Learning Hierarchical Generative Models

Ruslan Salakhutdinov

Department of Statistics and Computer Science
University of Toronto

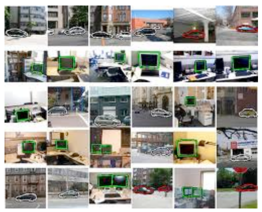
Machine Learning’s Successes

- Computer Vision:
  - Image inpainting/denoising, segmentation
  - object recognition/detection, scene understanding
- Information Retrieval / NLP:
  - Text, audio, and image retrieval
  - Parsing, machine translation, text analysis
- Speech processing:
  - Speech recognition, voice identification
- Robotics:
  - Autonomous car driving, planning, control
- Computational Biology
- Cognitive Science.

Mining for Structure

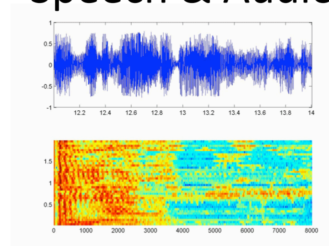
Massive increase in both computational power and the amount of data available from web, video cameras, laboratory measurements.

Images & Video

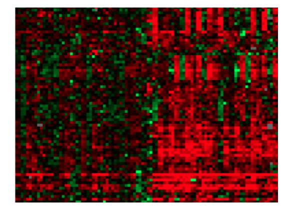


Text & Language

Speech & Audio

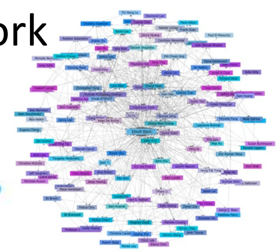


Gene Expression

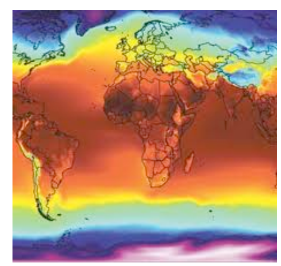


Product Recommendation

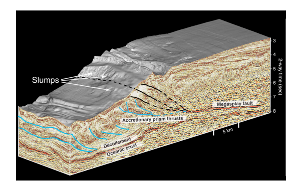
Relational Data/ Social Network



Climate Change



Geological Data



Mostly Unlabeled

- Develop statistical models that can discover underlying structure, cause, or statistical correlation from data in **unsupervised** or **semi-supervised** way.
- Multiple application domains.

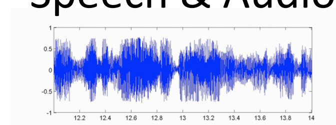
Mining for Structure

Massive increase in both computational power and the amount of data available from web, video cameras, laboratory measurements.

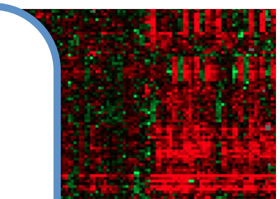
Images & Video

Text & Language

Speech & Audio



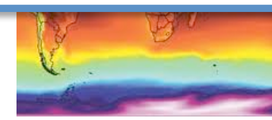
Gene Expression

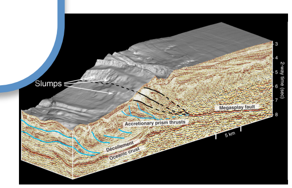


Prod

Reco

logical Data





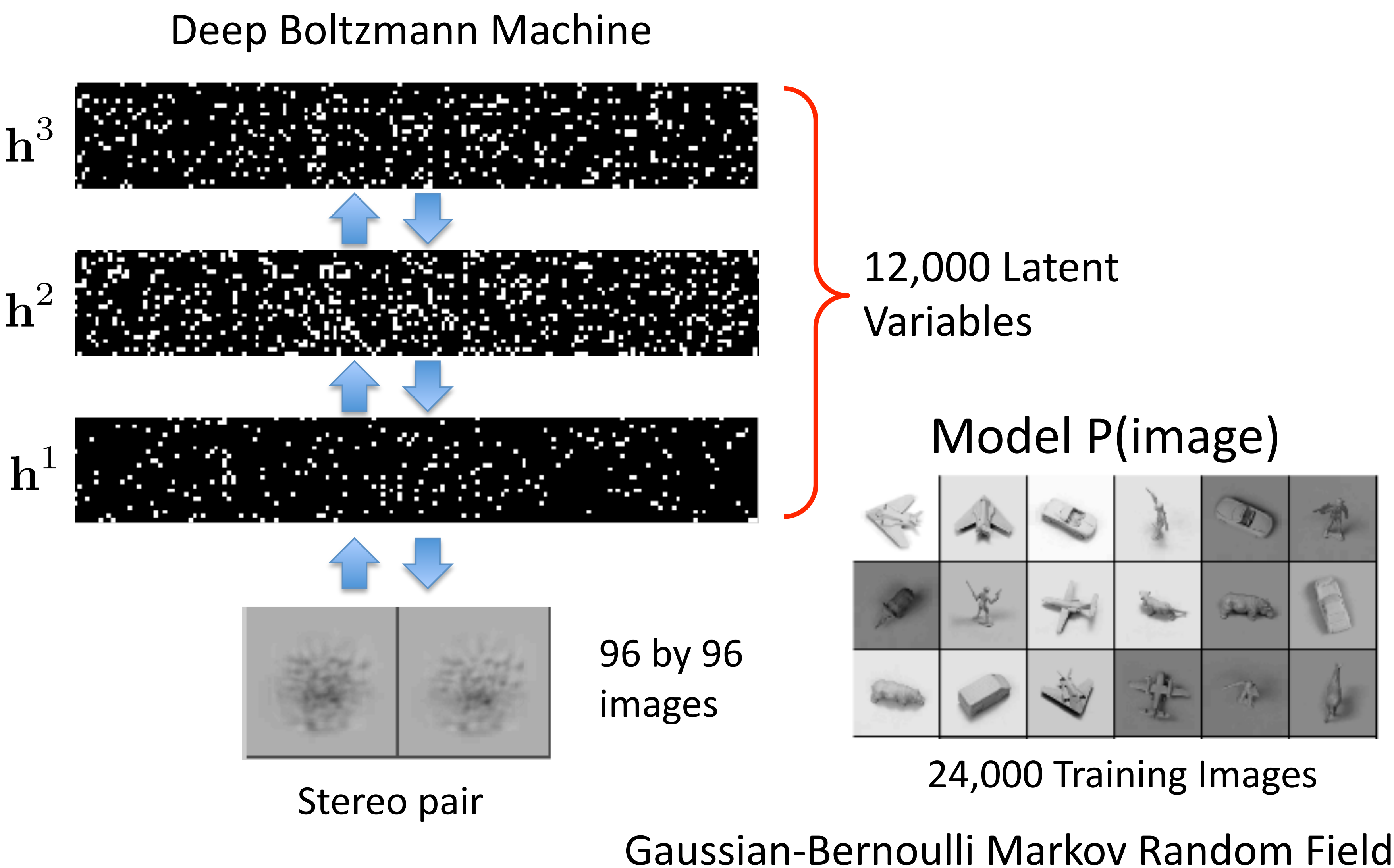
Deep Generative Models that support inferences and discover structure at multiple levels.

Mostly Unlabeled

- Develop statistical models that can discover underlying structure, cause, or statistical correlation from data in **unsupervised** or **semi-supervised** way.
- Multiple application domains.

Deep Generative Model

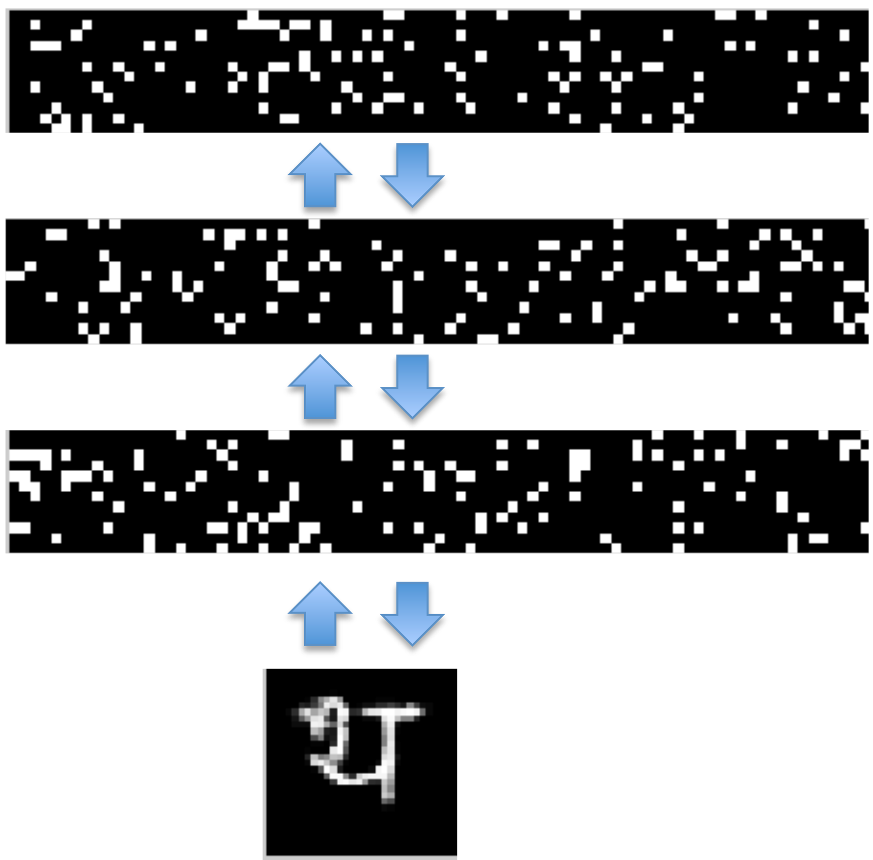
(Salakhutdinov, 2008; Salakhutdinov & Hinton, AI & Statistics 2009)



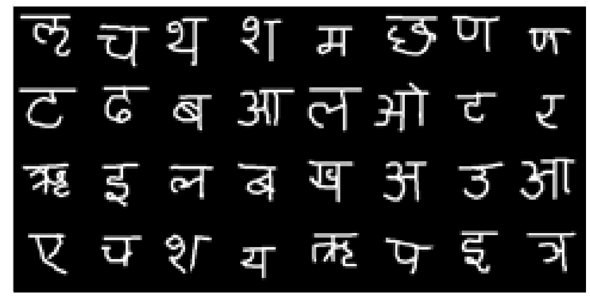
Deep Generative Model

(Salakhutdinov, 2008; Salakhutdinov & Hinton, AI & Statistics 2009)

Sanskrit



Model P(image)



25,000 characters from 50 alphabets around the world.

