Columbia University

Statistical NLP for the Web

Deep Belief Networks

Sameer Maskey

Week 9, October 31, 2012

Announcements

- Please ask HW2 related questions in courseworks
- HW2 due date has been moved to Nov 1 (Friday)
- HW3 will be released next week
- Monday and Tuesday are University Holidays
- Reading Assignments for next class posted in the course website

Intermediate Report II

- 20% of the Final project grade
- Intermediate Report II will be Oral
- Will put out time slots, please choose your appointment
- Everything related with your project is fair game, including theory related with algorithms you are using
- You need to have the first version of end to end system ready including UI

Looking Back

- We have learned various topics of Statistical NLP/Machine Learning
  - Classify text
  - Score text
  - Cluster text
  - Generate text
  - Represent text

Methods of Classification

-Fisher's Discriminant
- Perceptron
-Naïve Bayes

-Document Classification
-Text Classification
-Bag of Words Model

Looking Back

- We have learned various topics of Statistical NLP/Machine Learning
  - Classify text
  - Score text
  - Cluster text
  - Generate text
  - Represent text

Linear Methods of **Regression**

-Simple Regression
-Multiple Linear Regression

-Text Scoring
-Scoring Reviews

Looking Back

- We have learned various topics of Statistical NLP/Machine Learning
  - Classify text
  - Score text
  - Cluster text
  - Generate text
  - Represent text

Unsupervised Algorithms and Clustering

-K-Means
-Expectation Maximization

-Document Clustering
-Text Clustering

Looking Back

- We have learned various topics of Statistical NLP/Machine Learning
  - Classify text
  - Score text
  - Cluster text
  - Generate text
  - Represent text

Text Generation and Probabilistic Context Free Grammars

-Parsing Algorithm
-Writing your Grammar

Looking Back

- We have learned various topics of Statistical NLP/Machine Learning
  - Classify text
  - Score text
  - Cluster text
  - Generate text
  - Represent text

Features Beyond Words

Deep Networks
Restricted Boltzmann Machines
Neural Networks
Feature Vectors

Looking Back

- We have learned various topics of Statistical NLP/Machine Learning

  - Classify text
  - Score text
  - Cluster text
  - Generate text
  - Represent text

Features Beyond Words

Deep Networks
Restricted Boltzmann Machines
Neural Networks
Feature Vectors

Additionally we looked at how to scale these algorithms for large amount of data with using MapReduce

Going Forward

- All these machineries we have learned in Statistical NLP and Machine Learning will help us in learning more complex NLP tasks

- EM for Machine Translation
- Viterbi for Synchronous Grammar Decoding
- Feature Representation for Question Answering

Going Forward

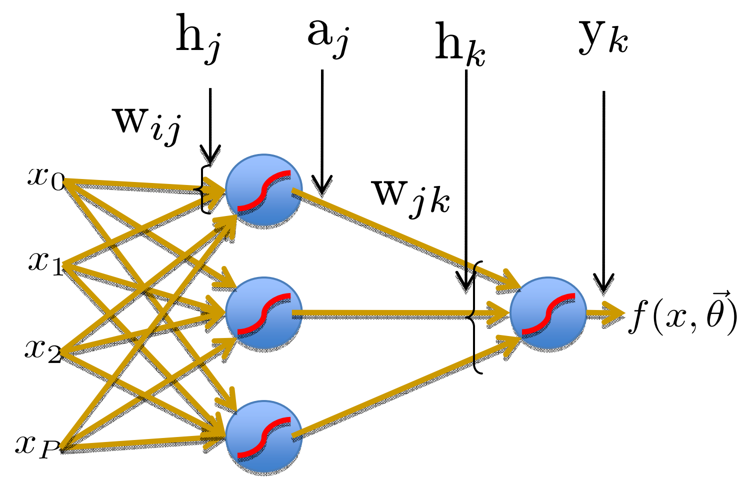
- (Week 9) Machine Translation I
- (Week 10) Machine Translation II and Log Linear Models
- (Week 11) Question Answering, Decoding Watson
- (Week 12) Dialog System, Decoding Siri
- (Week 13) Information Retrieval, Decoding Search Engines
- (Week 14) Equations to Implementations
- (Week 15) Project Due
- (Week 16) Finals Week (No Classes)

Reading Assignments will get more important as we may discuss assigned papers in the class!

Topics for Today

- Deep Networks
- Machine Translation

Neural Net Algorithm : Forward Phase



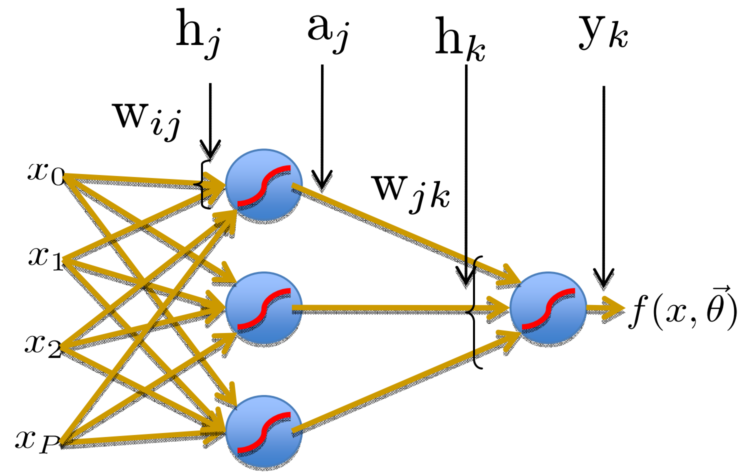
$$h_j = \sum_i x_i w_{ij}$$

$$a_j = g(h_j) = 1/(1+e^{-\beta h_j})$$

$$\mathrm{h}_k = \sum_j a_j w_{jk}$$

$$\mathrm{y}_k = g(h_k) = 1/(1+e^{-\beta h_k})$$

Neural Networks : Backward Phase



$$\delta_{ok} = (t_k - y_k)y_k(1 - y_k)$$

$$\delta_{hj} = a_j(1 - a_j)\sum_k w_{jk}\delta_{ok}$$

$$w_{jk} \leftarrow w_{jk} + \eta\delta_{ok}a_j$$

$$w_{ij} \leftarrow +\eta\delta_{hj}x_i$$

Deriving Backprop Again

- Remember

$$\frac{\partial}{\partial w_{ik}}(t_k - \sum_j w_{jk}x_j) = -x_i \quad \text{when i=j}$$

