x_0 + \lambda f'(x_0)$
4. Return to 2. and iterate until convergence

learning rate

For the Boltzmann Machine

1. Initialize $\{W_{ij}\}$ and $\{b_i\}$ randomly

2. Computing derivatives

$$\ln \mathscr{L} = \frac{1}{M}\sum_{\alpha=1}^{M} \ln P(\vec{s}^{\alpha})$$

$$\frac{\partial}{\partial W_{ij}} \ln \mathscr{L} = -\frac{1}{M}\sum_{\alpha=1}^{M}\sum_{i,j}^{N} s_i^{\alpha} s_j^{\alpha} - \sum_{\{s_i\}} P(\vec{s}) s_i s_j \quad = -\left(\langle s_i s_j \rangle_{\text{data}} - \langle s_i s_j \rangle_{\text{model}}\right)$$

$$\frac{\partial}{\partial b_i} \ln \mathscr{L} = -\frac{1}{M}\sum_{\alpha=1}^{M}\sum_{i}^{N} s_i^{\alpha} - \sum_{\{s_i\}} P(\vec{s}) s_i \quad = -\left(\langle s_i \rangle_{\text{data}} - \langle s_i \rangle_{\text{model}}\right)$$

* Maximizing $c_1 \ln f(x)$ is equivalent to maximizing $f(x)$

Gradient Ascent for Boltzmann Machines

Gradient Ascent Algorithm for the Boltzmann Machine

1. Initialize $\{W_{ij}\}$ and $\{b_i\}$ randomly

2. Compute derivatives

$$\ln \mathscr{L} = \frac{1}{M} \sum_{\alpha=1}^{M} \ln P(\vec{s}^{\alpha})$$

$$\frac{\partial}{\partial W_{ij}} \ln \mathscr{L} = -\left(\langle s_i s_j \rangle_{\text{data}} - \langle s_i s_j \rangle_{\text{model}} \right)$$

$$\frac{\partial}{\partial b_i} \ln \mathscr{L} = -\left(\langle s_i \rangle_{\text{data}} - \langle s_i \rangle_{\text{model}} \right)$$

3. Increment weights and biases

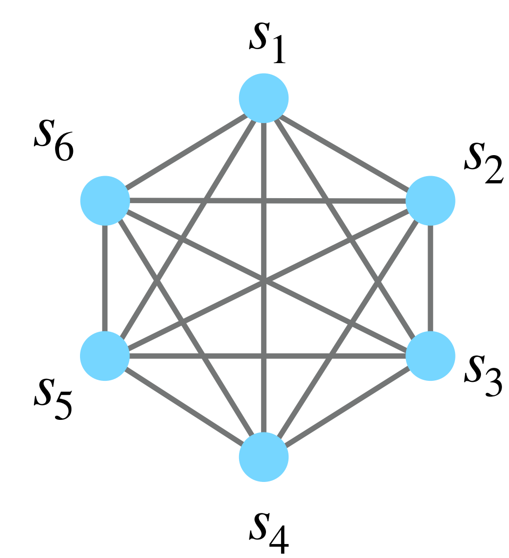
$$W_{ij} \to W_{ij} + \lambda \frac{\partial}{\partial W_{ij}} \ln \mathscr{L}$$

$$b_i \to b_i + \lambda \frac{\partial}{\partial b_i} \ln \mathscr{L}$$

4. Iterate until convergence

But there's a problem!

Computing the expectation values for the model requires a summation over all 2^N states



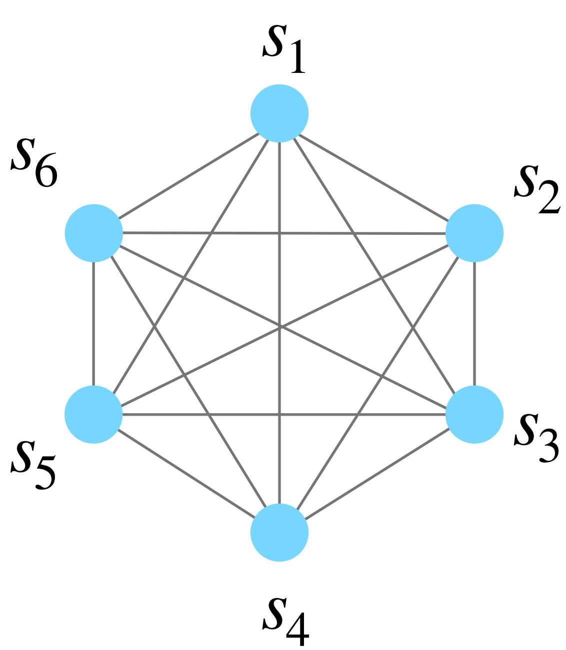
Sum over $s_i = \{0,1\}$
for $i = 1,\ldots,N$

This makes training the Boltzmann Machine computationally expensive !