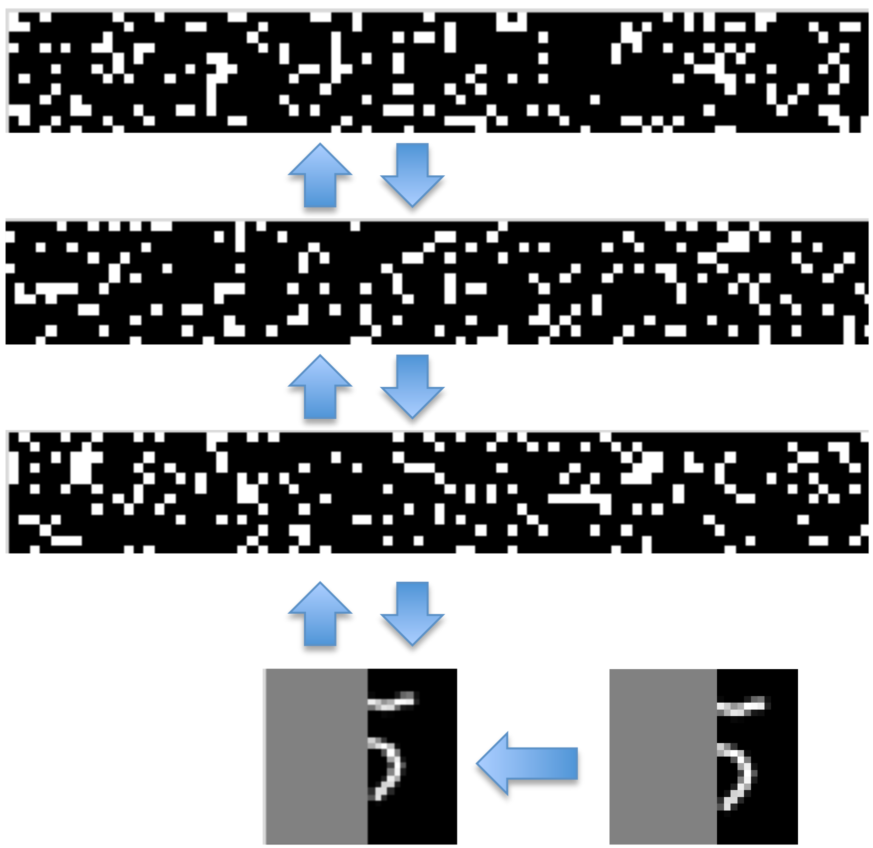
- 3,000 hidden variables
- 784 observed variables (28 by 28 images)
- Over 2 million parameters

Bernoulli Markov Random Field

Deep Generative Model

(Salakhutdinov, 2008; Salakhutdinov & Hinton, AI & Statistics 2009)

Conditional Simulation

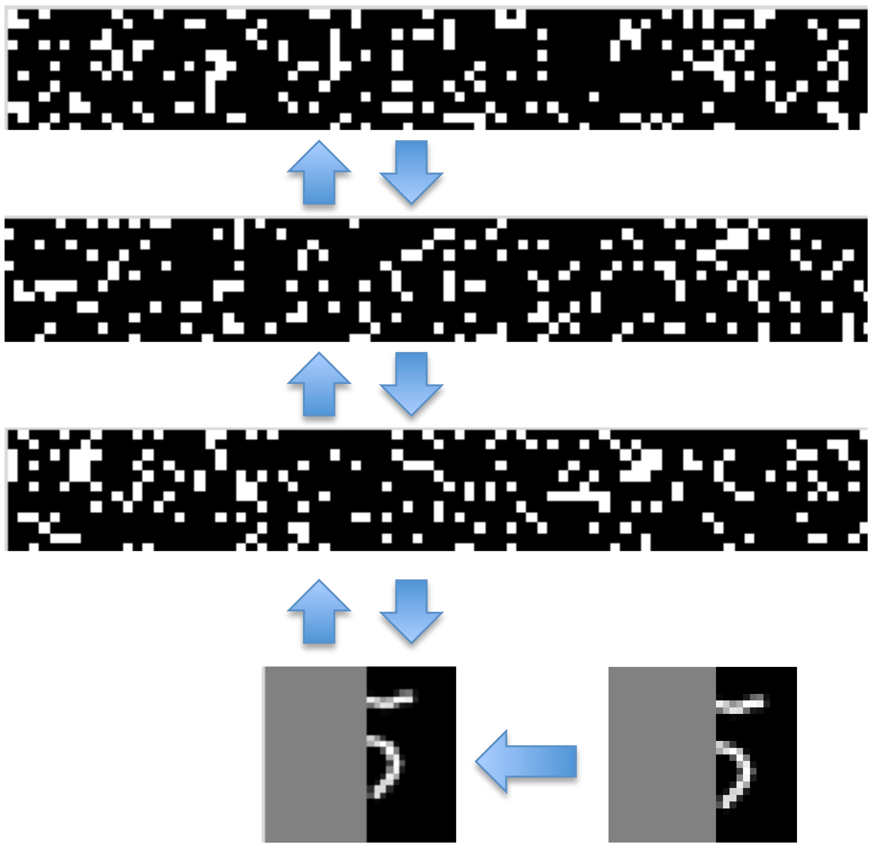


P(image|partial image)

Bernoulli Markov Random Field

Deep Generative Model

(Salakhutdinov, 2008; Salakhutdinov & Hinton, AI & Statistics 2009)



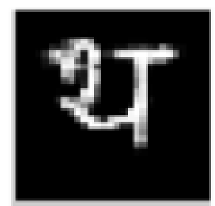
Conditional Simulation

P(image|partial image)

Why so difficult?

28

28



$2^{28 \times 28}$ possible images!

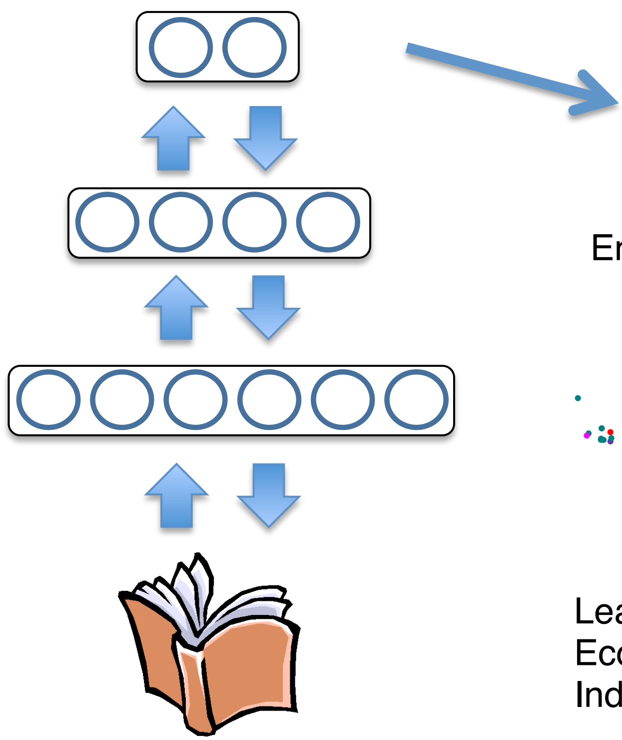
Bernoulli Markov Random Field

Deep Generative Model

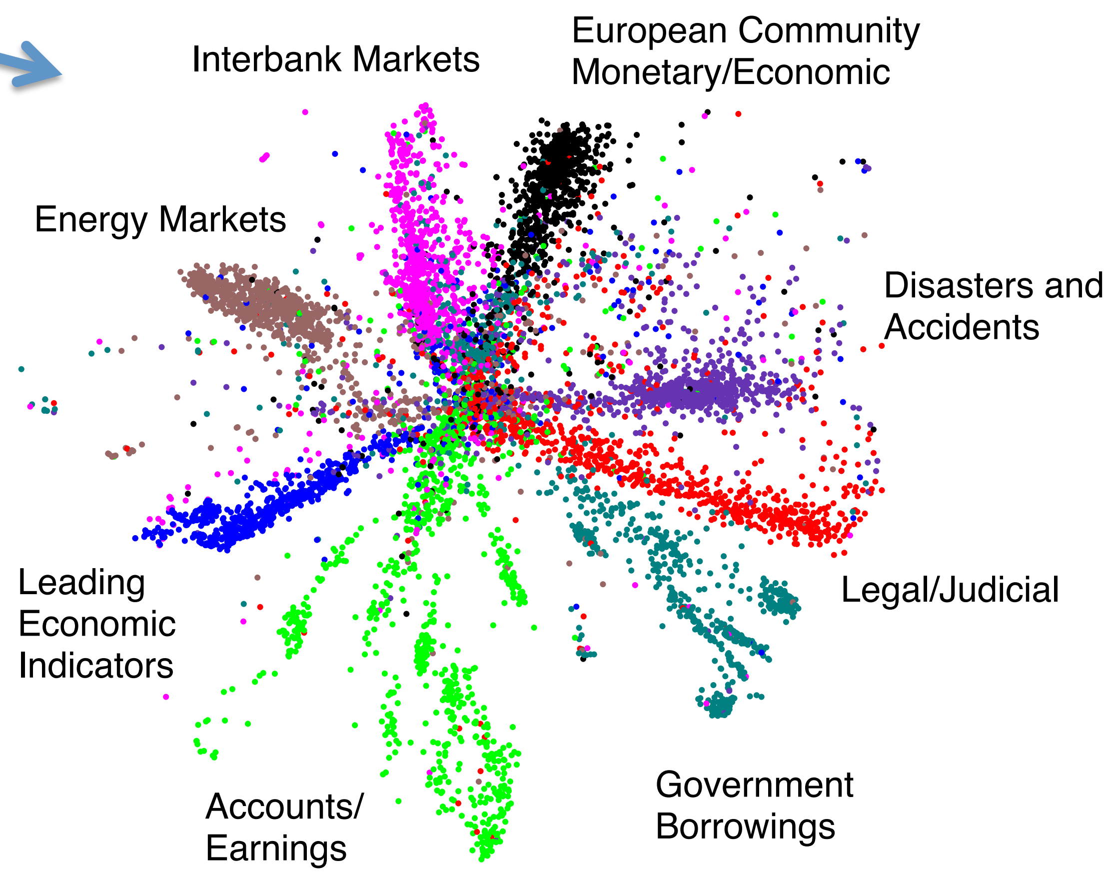
(Hinton & Salakhutdinov, Science 2006)

Model P(document)

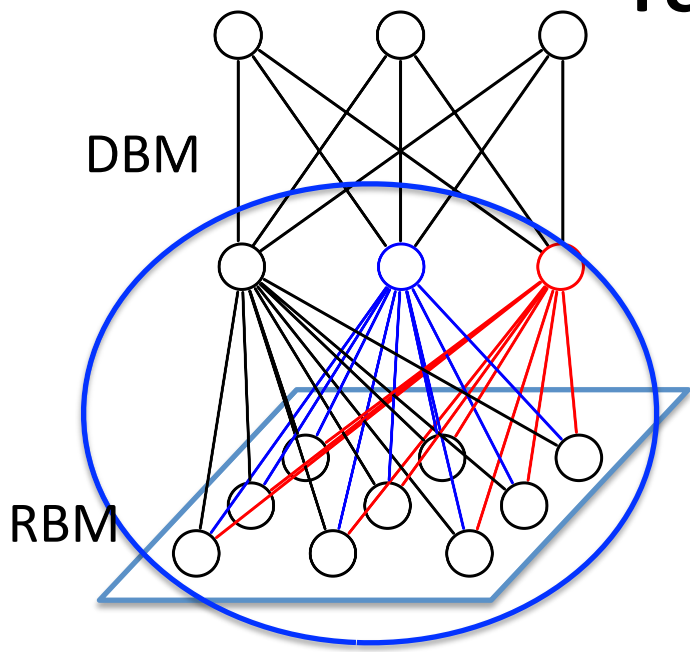
Reuters dataset: 804,414 newswire stories: **unsupervised**

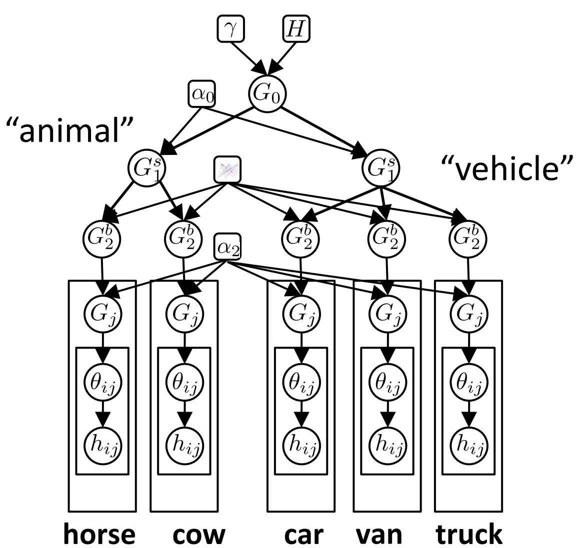


Bag of words



Talk Roadmap





Part 1: Deep Networks

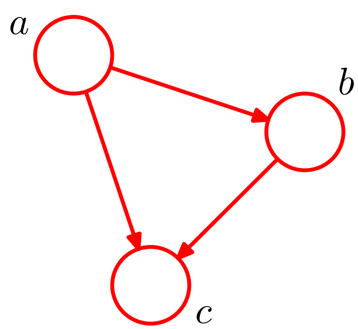
- Introduction: Graphical Models.
- Restricted Boltzmann Machines: Learning low-level features.
- Deep Belief Networks: Learning Part-based Hierarchies.
- Deep Boltzmann Machines.