Only part of the sum that is function of w_{ik} is when $i = j$

Also Derivative of Activation Function

$$g(h) = \frac{1}{1+e^{-\beta h}}$$

$$\frac{dg}{dh} = \frac{d}{dh}\frac{1}{1+e^{-\beta h}}$$

$$= \beta g(h)(1 - g(h))$$

Backpropagation of Error

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial h_k}\frac{\partial h_k}{\partial w_{jk}}$$

$$\frac{\partial E}{\partial w_{jk}} = \left(\frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial h_k}\right)\frac{\partial h_k}{\partial w_{jk}}$$

$$\frac{\partial}{\partial y_k}\frac{1}{2}\sum_k (y_k - t_k)^2$$

$$\frac{\partial h_k}{\partial w_{jk}} = \frac{\partial \sum_l w_{lk} a_l}{\partial w_{jk}}$$

$$= \sum_l \frac{\partial w_{lk} a_l}{\partial w_{jk}}$$

$$\overbrace{(y_k - t_k) \qquad y_k(1 - y_k)}^{\pm_o} \qquad \mathrm{a}_j$$

$$\mathrm{w}_{jk} \leftarrow w_{jk} - \text{´}\,(y_k - t_k) y_k (1 - y_k) a_j$$

Backpropagation of Error

Error from output layer

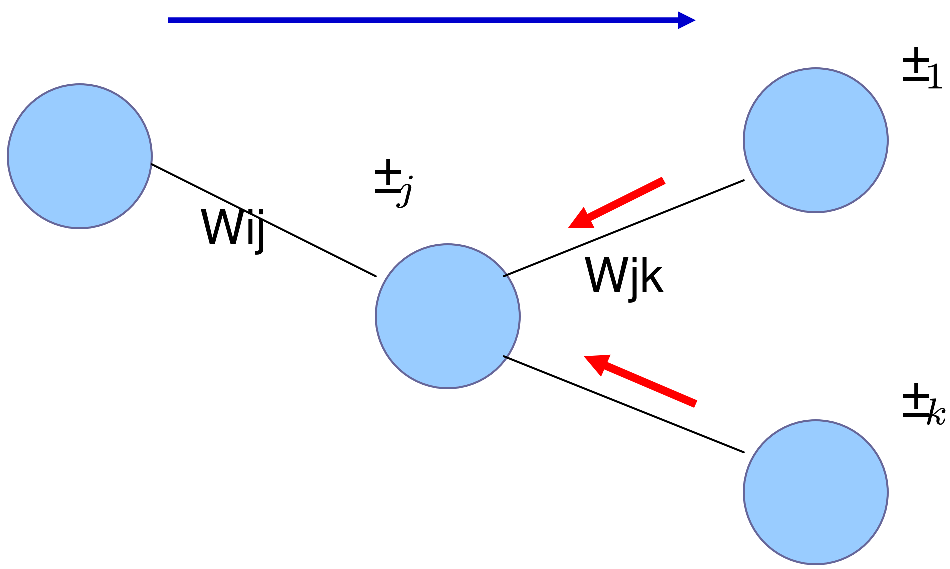
$$\frac{\partial E}{\partial w_{ij}} = a_j(1 - a_j)(\sum_k \delta_o w_{jk})x_i$$

$$\mathrm{w}_{ij} \leftarrow w_{ij} - \eta a_j(1 - a_j)(\sum_k \delta_o w_{jk})x_i$$

Update Equation for Hidden Layer

Error Backpropagation

- We are saying aj brings changes to error functions only through changes in ak



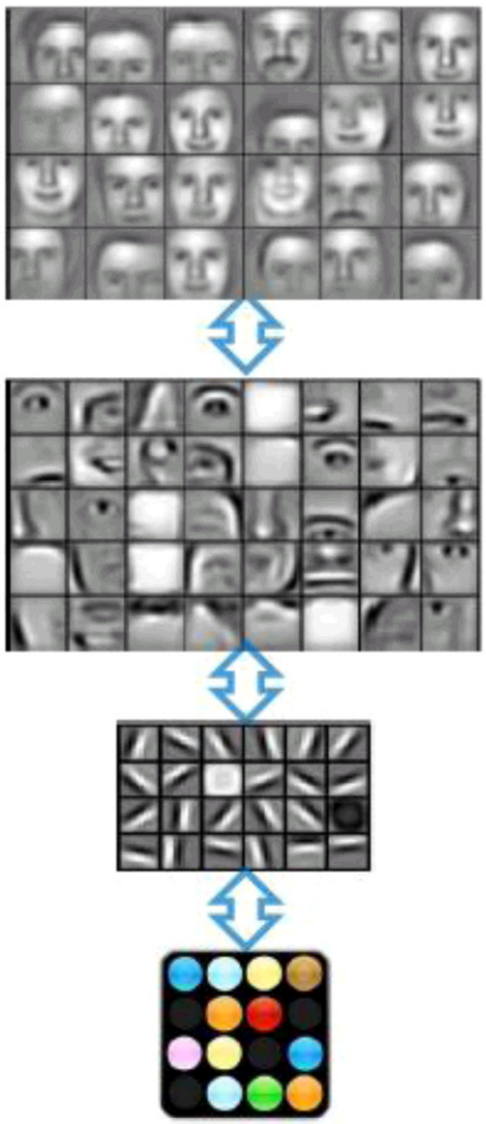
Backpropagation Problems

- Backpropagation does not scale well with many hidden layer
- Requires a lot of data
- Easily stuck in poor local minima

Related Work

- Deep Networks used in variety of NLP and Speech processing tasks
- [Colbert and Weston, 2008] Tagging, Chunking
  - Words into features
- [Mohamed et. al, 2009] ASR
  - Phone recognition
- [Maskey, et. al, 2012] MT
  - Speech to Speech Machine Translation
- [Dealaers et. al, 2007] Machine Transliteration

Deep Features from Image Research



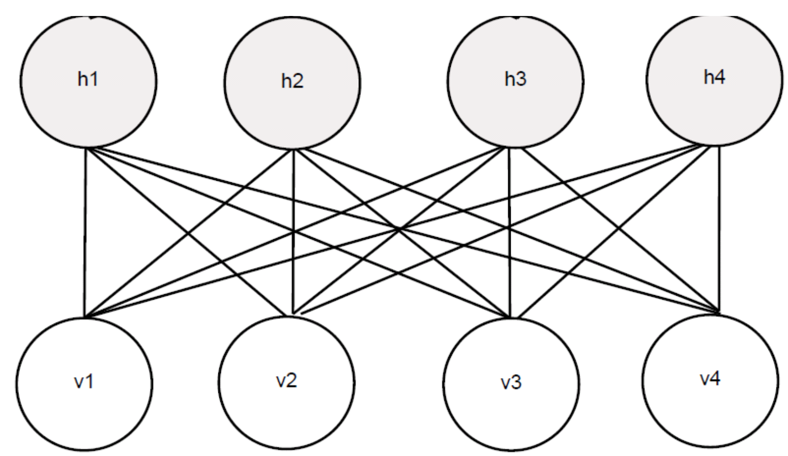
- Raw pixels (features) are passed through deep network
- Higher Level output in Deep Network seem to capture more abstract concepts
- Features at the top level almost look like faces

Can we see possibly extract such features for NLP?