This is where **Restricted Boltzmann Machines** come in

Boltzmann Machines vs. Restricted Boltzmann Machines

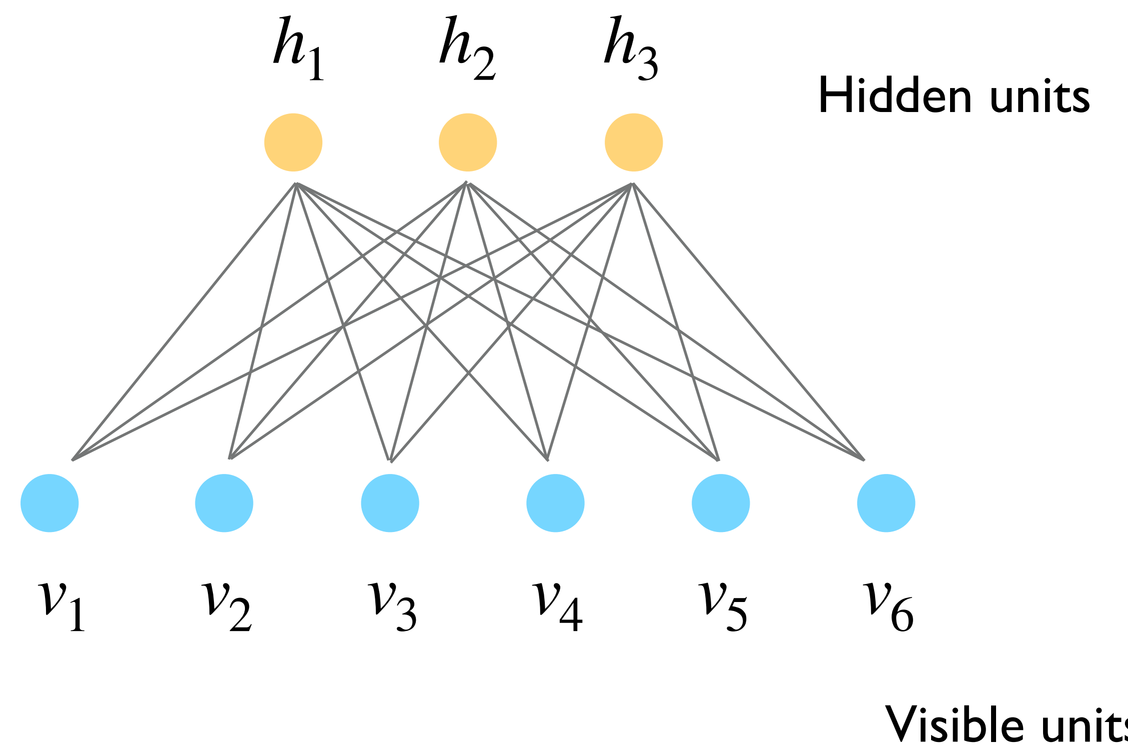
Boltzmann Machine



In a general Boltzmann Machine all of the **visible units** are connected to each other

(Visible units represent the data)

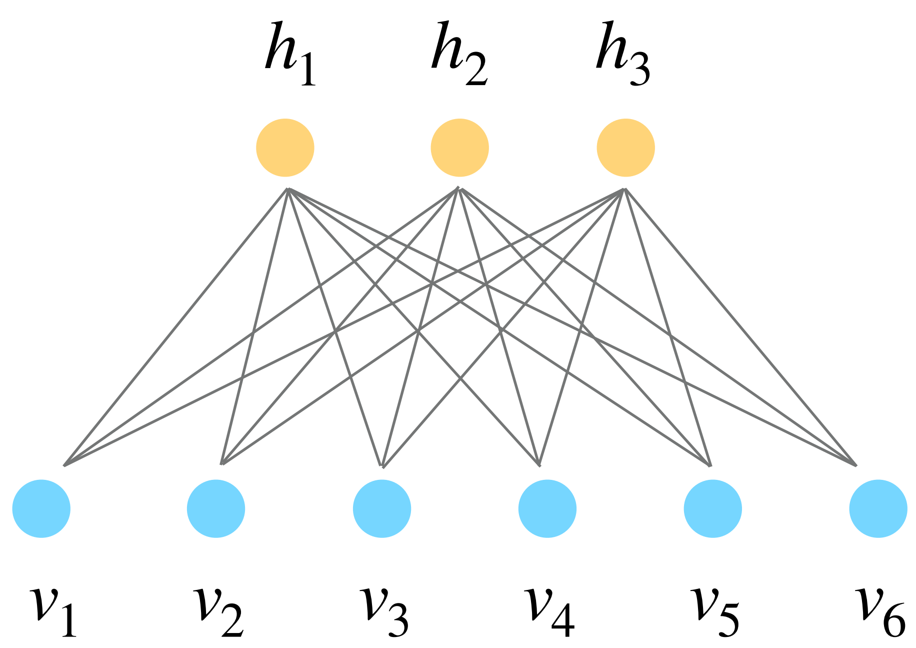
Restricted Boltzmann Machine



In a Restricted Boltzmann Machine the **<u>visible units</u>** only connect to **<u>hidden units</u>.** These **hidden units** are not associated with the data

Restricted Boltzmann Machines

Restricted Boltzmann Machine



The Energy Function

$$E(\vec{v}, \vec{h}) = -\sum_{i=1}^{N}\sum_{j=1}^{L} v_i W_{ij} h_j - \sum_{i=1}^{N} b_i v_i - \sum_{j=1}^{L} c_j h_j$$

Probability $P(\vec{v}, \vec{h}) = \exp(-E(\vec{v}, \vec{h}))/Z$

where $Z = \sum_{\vec{v}} \sum_{\vec{h}} \exp(-E(\vec{v}, \vec{h}))$

The visible units are independent of each other given the hidden units and vice versa.