Part 2: Advanced Hierarchical Models

- Learning Category Hierarchies.
- Transfer Learning / One-Shot Learning.

Graphical Models

Graphical Models: Powerful framework for representing dependency structure between random variables.



• The joint probability distribution over a set of random variables.

• The graph contains a set of nodes (vertices) that represent random variables, and a set of links (edges) that represent dependencies between those random variables.

• The joint distribution over all random variables decomposes into a **product of factors**, where each factor depends on a subset of the variables.

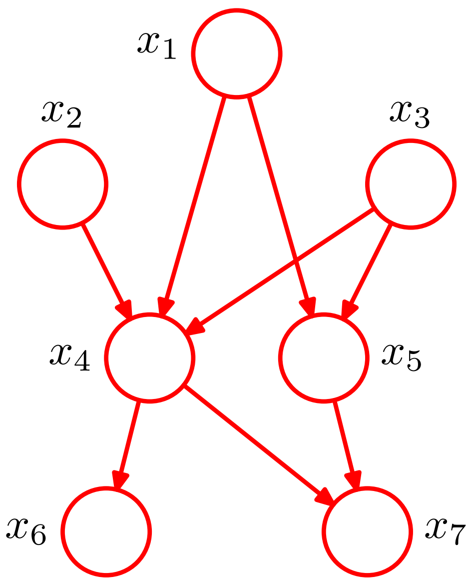
Two type of graphical models:

- **Directed** (Bayesian networks)
- **Undirected** (Markov random fields, Boltzmann machines)

Hybrid graphical models that combine directed and undirected models, such as Deep Belief Networks, Hierarchical-Deep Models.

Directed Graphical Models

Directed graphs are useful for expressing causal relationships between random variables.



- The joint distribution defined by the graph is given by the **product of a conditional distribution for each node conditioned on its parents.**

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k|\mathrm{pa}_k)$$

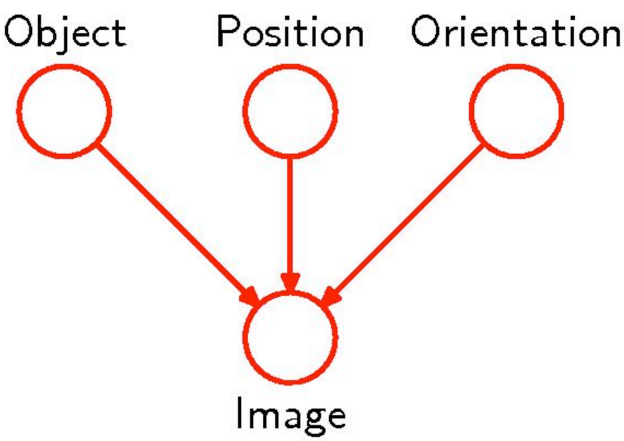
- For example, the joint distribution over x1,..,x7 factorizes:

$$p(\mathbf{x}) = p(x_1)p(x_2)p(x_3)p(x_4|x_1,x_2,x_3)p(x_5|x_1,x_3)p(x_6|x_4)p(x_7|x_4,x_5)$$

Directed acyclic graphs, or *DAGs*.

Directed Graphical Models

Example: Generative model of an image:



- Object identity (discrete variable) and the position and orientation (continuous variables) have **independent prior probabilities**.

- The image has a probability distribution that depends on the object identity, position, and orientation **(likelihood function)**.

The joint distribution:

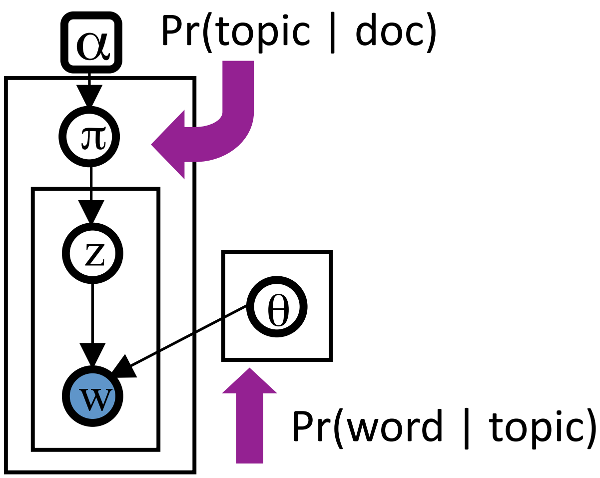
$$P(Im, Ob, Po, Or) = \underbrace{P(Im|Ob, Po, Or)}_{\text{Likelihood}} \underbrace{P(Ob)P(Po)P(Or)}_{\text{Prior}}$$

Inference: Computing posterior:

$$P(Ob, Po, Or|Im) = \frac{1}{\underbrace{P(Im)}_{\text{Marginal likelihood: Often difficult to compute}}} P(Im|Ob, Po, Or)P(Ob)P(Po)P(Or)$$

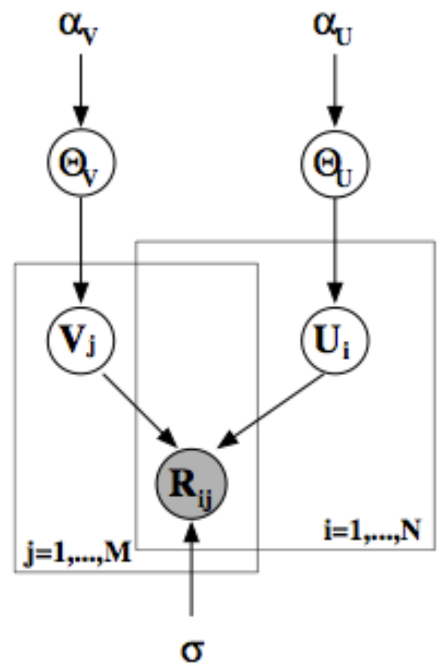
Popular Models

Latent Dirichlet Allocation



• One of the popular models for modeling word count vectors. We will see this model later.

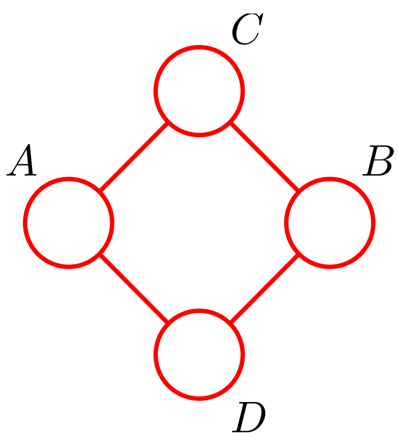
Probabilistic Matrix Factorization



• One of the popular models for collaborative filtering applications. Part of the winning solution in the Netflix contest.

Undirected Graphical Models

Directed graphs are useful for expressing causal relationships between random variables, whereas undirected graphs are useful for expression soft constraints between random variables



- The joint distribution defined by the graph is given by the **product of non-negative potential functions** over the maximal cliques (connected subset of nodes).

$$p(\mathbf{x}) = \frac{1}{\mathcal{Z}} \prod_C \phi_C(x_C) \qquad \mathcal{Z} = \sum_{\mathbf{x}} \prod_C \phi_C(x_C)$$

where the normalizing constant $\mathcal{Z}$ is called a partition function.

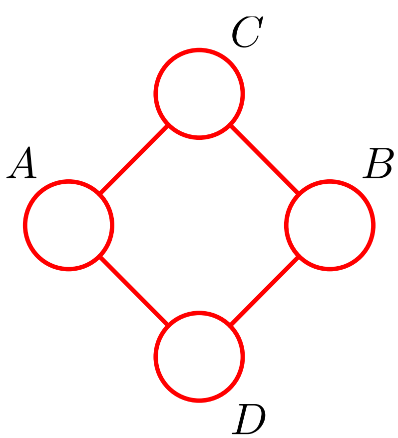
- For example, the joint distribution factorizes:

$$p(A,B,C,D) = \frac{1}{\mathcal{Z}} \phi(A,C)\phi(C,B)\phi(B,D)\phi(A,D)$$

Often called **pairwise Markov random field**, as it factorizes over pairs of random variables.

Markov random fields, Boltzmann machines.

Markov Random Fields



$$p(\mathbf{x}) = \frac{1}{\mathcal{Z}} \prod_C \phi_C(x_C)$$

- Each potential function is a mapping from joint configurations of random variables in a clique to non-negative real numbers.

- The choice of potential functions is not restricted to having specific probabilistic interpretations.

Potential functions are often represented as exponentials:

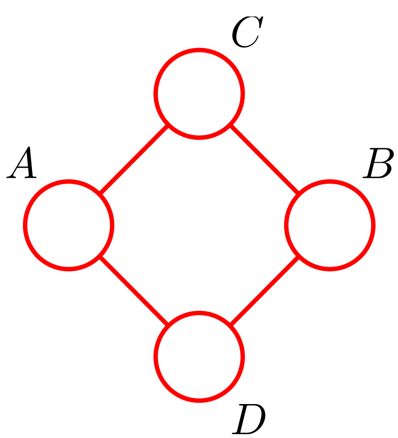
$$p(\mathbf{x}) = \frac{1}{\mathcal{Z}} \prod_C \phi_C(x_C) = \frac{1}{\mathcal{Z}} \exp\Big(-\sum_C E(x_c)\Big) = \underbrace{\frac{1}{\mathcal{Z}} \exp(-E(\mathbf{x}))}_{\text{Boltzmann distribution}}$$

where E(x) is called an energy function.

- Suppose x is a binary random vector with $x_i \in \{+1, -1\}$.
- If x is 100-dimensional, we need to sum over 2^{100} terms!

Computing Z is often very hard. This represents a major limitation of undirected models.

Markov Random Fields



$$p(\mathbf{x}) = \frac{1}{\mathcal{Z}} \prod_C \phi_C(x_C)$$

- Each potential function is a mapping from joint configurations of random variables in a clique to non-negative real numbers.

- The choice of potential functions is not restricted to having specific probabilistic interpretations.