Restricted Boltzmann Machines

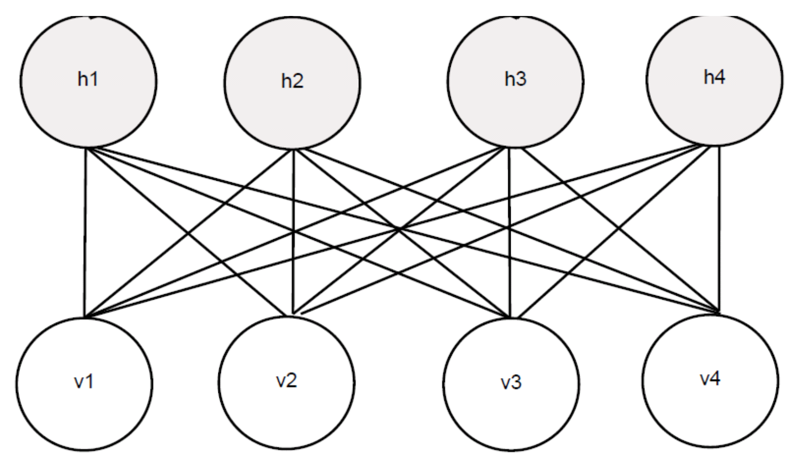
- Undirected graphical model with symmetric connection
- Bipartite graph
- Only two layers
  - Hidden
  - visible
- Connect binary stochastic neurons
- Restrict connections to make learning easy
- Hidden units conditionally independent give the visible units

If h1 is on or off is independent of activation of h2 … hn



Inference in RBM

- Inference is trivial
- Given a set of visible inputs we can easily tell the states of visible units



Energy of Joint Configuration

Binary state of visible node i

Binary state of hidden node j

$$E(v,h) = -\sum_{i,j} v_i h_j w_{ij}$$

Energy of Joint configuration

$$-\frac{\partial E(v,h)}{\partial w_{ij}} = v_i h_j$$

If we want to manipulate the energy to lower it all we need to weights

Energy to Probabilities

- If we give visible features and want to make the model say ‘make these features more probable’ we can manipulate the energy of the joint configuration
- Energy of joint configuration

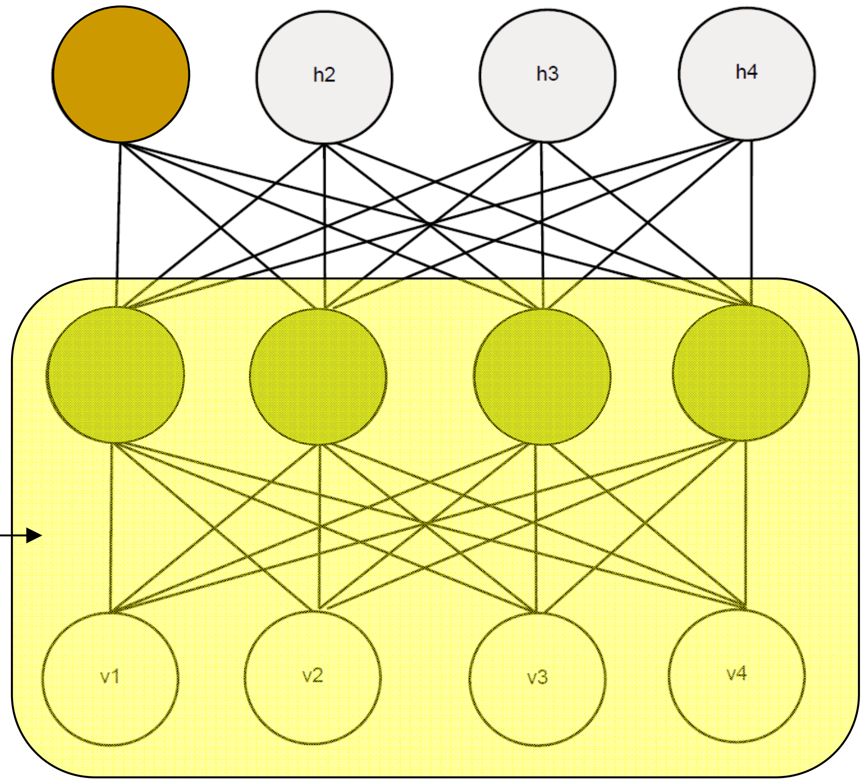
$$\mathrm{P}(\mathrm{v},\mathrm{h}) \propto e^{-E(v,h)}$$

- Manipulate probabilities by manipulating energy which can be manipulated with weights

Deep Network

- Key Parameters already sh
  - Weight matrix
  - Bias

One Layer Pair in our Deep Network is RBM

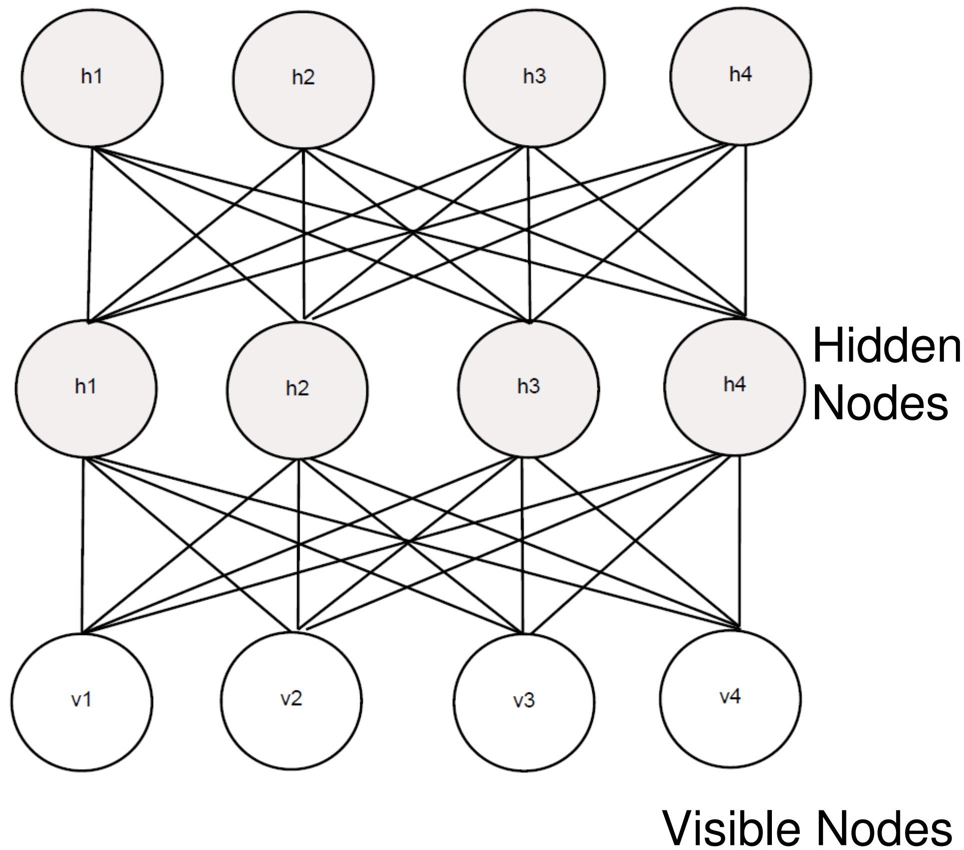


Deep Network with RBM

Deep Networks

$$p(\mathrm{v},\, \mathrm{h}^1, h^2, h^3, ..., h^l)$$

join distribution factored into conditionals across layers such as $p(\mathrm{h}^1|h^2)$