In a Restricted Boltzmann Machine the **visible units** only connect to **hidden units**. These **hidden units** are not associated with the data

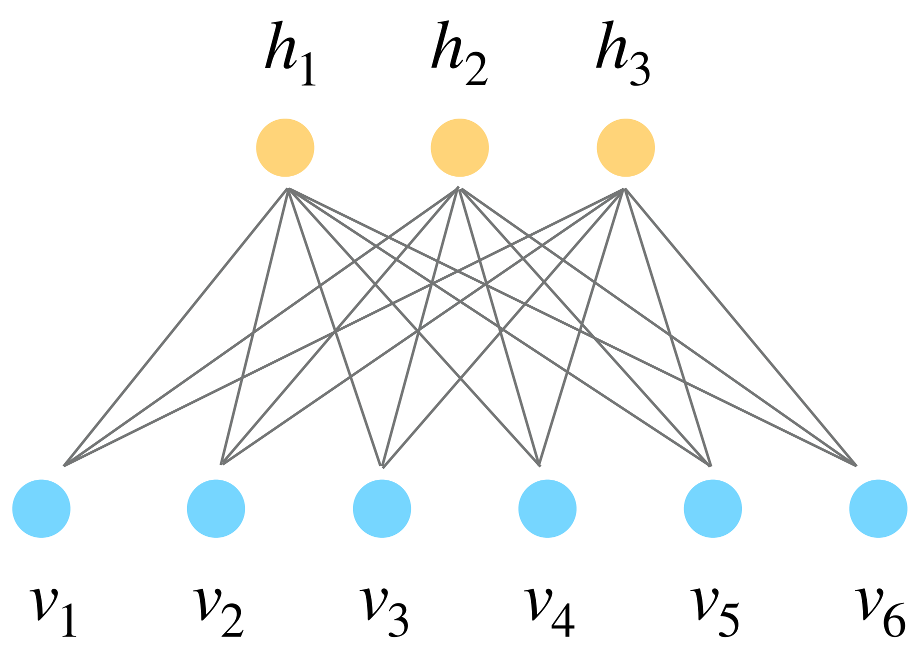
In particular: The probability that a visible element is activated given a hidden vector is simple to write (as is the converse)

$$P(h_j = 1 \mid \vec{v}) = \frac{1}{1 + e^{-(c_j + \sum_{i=1} W_{ij} v_i)}} = \sigma\Big(c_j + \sum_{i=1} W_{ij} v_i\Big)$$

$$P(v_i = 1 \mid \vec{h}) = \frac{1}{1 + e^{-(b_i + \sum_{j=1} W_{ij} h_j)}} = \sigma\Big(b_i + \sum_{j=1} W_{ij} h_j\Big)$$

Training Restricted Boltzmann Machines

Restricted Boltzmann Machine



The Energy Function

$$E(\vec{v},\vec{h}) = -\sum_{i=1}^{N}\sum_{j=1}^{L} v_i W_{ij} h_j - \sum_{i=1}^{N} b_i v_i - \sum_{j=1}^{L} c_j h_j$$

We train restricted Boltzmann machines in a way similar to how we train Boltzmann machines

We want to find the $\{W_{ij}\}$, $\{b_i\}$ and $\{c_j\}$ that maximize $P(\vec{v},\vec{h})$, except now this probability isn't uniquely determined by the data (since $\vec{h}$ is hidden)

Presuming $\vec{h}$ was not hidden, we would compute the weight and bias updates as

$$W_{ij} \to W_{ij} + \lambda \frac{\partial}{\partial W_{ij}} \ln \mathscr{L}$$

$$b_i \to b_i + \lambda \frac{\partial}{\partial b_i} \ln \mathscr{L}$$

$$c_j \to c_j + \lambda \frac{\partial}{\partial c_j} \ln \mathscr{L}$$

where

$$\frac{\partial}{\partial W_{ij}} \ln \mathscr{L} = -\left(\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}}\right)$$

$$\frac{\partial}{\partial b_i} \ln \mathscr{L} = -\left(\langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{model}}\right)$$

$$\frac{\partial}{\partial c_j} \ln \mathscr{L} = -\left(\langle h_j \rangle_{\text{data}} - \langle h_j \rangle_{\text{model}}\right)$$

Training Restricted Boltzmann Machines

Restricted Boltzmann Machine

$$W_{ij} \to W_{ij} + \lambda \frac{\partial}{\partial W_{ij}} \ln \mathscr{L}$$

$$b_i \to b_i + \lambda \frac{\partial}{\partial b_i} \ln \mathscr{L}$$

$$c_j \to c_j + \lambda \frac{\partial}{\partial c_j} \ln \mathscr{L}$$

where

$$\frac{\partial}{\partial W_{ij}} \ln \mathscr{L} = -\left(\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}} \right)$$

$$\frac{\partial}{\partial b_i} \ln \mathscr{L} = -\left(\langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{model}} \right)$$

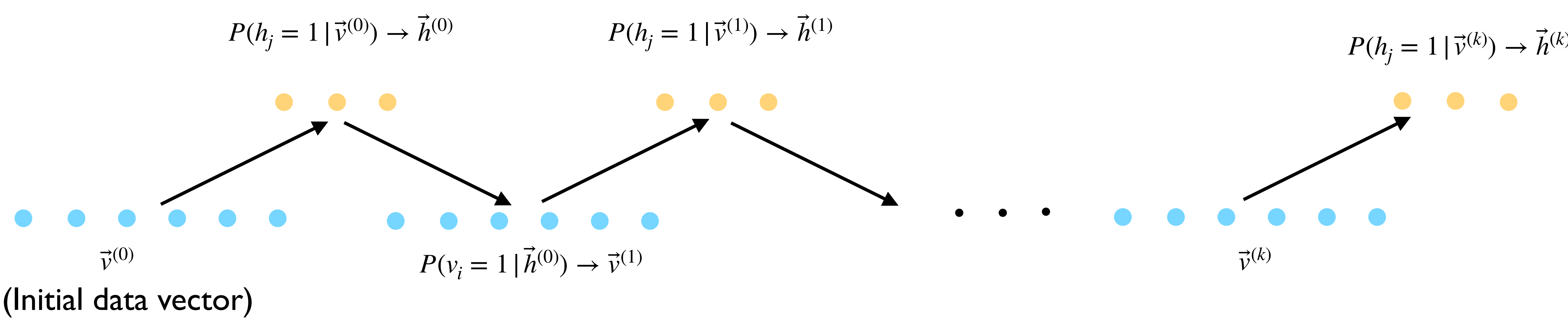
$$\frac{\partial}{\partial c_j} \ln \mathscr{L} = -\left(\langle h_j \rangle_{\text{data}} - \langle h_j \rangle_{\text{model}} \right)$$