Potential functions are often represented as exponentials:

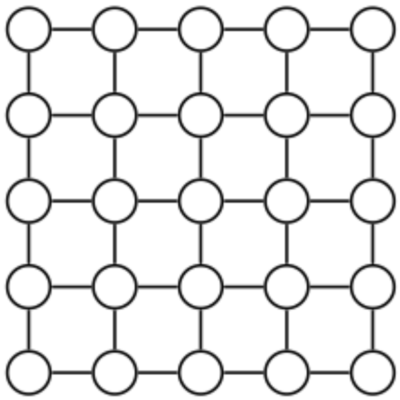
$$p(\mathbf{x}) = \frac{1}{\mathcal{Z}} \prod_C \phi_C(x_C) = \frac{1}{\mathcal{Z}} \exp(-\sum_C E(x_c)) = \underbrace{\frac{1}{\mathcal{Z}} \exp(-E(\mathbf{x}))}_{\text{Boltzmann distribution}}$$

where E(x) is called an energy function.

Compare to computing posterior:

$$P(\theta|\mathcal{D}) = \frac{1}{P(\mathcal{D})} P(\mathcal{D}|\theta) P(\theta) \quad \text{where} \quad P(\mathcal{D}) = \int P(\mathcal{D}, \theta) d\theta$$

Maximum Likelihood Learning



Consider binary pairwise MRF:

$$P_\theta(\mathbf{x}) = \frac{1}{\mathcal{Z}(\theta)} \exp\Big(\sum_{ij \in E} x_i x_j \theta_{ij} + \sum_{i \in V} x_i \theta_i \Big)$$

Given a set of *i.i.d.* training examples $\mathcal{D} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(N)}\}$, we want to learn model parameters θ.

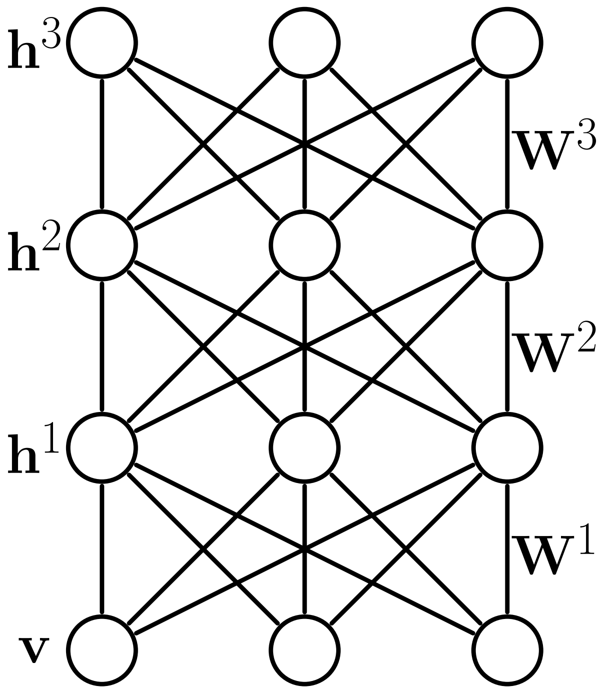
Maximize log-likelihood objective: $L(\theta) = \frac{1}{N} \sum_{n=1}^{N} \log P_\theta(\mathbf{x}^{(n)})$

Derivative of the log-likelihood:

$$\frac{\partial L(\theta)}{\partial \theta_{ij}} = \frac{1}{N} \sum_n [x_i^{(n)} x_j^{(n)}] - \underbrace{\sum_{\mathbf{x}} [x_i x_j P_\theta(\mathbf{x})]}_{\text{Difficult to compute: exponentially many configurations}} = \mathrm{E}_{P_{data}}[x_i x_j] - \mathrm{E}_{P_\theta}[x_i x_j]$$

MRFs with Latent Variables

For many interesting real-world problems, we need to introduce hidden or latent variables.



- Our random variables will contain both visible and hidden variables x=(v,h).

- Probability of observed input is given by marginalizing out the states of hidden variables:

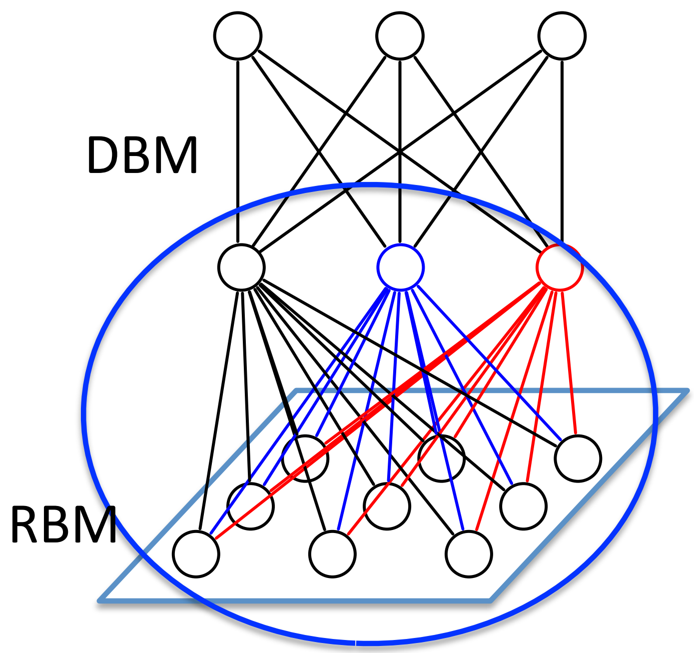
$$p(\mathbf{v}) = \frac{1}{\mathcal{Z}} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))$$

- In general computing both partition function and summation over hiddens will be intractable, except for special cases.

- Parameter learning becomes a very challenging task.

Deep Networks have to deal with this intractability.

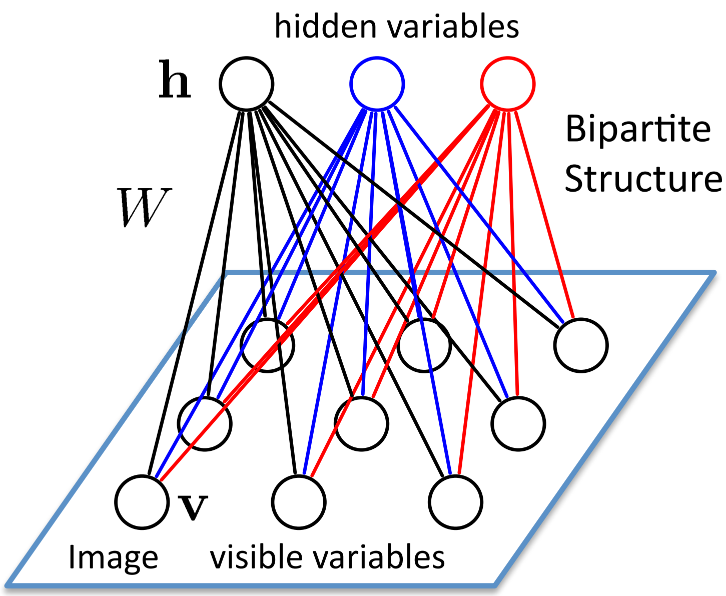
Talk Roadmap



Part 1: Deep Networks

- Introduction: Graphical Models.
- Restricted Boltzmann Machines: Learning low-level features.
- Deep Belief Networks: Learning Part-based Hierarchies.
- Deep Boltzmann Machines.

Restricted Boltzmann Machines



Stochastic binary visible variables $\mathbf{v} \in \{0,1\}^D$ are connected to stochastic binary hidden variables $\mathbf{h} \in \{0,1\}^F$.

The energy of the joint configuration:

$$E(\mathbf{v},\mathbf{h};\theta) = -\sum_{ij} W_{ij} v_i h_j - \sum_i b_i v_i - \sum_j a_j h_j$$

$\theta = \{W, a, b\}$ model parameters.

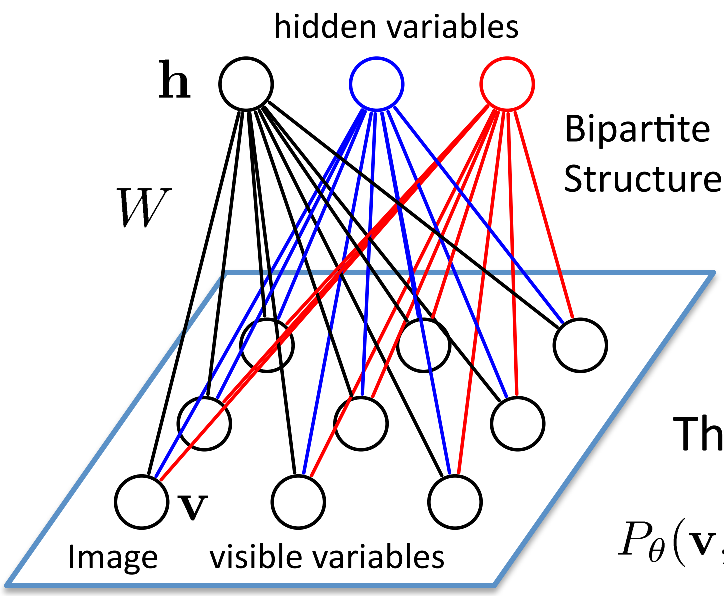
Probability of the joint configuration is given by the Boltzmann distribution:

$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\big(-E(\mathbf{v},\mathbf{h};\theta)\big) = \underbrace{\frac{1}{\mathcal{Z}(\theta)}}_{\text{partition function}} \prod_{ij} \underbrace{e^{W_{ij} v_i h_j}}_{\text{potential functions}} \prod_i e^{b_i v_i} \prod_j e^{a_j h_j}$$

$$\mathcal{Z}(\theta) = \sum_{\mathbf{h},\mathbf{v}} \exp\big(-E(\mathbf{v},\mathbf{h};\theta)\big)$$

Markov random fields, Boltzmann machines, log-linear models.

Restricted Boltzmann Machines



Product of Experts formulation.

The joint distribution is given by:

$$P_\theta(\mathbf{v}, \mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\Big(\sum_{ij} W_{ij} v_i h_j + \sum_i b_i v_i + \sum_j a_j h_j\Big)$$

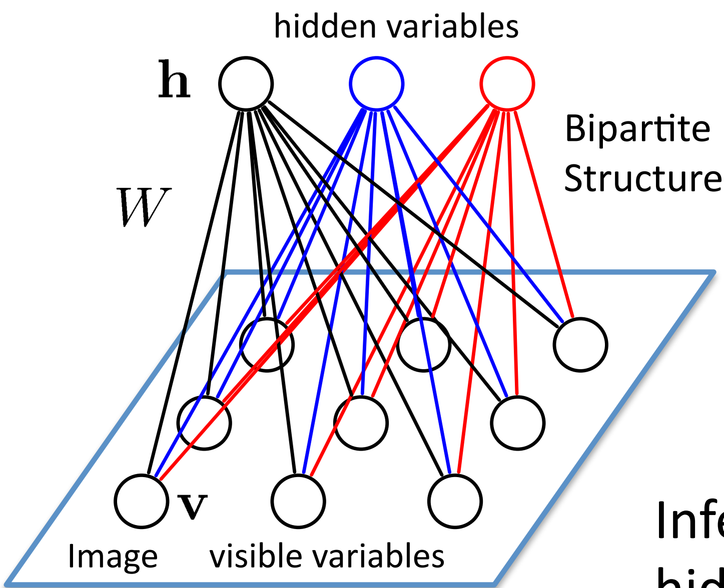
where the undirected edges in the graphical model represent $\{W_{ij}\}$.

Marginalizing over the states of hidden variables:

$$P_\theta(\mathbf{v}) = \sum_{\mathbf{h}} P_\theta(\mathbf{v}, \mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \prod_i \exp(b_i v_i) \underbrace{\prod_j \Big(1 + \exp(a_j + \sum_i W_{ij} v_i)\Big)}_{\text{Product of experts}}$$

Markov random fields, Boltzmann machines, log-linear models.