Conditional Distributions of Layers

- Conditionals are given by

$$p(h^k|h^{k+1}) = \prod_i p(h_i^k|h^{k+1})$$

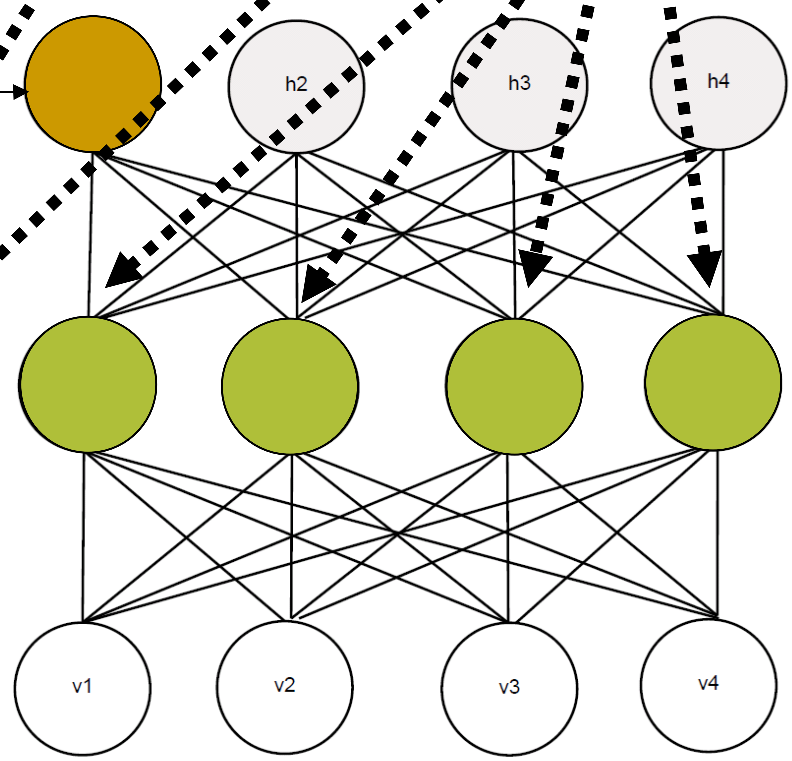
where

$$\mathrm{p}(\mathrm{h}_i^k|h^{k+1}) = sig(b_i^k + \sum_j W_{ij}^k h_j^{k+1})$$

Conditional Distribution per Node

$$p(h_i^k|h^{k+1}) = sig(b_i^k + \sum_j W_{ij}^k h_j^{k+1})$$

- This is basically saying



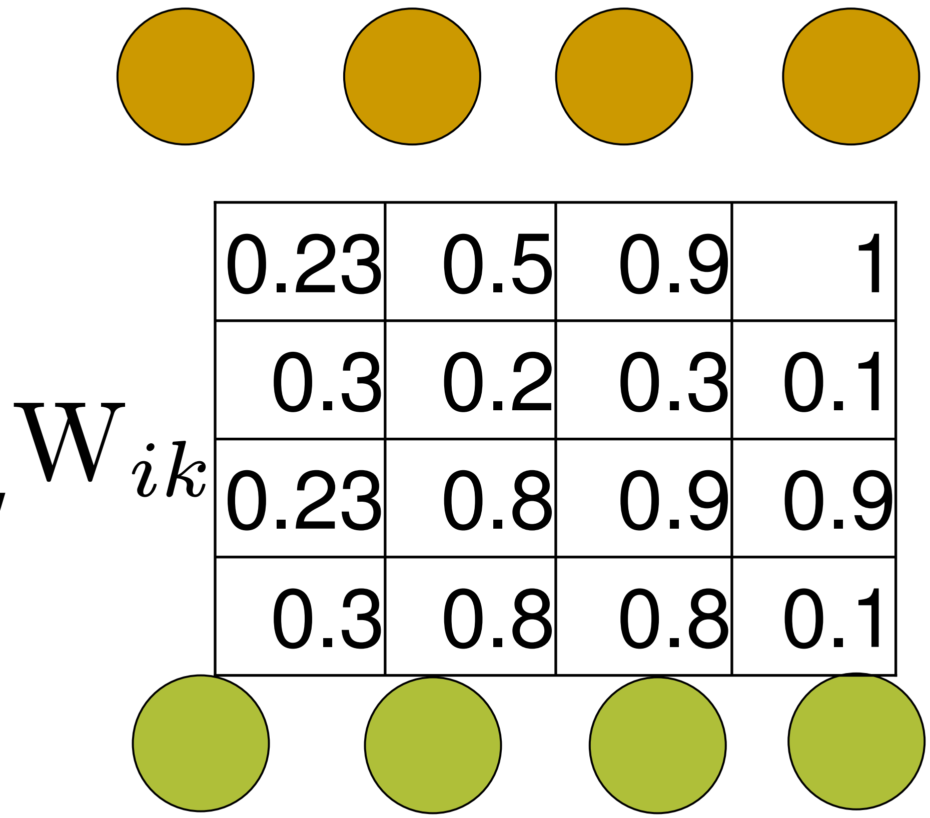
Sigmoid function

W_{ik}

Weight matrix if NXM size

Weight Matrix

- Number of weight matrices equals number of layers -1
- Linear combination of weights and passing it through sigmoid to give non-linearity to the output adds the power to this generative model
- Remember: we don't allow same layer connection



$$
W_{ik}
\begin{array}{|r|r|r|r|}
\hline
0.23 & 0.5 & 0.9 & 1 \\
\hline
0.3 & 0.2 & 0.3 & 0.1 \\
\hline
0.23 & 0.8 & 0.9 & 0.9 \\
\hline
0.3 & 0.8 & 0.8 & 0.1 \\
\hline
\end{array}
$$

Estimating the Weight Matrix

- Key difference between Neural Net and Deep Network comes about in the algorithm designed to estimate the weight matrix
- In order to train unsupervised deep network instead of using back propagation like neural network we use Hinton’s [Hinton, 2002] contrastive divergence algorithm to train each layer pair
- Each layer pair is a Restricted Boltzmann Machine

Deep Features

- Given a set of weight matrix of the Deep Network it lets you compute overall joint distribution based on factors of conditional distributions
- We can also ask $p(v|h)$ and $p(h|h+1)$ where v is visible layer and h is hidden layer
  - i.e. we are asking from the model to give us estimates from the input visible feature
  - Then we are asking it again to give us estimates from the hidden features from previous layer
  - Can think of as feature of feature of feature…

In Ideal World?