We can’t determine hidden states directly from the data

So we sample them given our visible states $\longrightarrow$

$$P(h_j = 1 \,|\, \vec{v}) = \frac{1}{1 + e^{-(c_j + \sum_{i=1} W_{ij} v_i)}} = \sigma\left(c_j + \sum_{i=1} W_{ij} v_i \right)$$

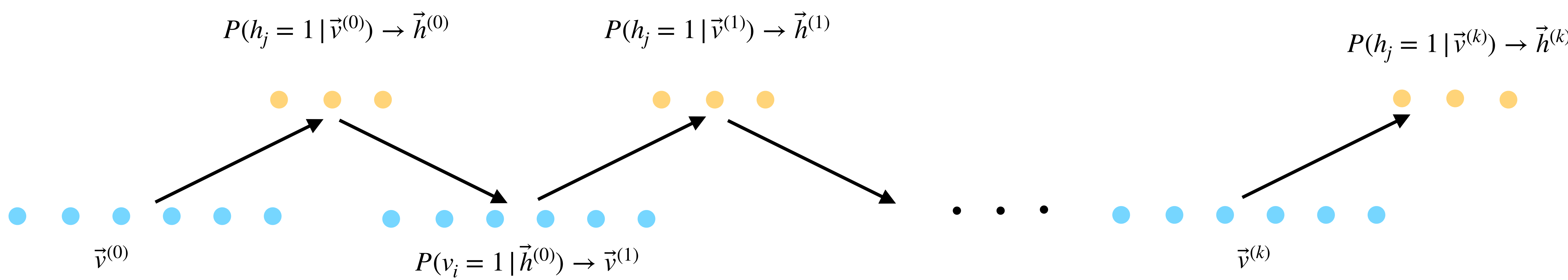
$$P(v_i = 1 \,|\, \vec{h}) = \frac{1}{1 + e^{-(b_i + \sum_{j=1} W_{ij} h_j)}} = \sigma\left(b_i + \sum_{j=1} W_{ij} h_j \right)$$



Gibbs Sampling and Contrastive Divergence

Restricted Boltzmann Machine

Sampling our hidden states and then visible states and then hidden states again (and so on) in this way is called <u>Gibbs Sampling</u>



We cut this procedure off at a certain number (k) of iterations, and use it to estimate our expectation values

This procedure is termed <u>Contrastive Divergence</u>

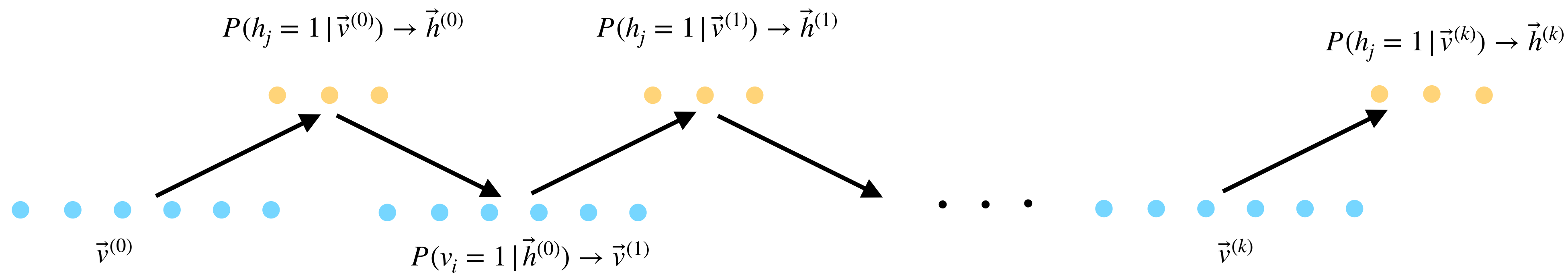
$$\frac{\partial}{\partial W_{ij}} \ln \mathcal{L} = -\left(\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}}\right) \quad \rightarrow -\left(v_i^{(0)} p(h_j = 1 \,|\, \vec{v}^{(0)}) - v_i^{(k)} p(h_j = 1 \,|\, \vec{v}^{(k)})\right)$$

$$\frac{\partial}{\partial b_i} \ln \mathcal{L} = -\left(\langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{model}}\right) \quad \rightarrow -\left(v_i^{(0)} - v_i^{(k)}\right)$$

$$\frac{\partial}{\partial c_j} \ln \mathcal{L} = -\left(\langle h_j \rangle_{\text{data}} - \langle h_j \rangle_{\text{model}}\right) \quad \rightarrow -\left(p(h_j = 1 \,|\, \vec{v}^{(0)}) - p(h_j = 1 \,|\, \vec{v}^{(k)})\right)$$