Restricted Boltzmann Machines



Restricted: No interaction between hidden variables

Inferring the distribution over the hidden variables is easy:

$$\underbrace{P(\mathbf{h}|\mathbf{v}) = \prod_j P(h_j|\mathbf{v})}_{\text{Factorizes: Easy to compute}} \quad P(h_j = 1|\mathbf{v}) = \frac{1}{1 + \exp(-\sum_i W_{ij} v_i - a_j)}$$

Similarly:

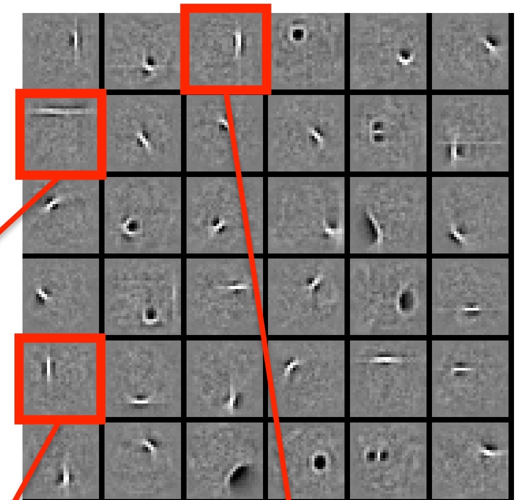
$$P(\mathbf{v}|\mathbf{h}) = \prod_i P(v_i|\mathbf{h}) \quad P(v_i = 1|\mathbf{h}) = \frac{1}{1 + \exp(-\sum_j W_{ij} h_j - b_i)}$$

Markov random fields, Boltzmann machines, log-linear models.

Learning Features

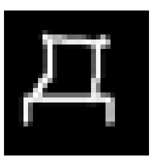
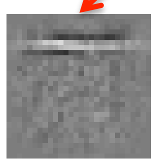
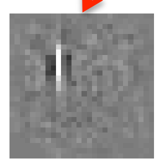
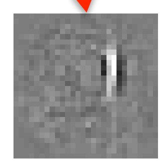
Observed Data
Subset of 25,000 characters

Learned W: “edges”
Subset of 1000 features



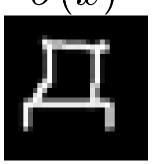
Most hidden variables are off

New Image: $p(h_7 = 1|v)$ $p(h_{29} = 1|v)$

 = $\sigma\Big($ $0.99 \times$  $+\ 0.97 \times$  $+\ 0.82 \times$  .... $\Big)$

$\sigma(x) = \frac{1}{1+\exp(-x)}$

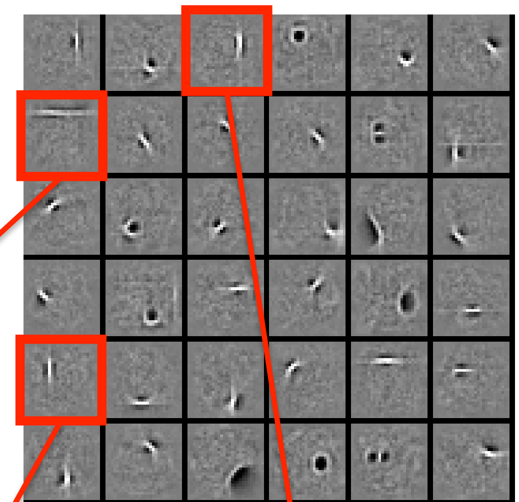
Logistic Function: Suitable for modeling binary images

Represent:  as $P(\mathbf{h}|\mathbf{v}) = [0,\ 0,\ 0.82,\ 0,\ 0,\ 0.99,\ 0,\ 0\ \ldots\ ]$

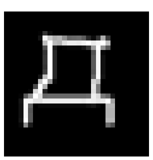
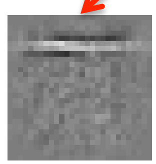
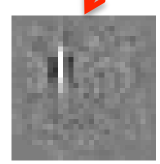
Learning Features

Observed Data
Subset of 25,000 characters

Learned W: "edges"
Subset of 1000 features



Most hidden variables are off

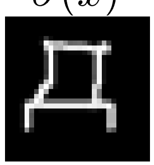
New Image:  $= \sigma\Big(0.99 \times$  $+ \; 0.97 \times$  $+$

$p(h_7 = 1|v)$ $\quad$ $p(h_{29} = 1|v)$

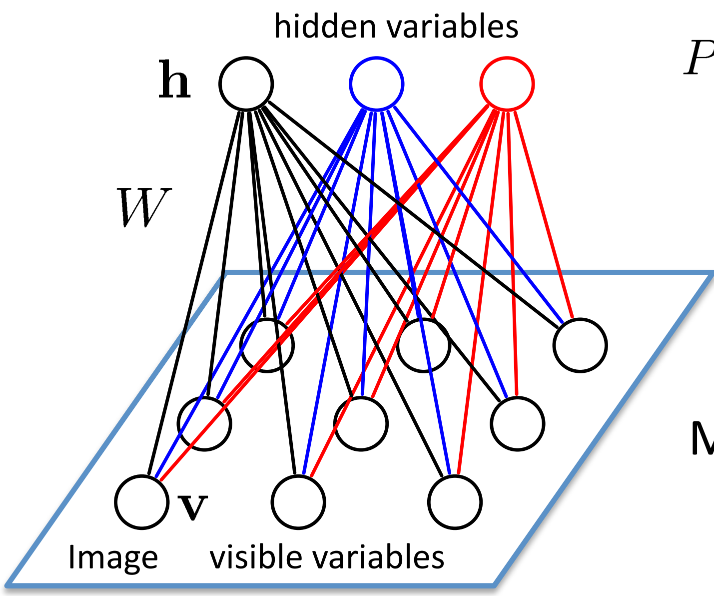
Easy to compute

$\sigma(x) = \frac{1}{1+\exp(-x)}$

Logistic Function: Suitable for modeling binary images

Represent:  as $P(\mathbf{h}|\mathbf{v}) = [0,\ 0,\ 0.82,\ 0,\ 0,\ 0.99,\ 0,\ 0\ \ldots\ ]$

Model Learning



$$P_\theta(\mathbf{v}) = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}} \exp\left[\mathbf{v}^\top W \mathbf{h} + \mathbf{a}^\top \mathbf{h} + \mathbf{b}^\top \mathbf{v}\right]$$

Given a set of *i.i.d.* training examples $\mathcal{D} = \{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, ..., \mathbf{v}^{(N)}\}$, we want to learn model parameters $\theta = \{W, a, b\}$.

Maximize (penalized) log-likelihood objective:

$$L(\theta) = \frac{1}{N} \sum_{n=1}^{N} \log P_\theta(\mathbf{v}^{(n)}) - \underbrace{\frac{\lambda}{N} ||W||_F^2}_{\text{Regularization}}$$

Derivative of the log-likelihood:

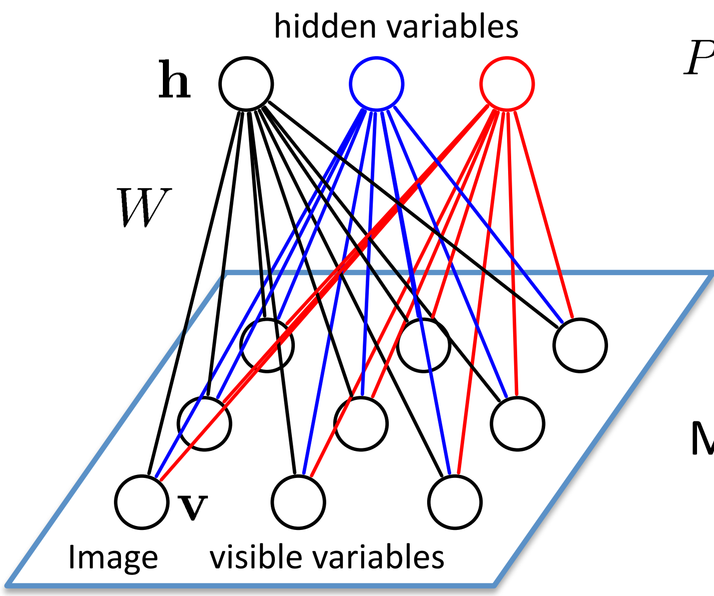
$$\frac{\partial L(\theta)}{\partial W_{ij}} = \frac{1}{N} \sum_{n=1}^{N} \frac{\partial}{\partial W_{ij}} \log\left(\sum_{\mathbf{h}} \exp\left[\mathbf{v}^{(n)\top} W \mathbf{h} + \mathbf{a}^\top \mathbf{h} + \mathbf{b}^\top \mathbf{v}^{(n)}\right]\right) - \frac{\partial}{\partial W_{ij}} \log \mathcal{Z}(\theta) - \frac{2\lambda}{N} W_{ij}$$

$$= \mathrm{E}_{P_{data}}[v_i h_j] - \underbrace{\mathrm{E}_{P_\theta}[v_i h_j]}_{\text{Difficult to compute: exponentially many configurations}} - \frac{2\lambda}{N} W_{ij}$$

$$P_{data}(\mathbf{v}, \mathbf{h}; \theta) = P(\mathbf{h}|\mathbf{v}; \theta) P_{data}(\mathbf{v})$$

$$P_{data}(\mathbf{v}) = \frac{1}{N} \sum_{n} \delta(\mathbf{v} - \mathbf{v}^{(n)})$$

Model Learning



$$P_\theta(\mathbf{v}) = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}} \exp\left[\mathbf{v}^\top W\mathbf{h} + \mathbf{a}^\top\mathbf{h} + \mathbf{b}^\top\mathbf{v}\right]$$

Given a set of *i.i.d.* training examples $\mathcal{D} = \{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, ..., \mathbf{v}^{(N)}\}$, we want to learn model parameters $\theta = \{W, a, b\}$.