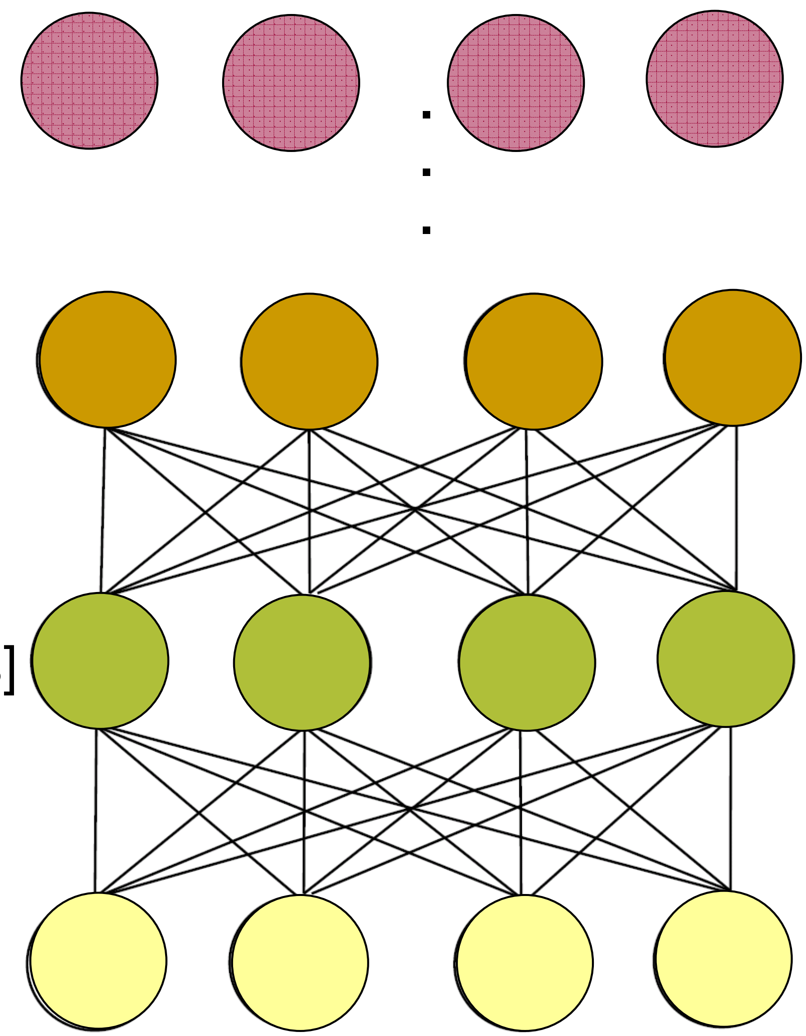
Unlike images it is hard to interpret what do the hidden layers of Deep Belief Net for NLP really mean?

[Hidden nodes = Topic IDs]

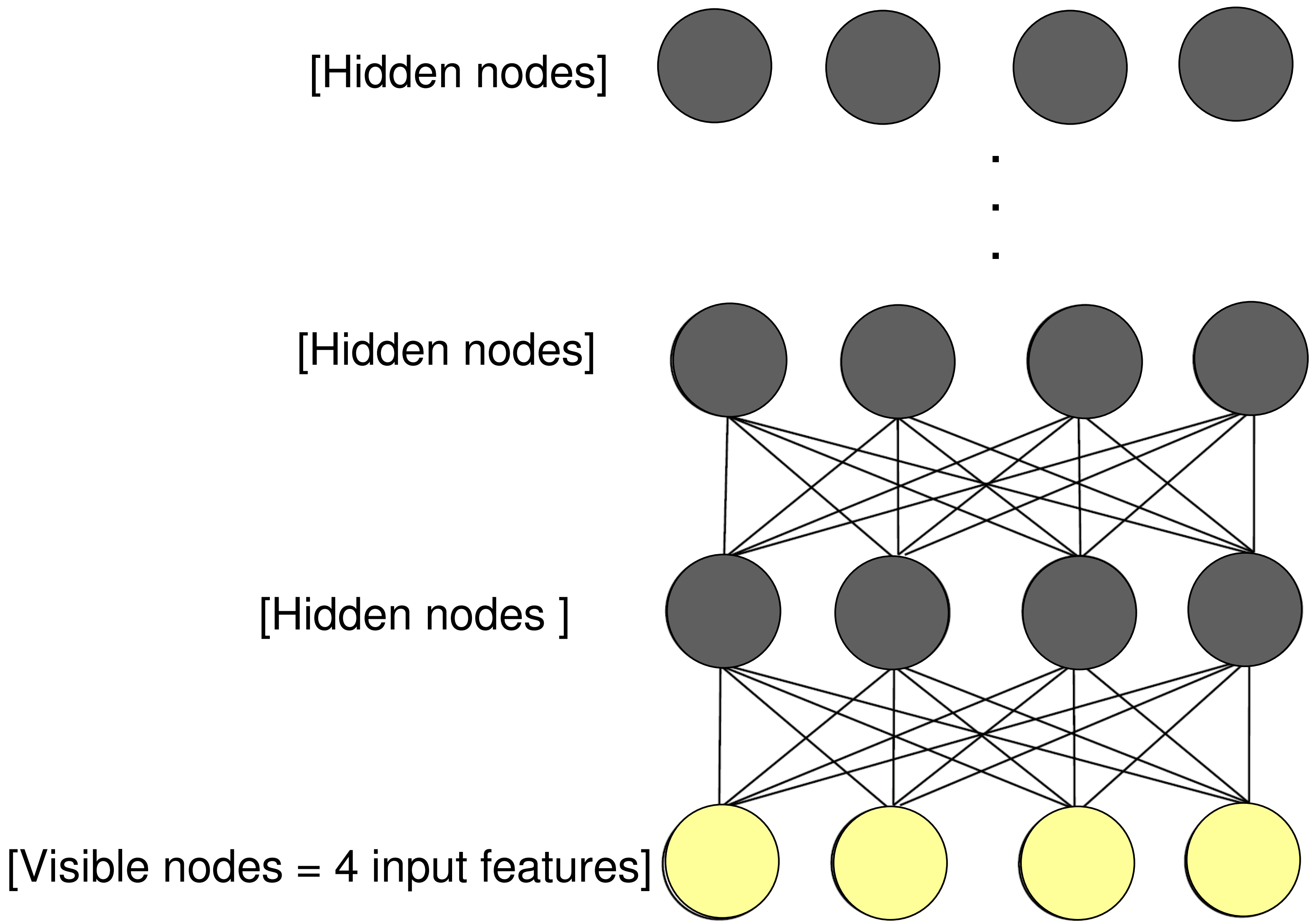
[Hidden nodes = Paragraphs]

[Hidden nodes = Sentences]

[Visible nodes = Words]