Contrastive Divergence Summary

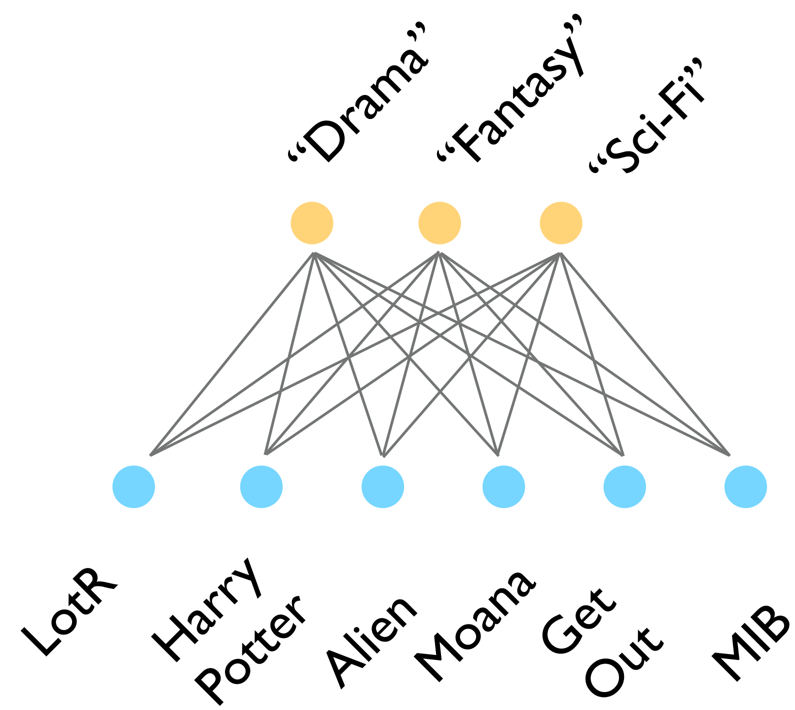
Restricted Boltzmann Machine



(Initial data vector)

After many iterations of training, we have the weights and biases that make our data most likely.

The hidden states function as a condensed representation of the visible states



Contrastive Divergence Algorithm

$$W_{ij} \to W_{ij} - \lambda \left(v_i^{(0)} p(h_j = 1 \,|\, \vec{v}^{(0)}) - v_i^{(k)} p(h_j = 1 \,|\, \vec{v}^{(k)}) \right)$$

$$b_i \to b_i - \lambda \left(v_i^{(0)} - v_i^{(k)} \right)$$

$$c_j \to c_j - \lambda \left(p(h_j = 1 \,|\, \vec{v}^{(0)}) - p(h_j = 1 \,|\, \vec{v}^{(k)}) \right)$$

What can we do with this trained model?

Back to Movie Reviews

Restricted Boltzmann Machine

	LotR	Harry Potter	Alien	Moana	Get Out	MIB
Alice	Yes	No	No	Yes	Yes	No
Bob	Yes	Yes	Yes	No	No	No
Carl	Yes	No	No	Yes	No	Yes
David	?	?	Yes	Yes	?	?

Question:
If we have a new viewer David and we know that he likes "Alien" and "Moana", (but we don't know any other movies he's seen), how can we recommend movies to him?

Back to Movie Reviews

Restricted Boltzmann Machine

	LotR	Harry Potter	Alien	Moana	Get Out	MIB
Alice	Yes	No	No	Yes	Yes	No
Bob	Yes	Yes	Yes	No	No	No
Carl	Yes	No	No	Yes	No	Yes
David	?	?	Yes	Yes	?	?

Train the RBM on the given data set

$\vec{v}_A = [1, 0, 0, 1, 1, 0]$

$\vec{v}_B = [1, 1, 1, 0, 0, 0]$

$\vec{v}_C = [1, 0, 0, 1, 0, 1]$

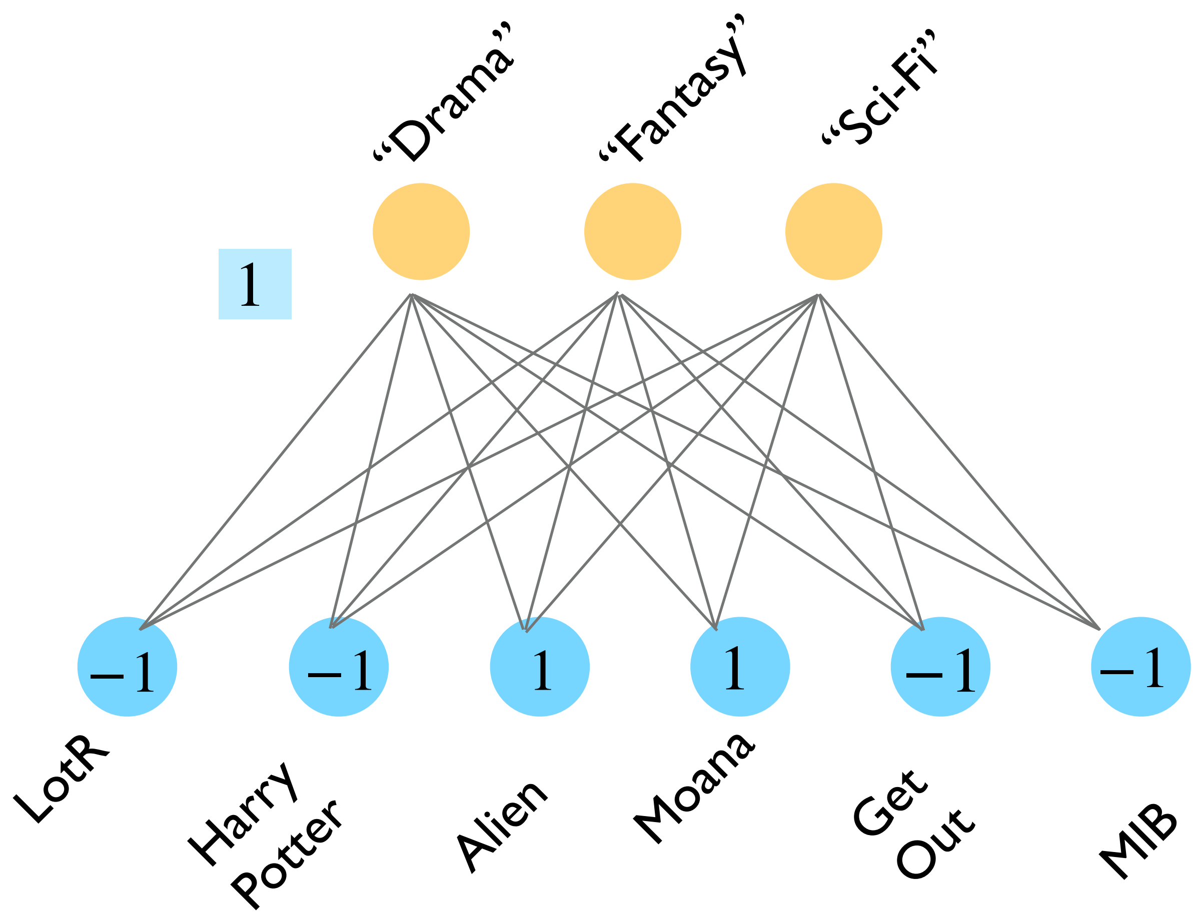
1 Define the unknown vector (with -1 for missing values)