Maximize (penalized) log-likelihood objective:

$$L(\theta) = \frac{1}{N}\sum_{n=1}^{N} \log P_\theta(\mathbf{v}^{(n)}) - \frac{\lambda}{N}||W||_F^2$$

Derivative of the log-likelihood:

$$\frac{\partial L(\theta)}{\partial W_{ij}} = \mathrm{E}_{P_{data}}[v_i h_j] - \mathrm{E}_{P_\theta}[v_i h_j] - \frac{2\lambda}{N}W_{ij}$$

Approximate maximum likelihood learning:

Contrastive Divergence (Hinton 2000)
MCMC-MLE estimator (Geyer 1991)

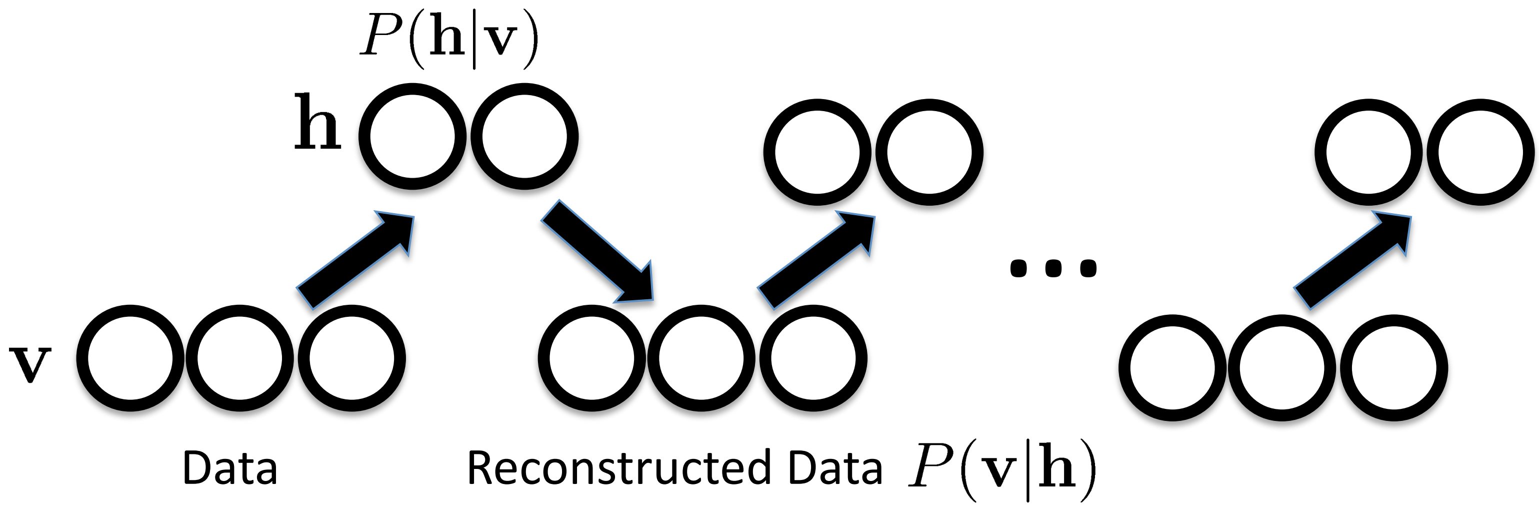
Tempered MCMC
(Salakhutdinov, NIPS 2009)

Pseudo Likelihood (Besag 1977)
Composite Likelihoods (Lindsay, 1988; Varin 2008)

Adaptive MCMC
(Salakhutdinov, ICML 2010)

Contrastive Divergence

Run Markov chain for a few steps (e.g. one step):



$$P(\mathbf{h}|\mathbf{v}) = \prod_j P(h_j|\mathbf{v}) \quad P(h_j = 1|\mathbf{v}) = \frac{1}{1 + \exp(-\sum_i W_{ij} v_i - a_j)}$$

$$P(\mathbf{v}|\mathbf{h}) = \prod_i P(v_i|\mathbf{h}) \quad P(v_i = 1|\mathbf{h}) = \frac{1}{1 + \exp(-\sum_j W_{ij} h_j - b_i)}$$

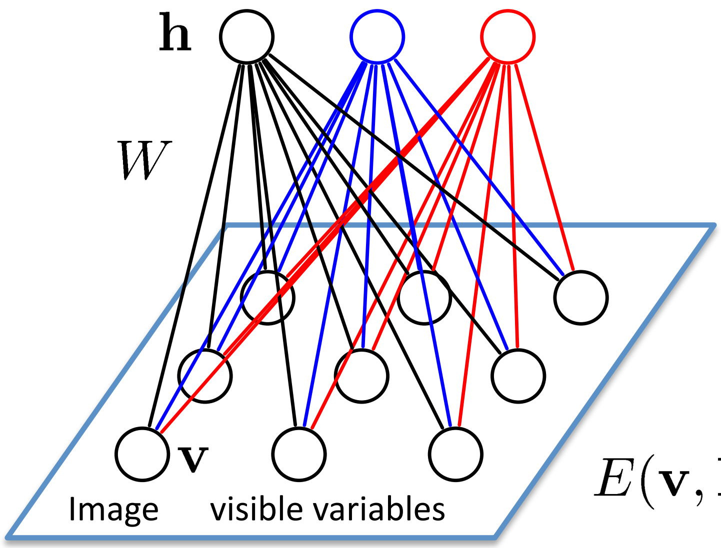
Update model parameters:

$$\Delta W_{ij} = \mathrm{E}_{P_{data}}[v_i h_j] - \mathrm{E}_{P_1}[v_i h_j]$$

RBMs for Images

(Salakhutdinov & Hinton, NIPS 2007)

Gaussian-Bernoulli RBM:



$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)}\exp(-E(\mathbf{v},\mathbf{h};\theta))$$

Define energy functions for various data modalities:

$$E(\mathbf{v},\mathbf{h};\theta) = \sum_i \frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_{ij} W_{ij} h_j \frac{v_i}{\sigma_i} - \sum_j a_j h_j$$

$$P(v_i = x|\mathbf{h}) = \frac{1}{\sqrt{2\pi}\sigma_i}\exp\left(-\frac{(x - b_i - \sigma_i \sum_j W_{ij} h_j)^2}{2\sigma_i^2}\right)$$ Gaussian

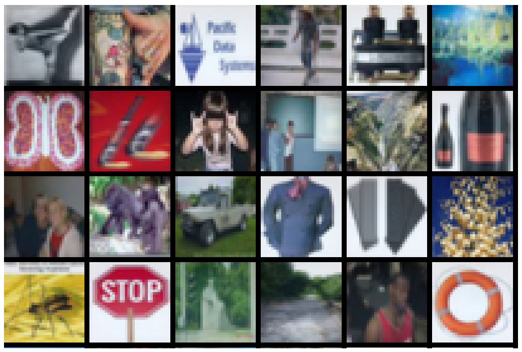
$$P(h_j = 1|\mathbf{v}) = \frac{1}{1 + \exp(-\sum_i W_{ij}\frac{v_i}{\sigma_i} - a_j)}$$ Bernoulli

RBMs for Images and Text

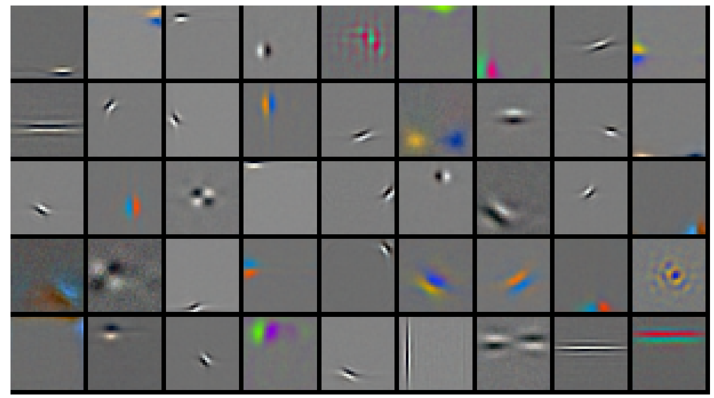
(Salakhutdinov & Hinton SIGIR 2007, NIPS 2010)

Images: Gaussian-Bernoulli RBM

4 million **unlabelled** images



Learned features (out of 10,000)



Text: Multinomial-Bernoulli RBM

WIKIPEDIA
The Free Encyclopedia

REUTERS

Associated Press

Reuters dataset:
804,414 **unlabeled**
newswire stories

Bag-of-Words

Learned features: ``topics''

russian	clinton	computer	trade	stock
russia	house	system	country	wall
moscow	president	product	import	street
yeltsin	bill	software	world	point
soviet	congress	develop	economy	dow

Collaborative Filtering

- Natural Images
- Text/Documents
- Collaborative Filtering/ Product Recommendation

Learned bases: ``genre''

Netflix dataset:
480,189 users
17,770 movies
Over 100 million ratings

Fahrenheit 9/11
Bowling for Columbine
The People vs. Larry Flynt
Canadian Bacon
La Dolce Vita

Independence Day
The Day After Tomorrow
Con Air
Men in Black II
Men in Black

Friday the 13th
The Texas Chainsaw Massacre
Children of the Corn
Child's Play
The Return of Michael Myers

State-of-the-art performance on the Netflix dataset.

Part of the wining solution in the Netflix contest.

Relates to **Probabilistic Matrix Factorization**
(Salakhutdinov & Mnih, NIPS 2008)

Salakhutdinov, Mnih, & Hinton, ICML 2007

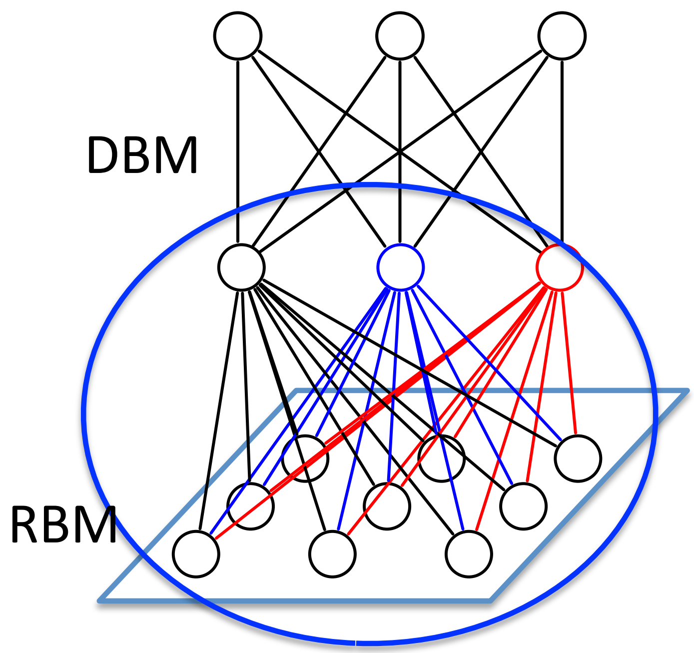
Multiple Application Domains

- Natural Images
- Text/Documents
- Collaborative Filtering / Matrix Factorization
  - Salakhutdinov & Mnih, NIPS 2008, ICML 2008;
  - Salakhutdinov & Srebro, NIPS 2011
  - Sutskever, Salakhutdinov, and Tenenbaum, NIPS 2010
- Video (Langford, Salakhutdinov and Zhang, ICML 2009)
- Motion Capture (Taylor et.al. NIPS 2007)
- Speech Perception (Dahl et. al. NIPS 2010, Lee et.al. NIPS 2010)

Same learning algorithm --
multiple input domains.

Limitations on the types of structure that can be represented by a single layer of low-level features!

Talk Roadmap

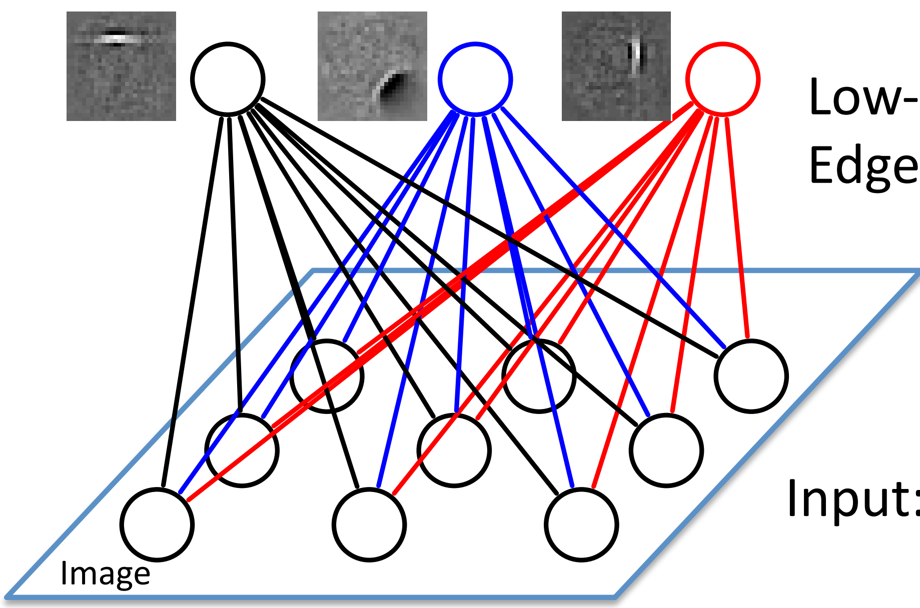


Part 1: Deep Networks

- Introduction: Graphical Models.
- Restricted Boltzmann Machines: Learning low-level features.
- Deep Belief Networks: Learning Part-based Hierarchies.
- Deep Boltzmann Machines.