In Real World



Estimating the Weight Matrix

- Our hidden node update equation seem similar to feed forward network
- Key difference here is how we update the weights
- Derivative of log-likelihood for RBM given by

$$\frac{\partial logp(v,h)}{\partial \theta} = -\left\langle \frac{\partial logE(v,h)}{\partial \theta} \right\rangle_0 + \left\langle \frac{\partial logE(v,h)}{\partial \theta} \right\rangle_\infty$$

- where

$$\mathrm{E(v,h) = h'Wv + b'v + c'h}$$

- Instead of back propagation we use algorithm proposed by [Hinton, 2002] : contrastive divergence

Contrastive Divergence in Words

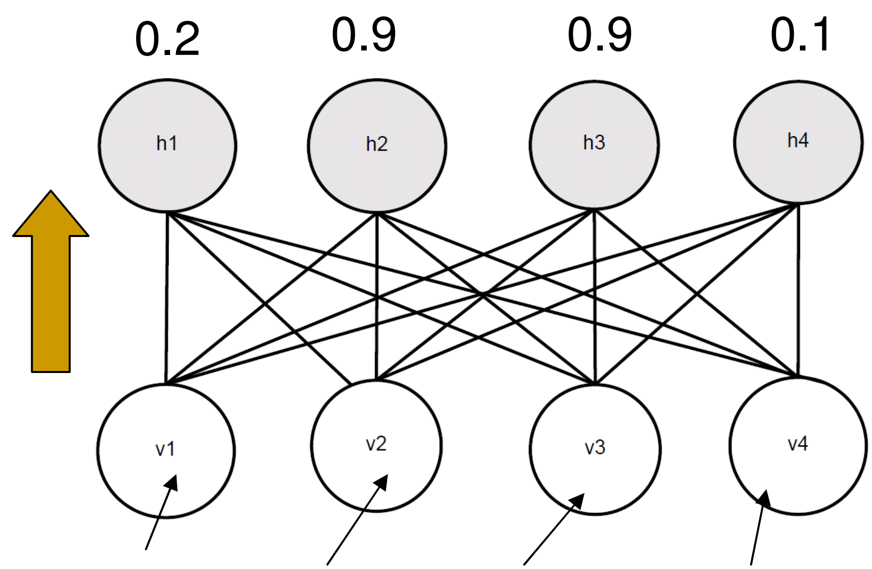
$$\mathrm{p}(\mathrm{h}_i = 1|v;\theta) = \sigma(\textstyle\sum_i W_{ij}v_j + c_i)$$ Hidden update: Eqn. H

$$\mathrm{p}(\mathrm{v}_j = 1|h;\theta) = \sigma(\textstyle\sum_i W_{ij}h_i + b_j)$$ Visible update: Eqn. V

- Loop
  - Find Activations A(hi) of Hidden Nodes using Eqn H
  - Using the activations A(hi) generate data A(vj) using Eqn V

Visible –Hidden Update for Learning

$$p(h_i = 1|v;\theta) = \sigma(\textstyle\sum_i W_{ij}v_j + c_i)$$ Hidden update: Eqn. H

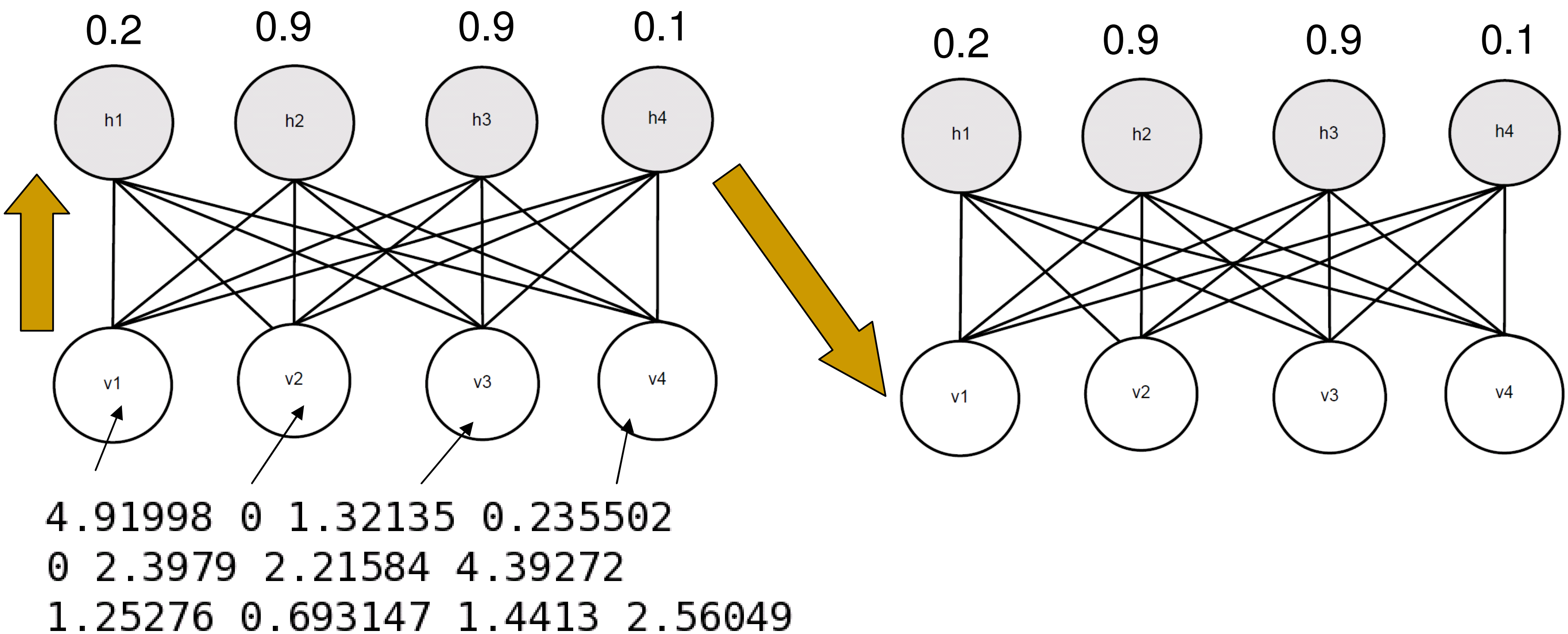


@1 آب @2 || @1 water the @2	4.91998 0 1.32135 0.235502
@1 آب اینکه || @1 water that	0 2.3979 2.21584 4.39272
@1 آب برای || @1 water to	1.25276 0.693147 1.4413 2.56049