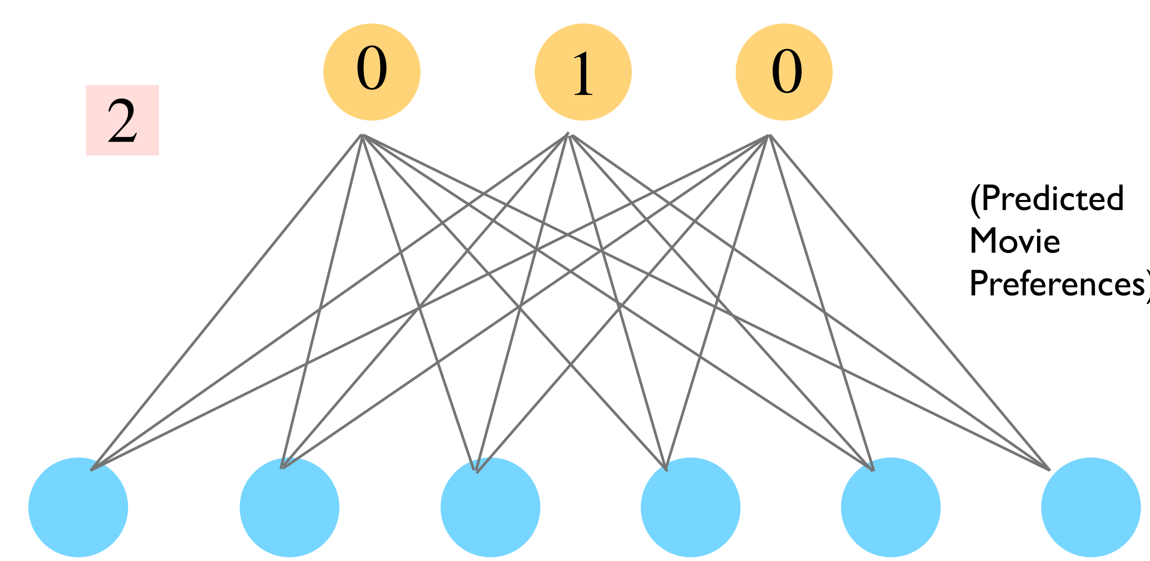
$\vec{v}_D = [-1, -1, 1, 1, -1, -1]$

(*Not sure why -1 is used for missing values)

2 Predict value of hidden state $p(h_j = 1 \mid \vec{v}_D) \rightarrow \vec{h}$

3 Use predicted hidden state to predict a new visible state $p(v_i = 1 \mid \vec{h}) \rightarrow \vec{v}$





(Predicted Movie Preferences)

Back to Movie Reviews

Restricted Boltzmann Machine

	LotR	Harry Potter	Alien	Moana	Get Out	MIB
Alice	Yes	No	No	Yes	Yes	No
Bob	Yes	Yes	Yes	No	No	No
Carl	Yes	No	No	Yes	No	Yes
David	?	?	Yes	Yes	?	?

Train the RBM on the given data set

$\vec{v}_A = [1, 0, 0, 1, 1, 0]$

$\vec{v}_B = [1, 1, 1, 0, 0, 0]$

$\vec{v}_C = [1, 0, 0, 1, 0, 1]$

1

Define the unknown vector (with -1 for missing values)

$\vec{v}_D = [-1, -1, 1, 1, -1, -1]$

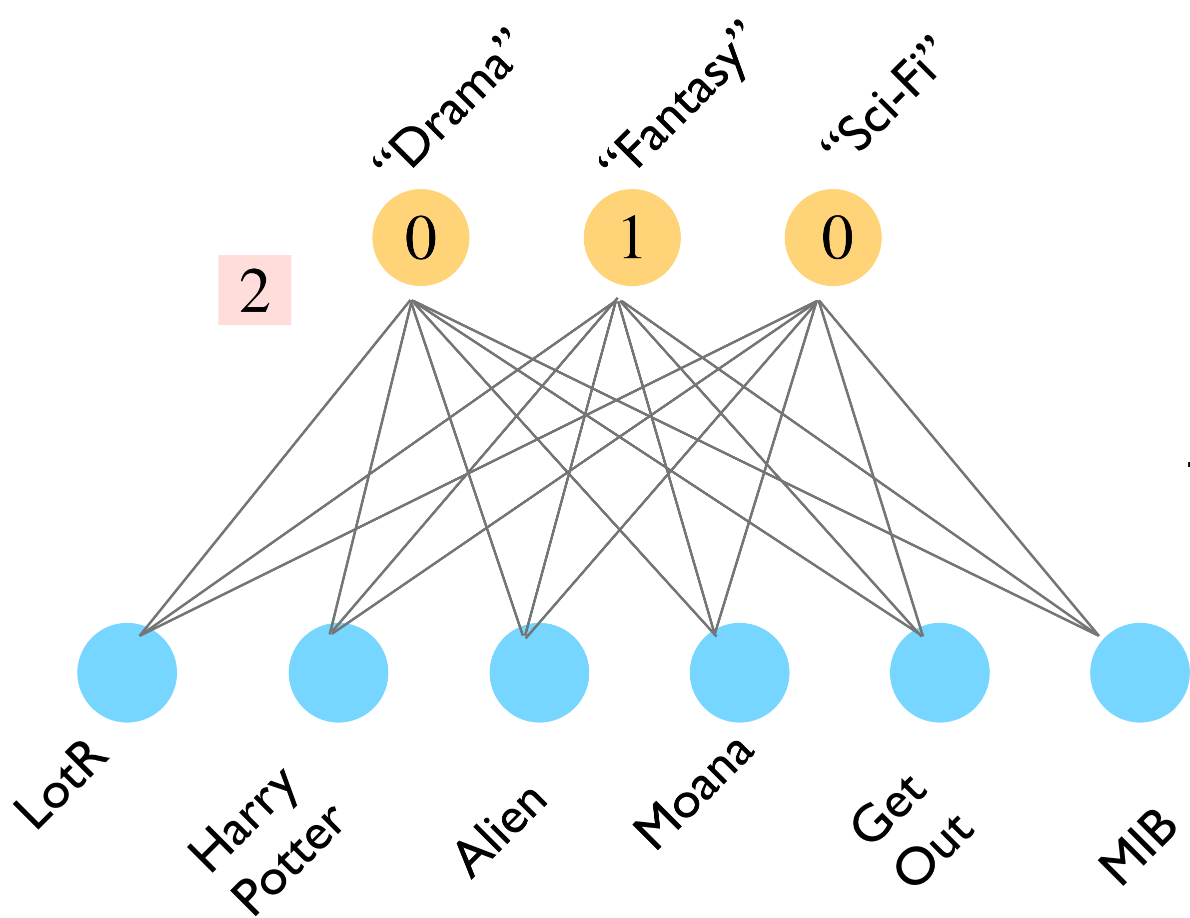
(*Not sure why -1 is used for missing values)

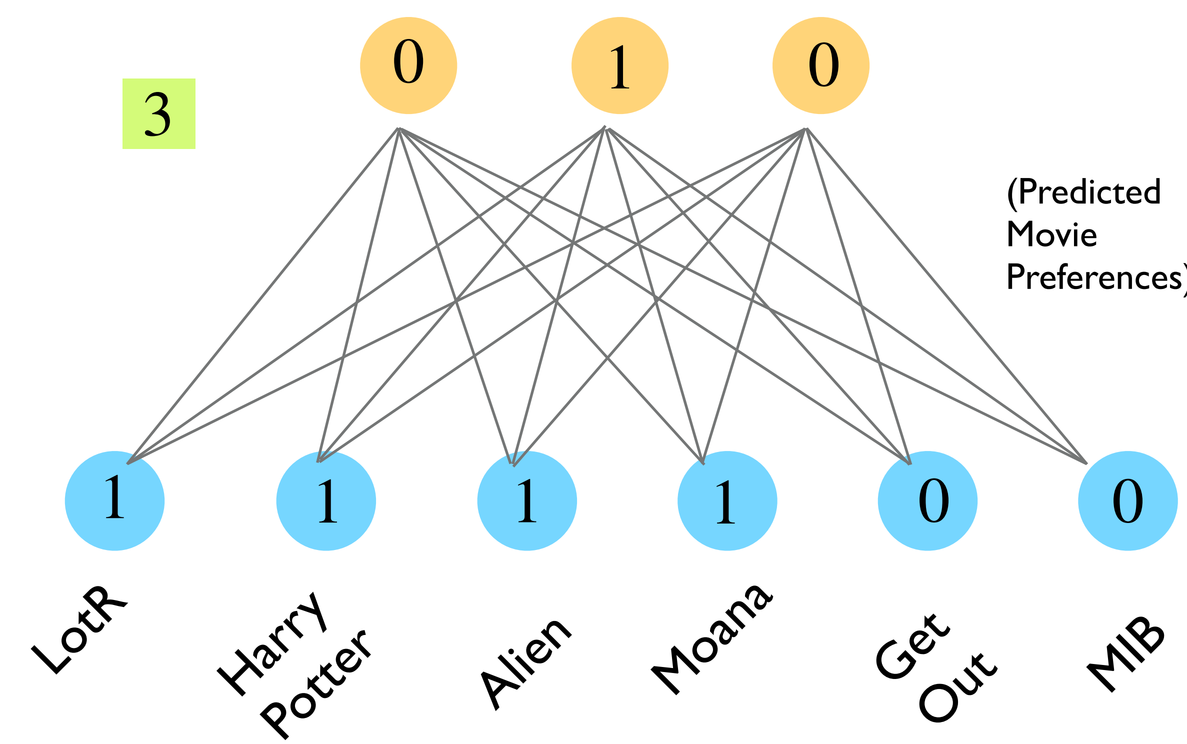
2

Predict value of hidden state $p(h_j = 1 \mid \vec{v}_D) \rightarrow \vec{h}$

3

Use predicted hidden state to predict a new visible state $p(v_i = 1 \mid \vec{h}) \rightarrow \vec{v}$





General Uses of Restricted Boltzmann Machines

Finding Correlations in Data

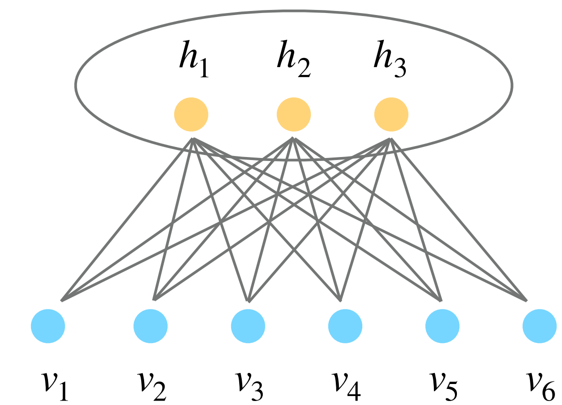
General Uses of Restricted Boltzmann Machines