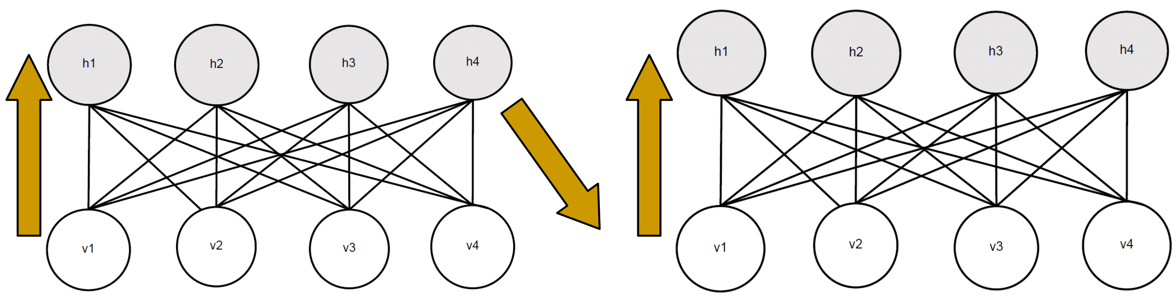
Training of Deep Network for MT

$$p(\mathrm{v}_j = 1|h;\theta) = \sigma(\textstyle\sum_i W_{ij}h_i + b_j)$$

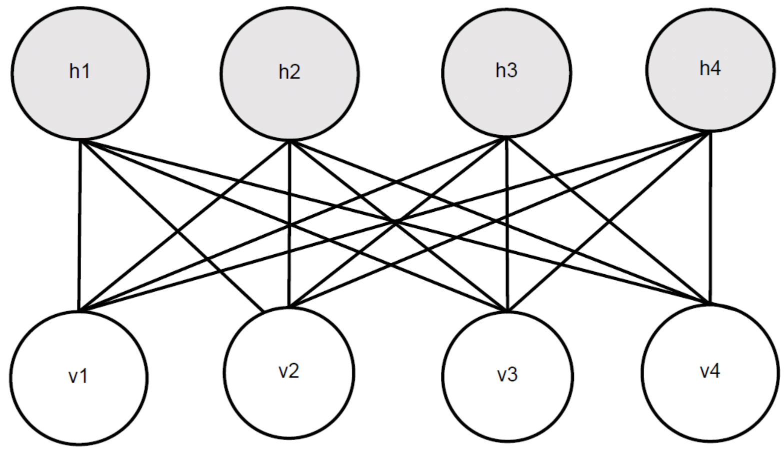
Visible update: Eqn. V



Training Deep Network for MT



Generating Features



$$\mathrm{W}_{ik}$$

Trained

```
4.91998 0 1.32135 0.235502
0 2.3979 2.21584 4.39272
1.25276 0.693147 1.4413 2.56049
```

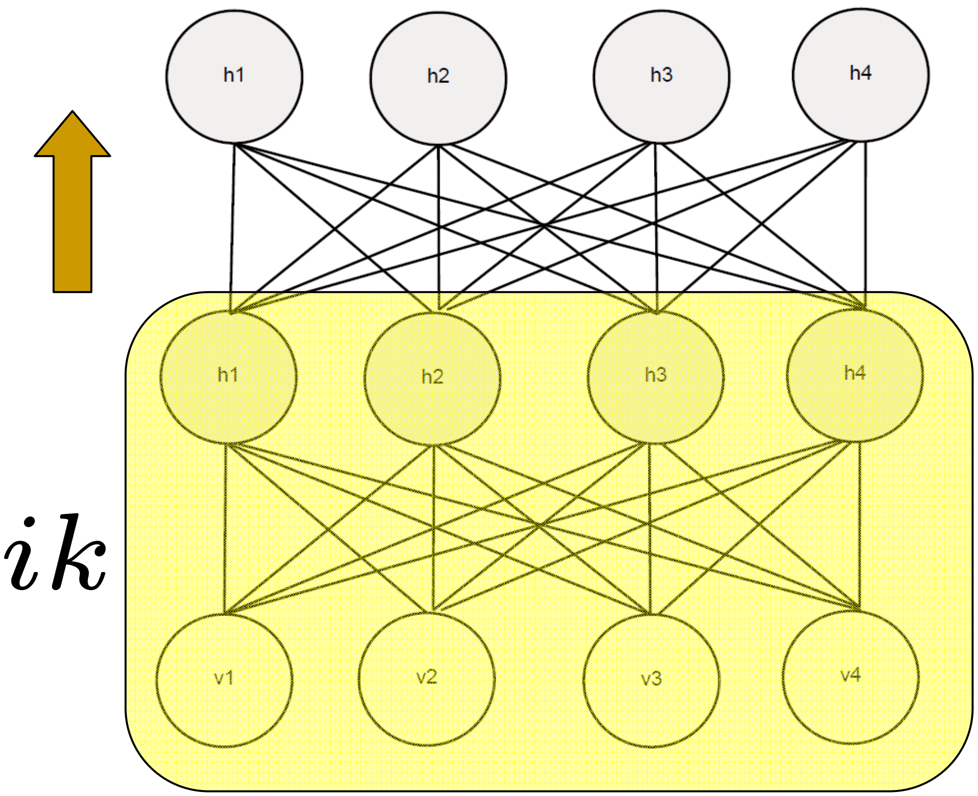
Once the weights are tuned we can simply generate features by using

$$\mathrm{p}(\mathrm{h}_i = 1|v;\theta) = \sigma(\textstyle\sum_i W_{ij} v_j + c_i)$$

Greedy Layerwise Training

- Deep Network with stacks of RBM can be trained efficiently as done with single RBMS
- Use the same contrastive divergence algorithm to train first layer
- Fix the weights and generate features for 1st layer
- Use the generated features as input to the 2nd layer
- Repeat this procedure until the weights for all layers are trained

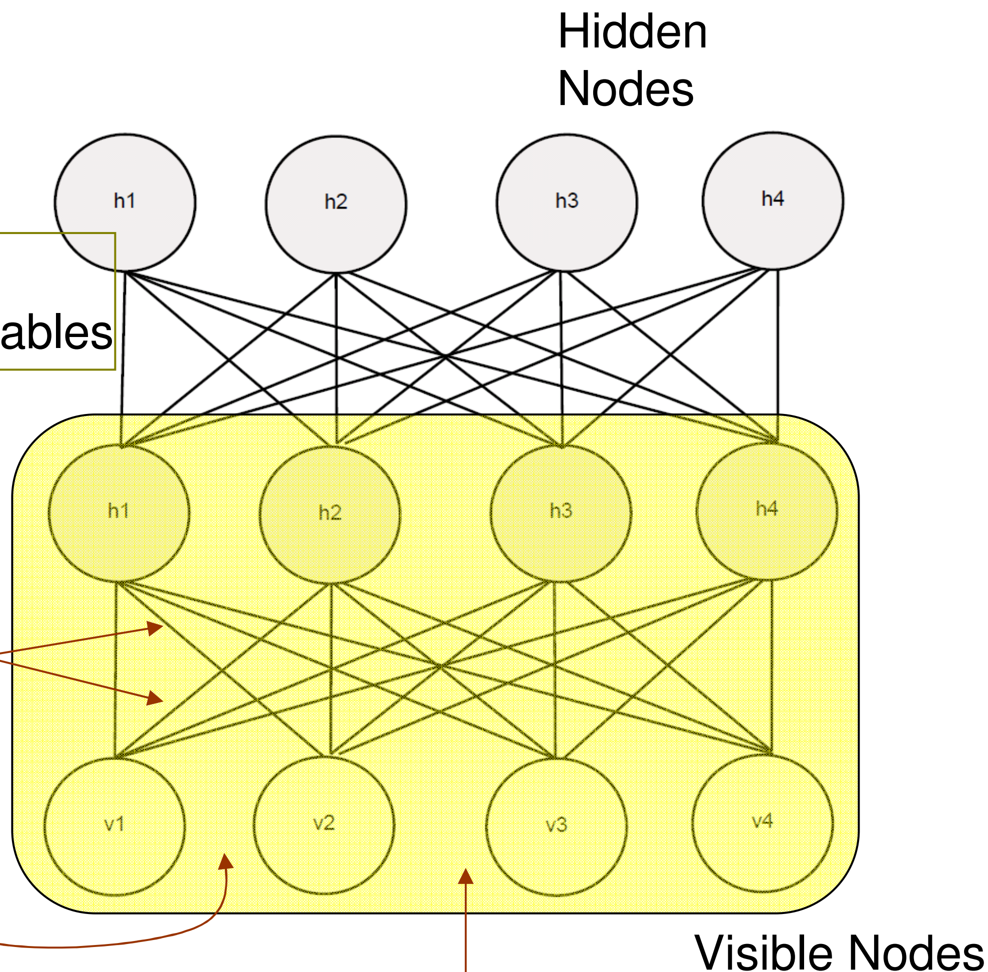
Greedy Layerwise Training



$$W_{ik}$$

Once weights are trained features generated by propagating through the network

Deep Belief Networks



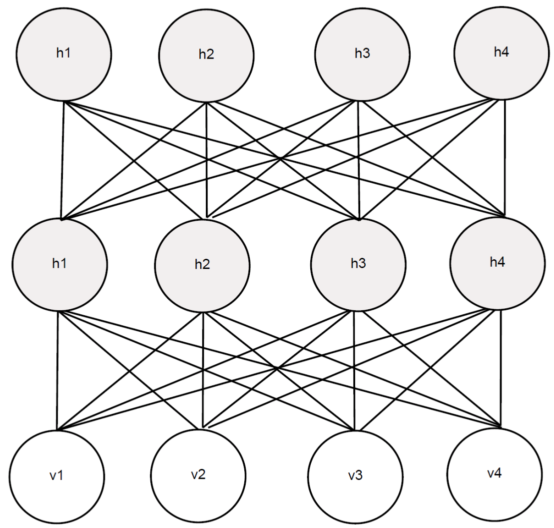
Generative Model composed with multiple layers of stochastic hidden variables

Undirected symmetric connections

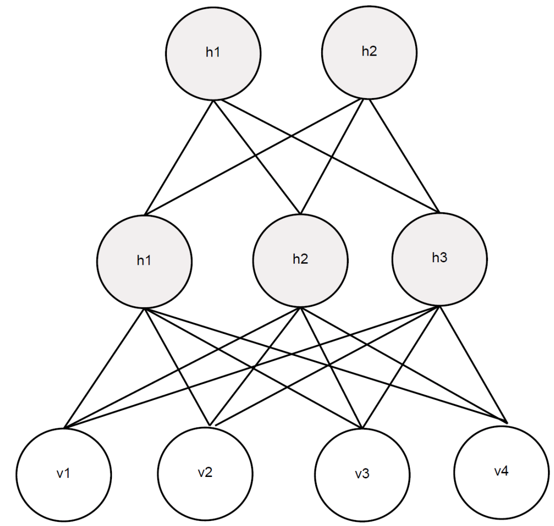
No connections within the same layer nodes

Efficient inference, layer by layer training procedure

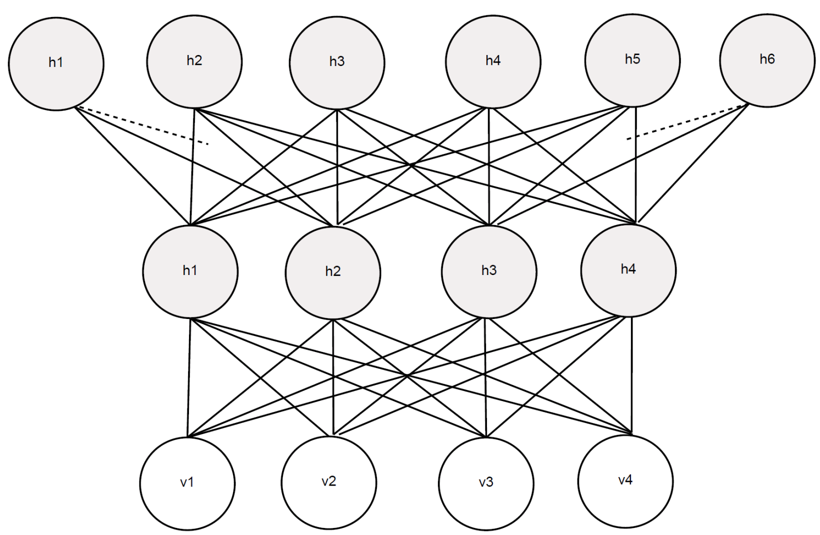
Restricted Boltzmann Machines (can use other auto encoders)


h1
h2
h3
h4
h1
h2
h3
h4
v1
v2
v3
v4

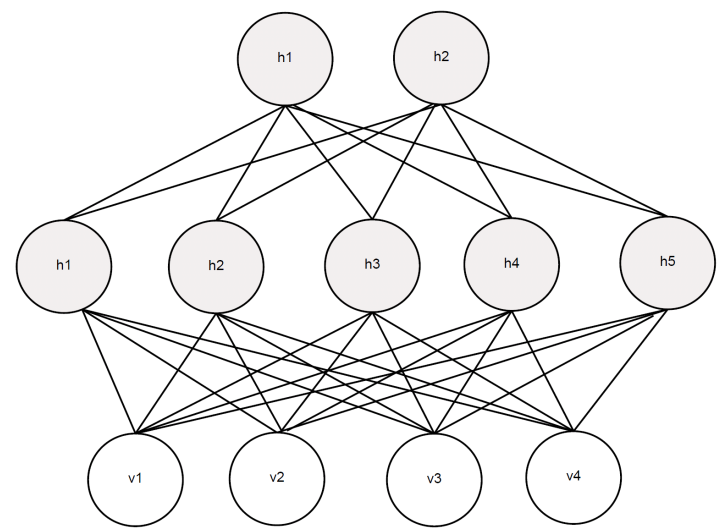
(R)


h1
h2
h1
h2
h3
v1
v2
v3
v4

(P)


h1
h2
h3
h4
h5
h6
h1
h2
h3
h4
v1
v2
v3
v4

(InvP)


h1
h2
h1
h2
h3
h4
h5
v1
v2
v3
v4

(D)

Reference

- Hinton et. al, “A Fast Learning Algorithm for Deep Belief Nets” 2006