Besides being used for recommender systems, RBMs are also useful for

Dimensionality reduction

With trained $\{W_{ij}\}$, $\{b_i\}$, and $\{c_j\}$, we can compute the hidden state $\vec{h}$ for each visible state $\vec{v}$, and use the hidden state as a feature vector for another model

Dimensionally Reduced Features



Factor Analysis

With trained $\{W_{ij}\}$ we can identify factors that separate our data elements

Example

	Bias Unit	Hidden 1	Hidden 2
Harry Potter	-0.82602559	-7.08986885	4.96606654
Avatar	-1.84023877	-5.18354129	2.27197472
LOTR 3	3.92321075	2.51720193	4.11061383
Gladiator	0.10316995	6.74833901	-4.00505343
Titanic	-0.97646029	3.25474524	-5.59606865
Glitter	-4.44685751	-2.81563804	-2.91540988

Hidden 2 is highly activated for "Fantasy Films"

(Weights)

Most people like LotR3

Hidden 1 is highly activated for "Oscar Winners"

Most people dislike Glitter

(biases)

[(http://blog.echen.me/2011/07/18/introduction-to-restricted-boltzmann-machines/)](http://blog.echen.me/2011/07/18/introduction-to-restricted-boltzmann-machines/)

Applications of RBMs to Jellyfish

Applications to Jellyfish

The simplest RBMs require binary-valued data which makes most of our continuous valued data (e.g., metrics), unusable for RBMs without data processing

(Person Metrics)

	PRs	Issues Resolved	Coding days	Confl. edits
Adam	5	3	2.5	10
Beth	2	8	5.1	0
Cathy	7	3	4.3	5
Diana	1	3	3	15

But we establish cutoffs to define "ones" and "zeros" for above and below average values

	PRs	Issues Resolved	Coding days	Confl. edits
Adam	1	0	0	1
Beth	0	1	1	0
Cathy	1	0	1	0
Diana	0	0	0	1

We can then use RBMs to

- make predictions about missing metric data
- find condensed representations of metric data

Applications of RBMs to Jellyfish

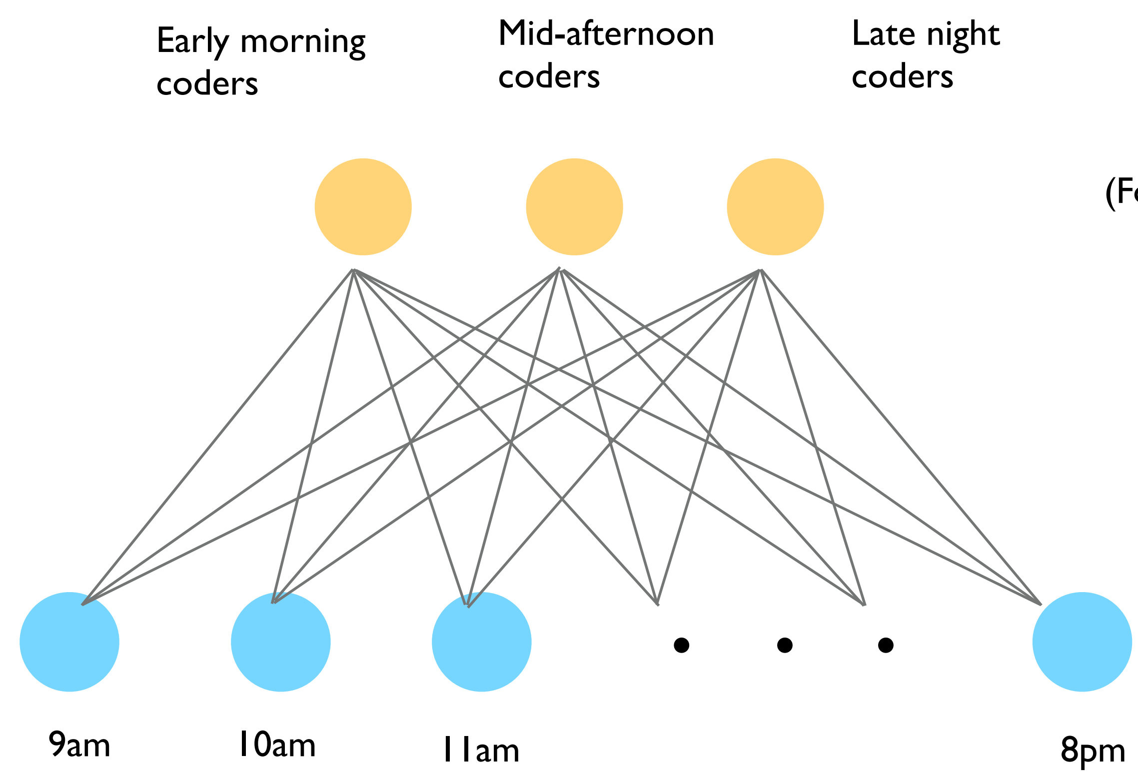
Applications to Jellyfish

We can use RBMs to find condensed representation of coding time data. Could allow us to determine general trends for when people are coding

(Commit Activity by Time)

	9am	10am	11am	12pm	. . .
Adam	1	0	0	1	
Beth	0	1	1	0	. . .
Cathy	0	1	0	1	
Diana	1	0	0	0	

1 or 0 depending on if person made a commit in that hour



(For Example)

End