Deep Belief Network

(Hinton et.al. Neural Computation 2006)



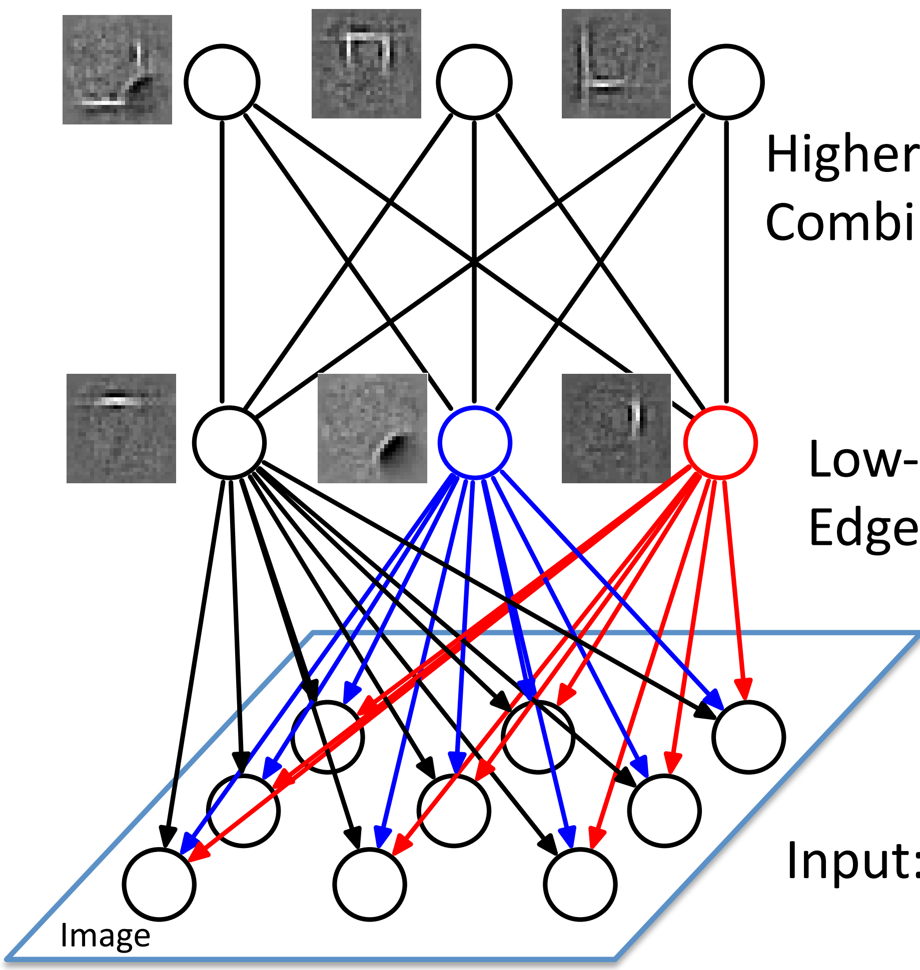
Low-level features:
Edges

Built from **unlabeled** inputs.

Input: Pixels

Deep Belief Network

(Hinton et.al. Neural Computation 2006)



Internal representations capture higher-order statistical structure

Higher-level features: Combination of edges

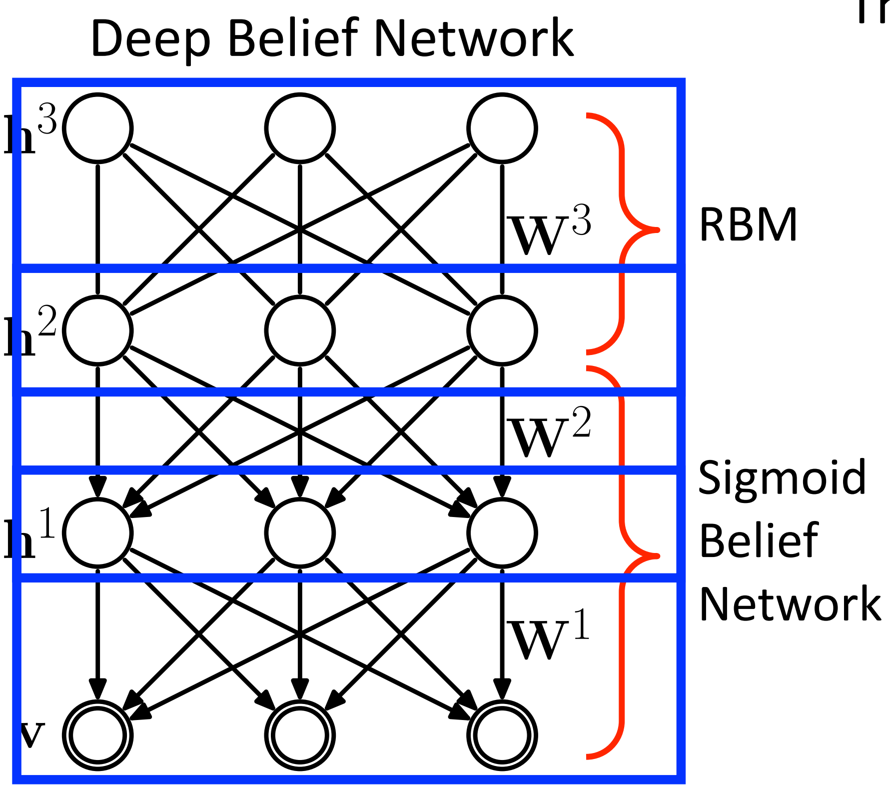
Low-level features: Edges

Built from **unlabeled** inputs.

Input: Pixels

Unsupervised feature learning.

Deep Belief Network



Unsupervised Feature Learning.

The joint probability distribution factorizes:

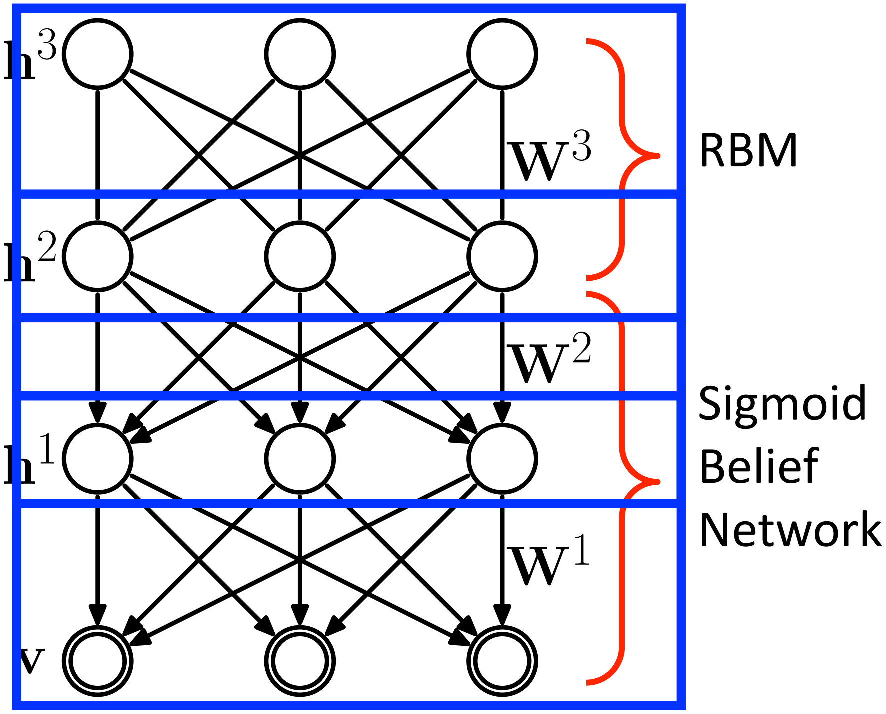
$$P(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2, \mathbf{h}^3) = P(\mathbf{v}|\mathbf{h}^1)P(\mathbf{h}^1|\mathbf{h}^2)P(\mathbf{h}^2, \mathbf{h}^3)$$

Layerwise Pretraining:

- Learn and freeze 1st layer RBM
- Treat inferred values $P(\mathbf{h}^1|\mathbf{v})$ as the data for training 2nd-layer RBM.
- Learn and freeze 2nd layer RBM.
- Proceed to the next layer.

Deep Belief Network

Deep Belief Network



Unsupervised Feature Learning.

The joint probability distribution factorizes:

$$P(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2, \mathbf{h}^3)$$
$$= P(\mathbf{v}|\mathbf{h}^1)P(\mathbf{h}^1|\mathbf{h}^2)P(\mathbf{h}^2, \mathbf{h}^3)$$

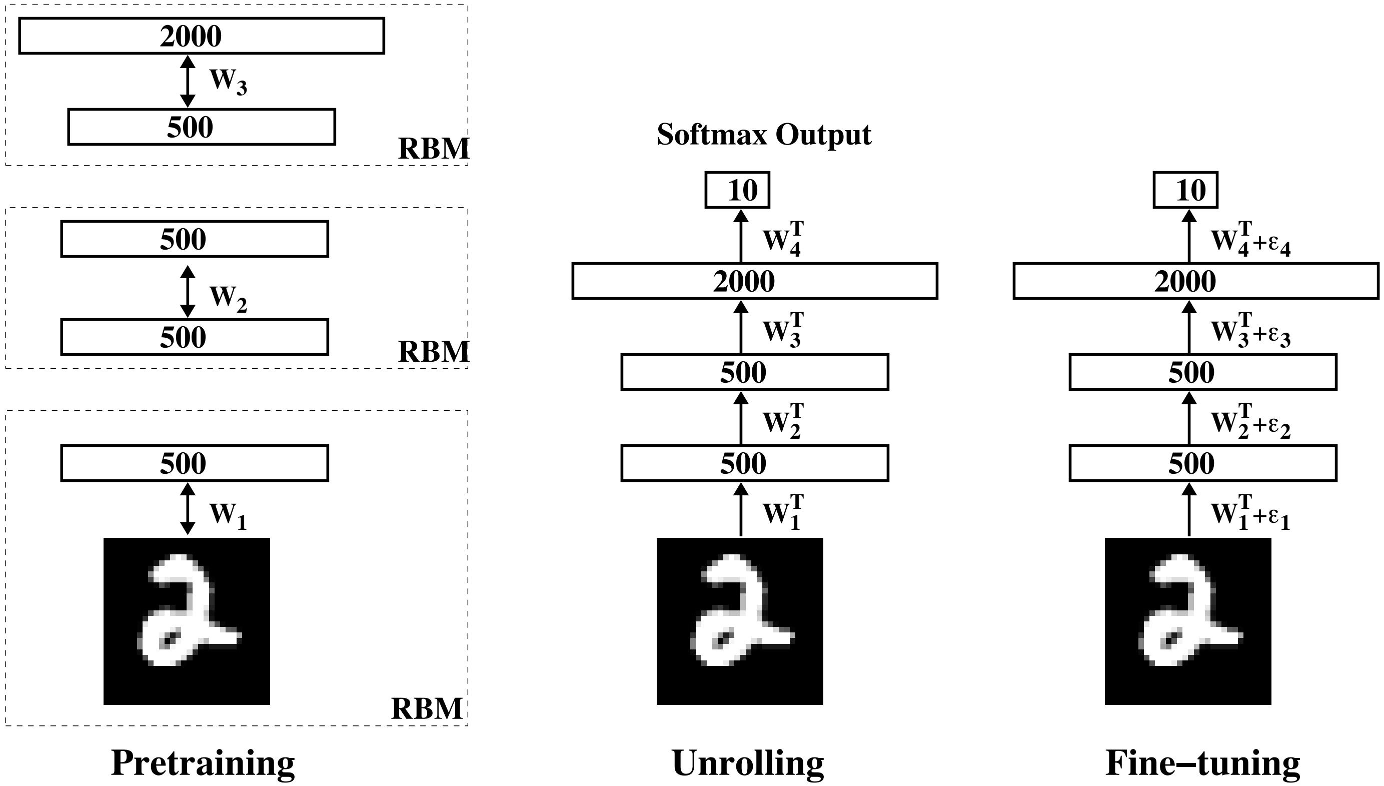
Layerwise Pretraining:

- Learn and freeze 1st layer RBM
- Treat inferred values $P(\mathbf{h}^1|\mathbf{v})$ as the data for training 2nd-

Layerwise pretraining improves variational lower bound

DBNs for Classification

(Hinton and Salakhutdinov, Science 2006)



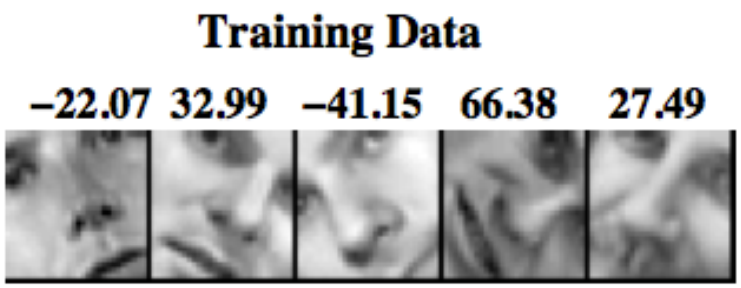
- After layer-by-layer **unsupervised pretraining**, discriminative fine-tuning by backpropagation achieves an error rate of 1.2% on MNIST. SVM's get 1.4% and randomly initialized backprop gets 1.6%.

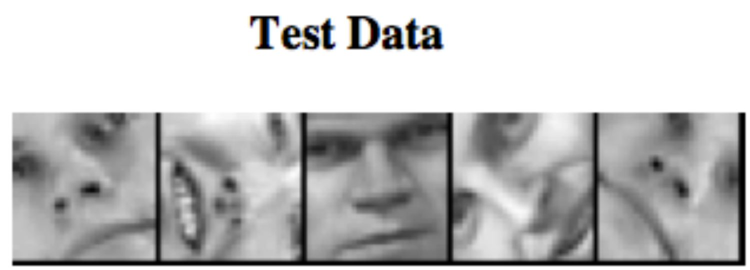
- Clearly unsupervised learning helps generalization. It ensures that most of the information in the weights comes from modeling the input data.

DBNs for Regression

(Salakhutdinov and Hinton, NIPS 2007)

Predicting the orientation of a face patch





Training Data: 1000 face patches of 30 training people.

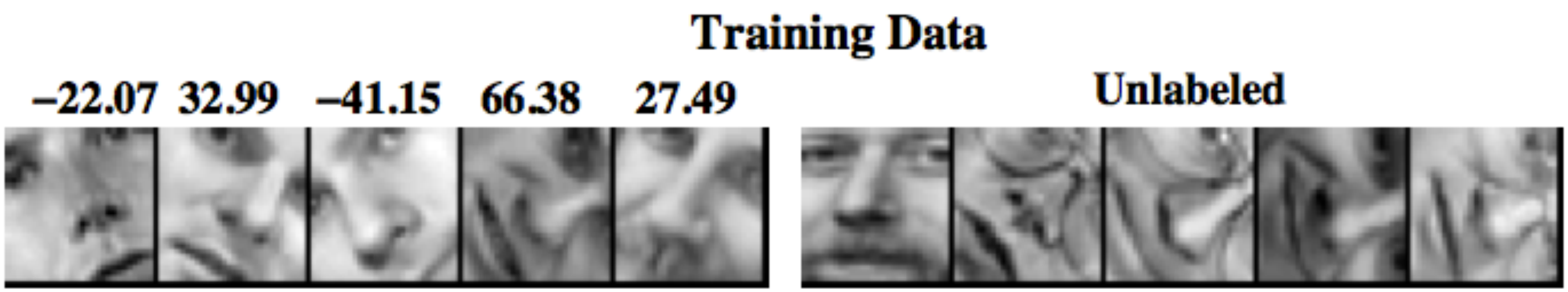
Test Data: 1000 face patches of **10 new people**.

Regression Task: predict orientation of a new face.

Gaussian Processes with spherical Gaussian kernel achieves a RMSE (root mean squared error) of 16.33 degree.

DBNs for Regression

(Salakhutdinov and Hinton, NIPS 2007)



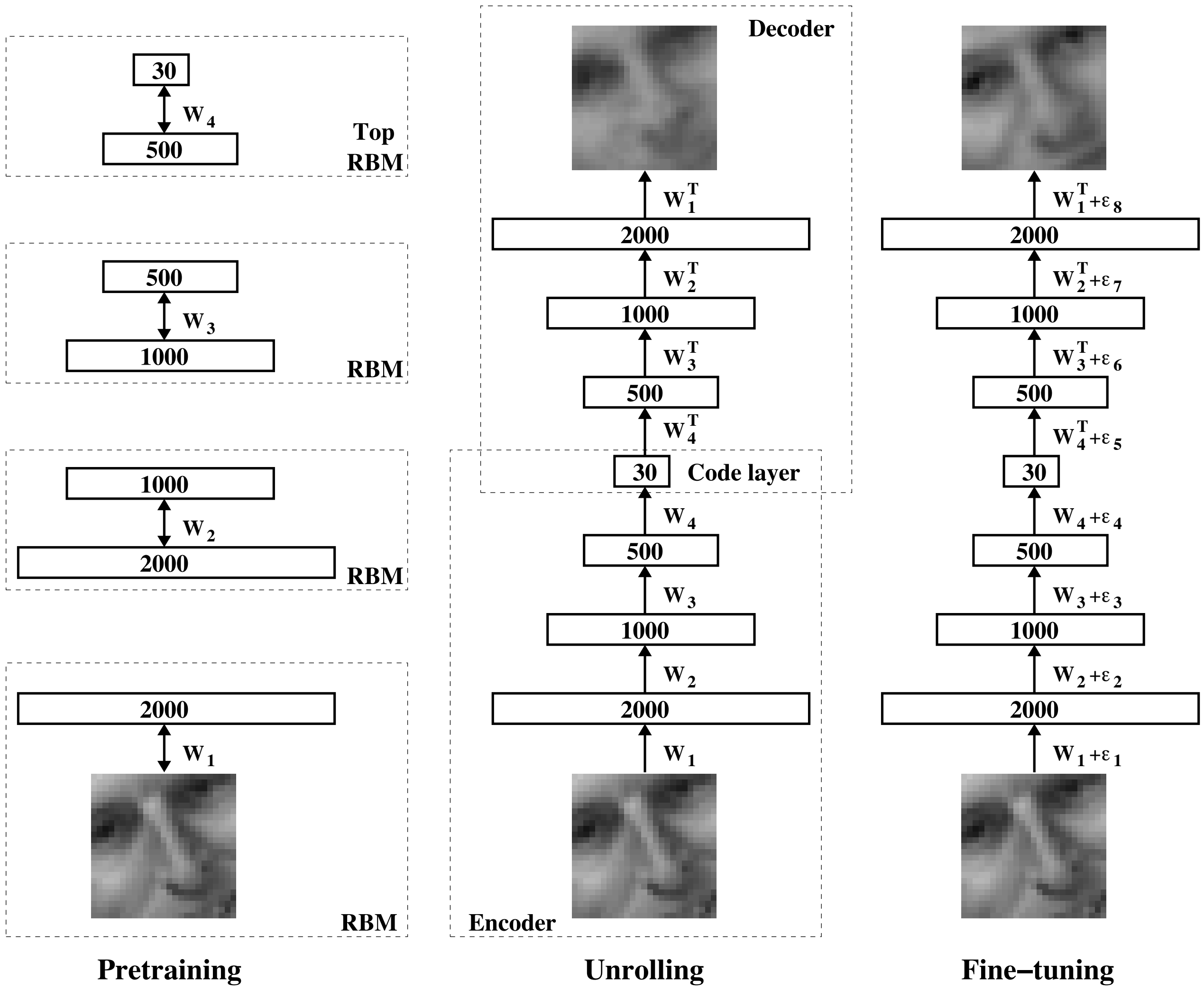
Additional Unlabeled Training Data: 12000 face patches from 30 training people.

- Pretrain a stack of RBMs: 784-1000-1000-1000.
- **Features were extracted with no idea of the final task.**

The same GP on the top-level features:	RMSE: 11.22
GP with fine-tuned covariance Gaussian kernel:	RMSE: 6.42
Standard GP without using DBNs:	RMSE: 16.33

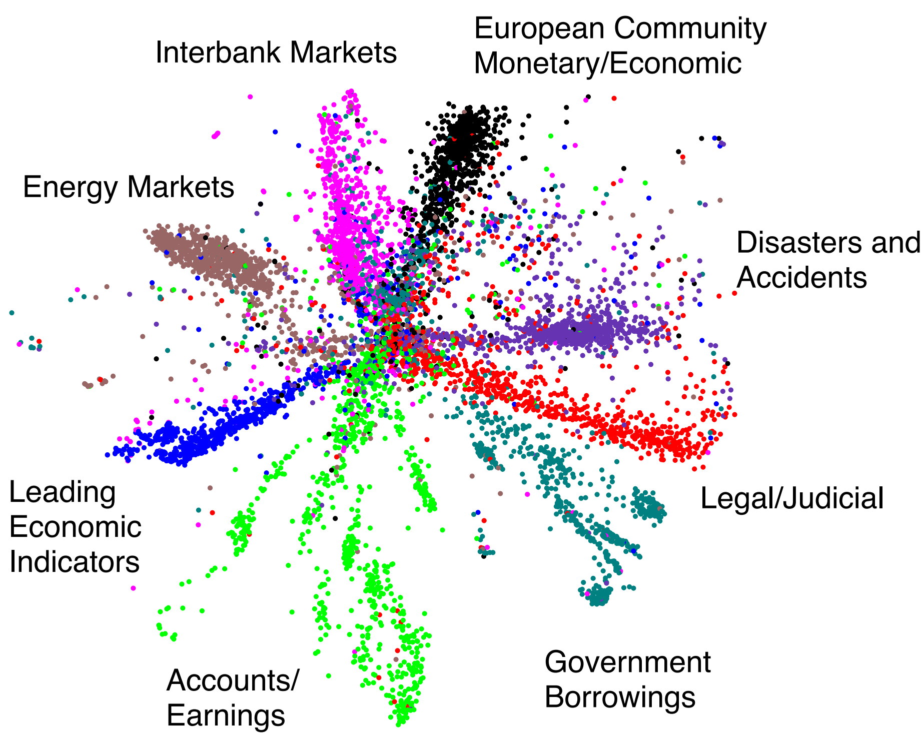
Deep Autoencoders

(Hinton and Salakhutdinov, Science 2006)

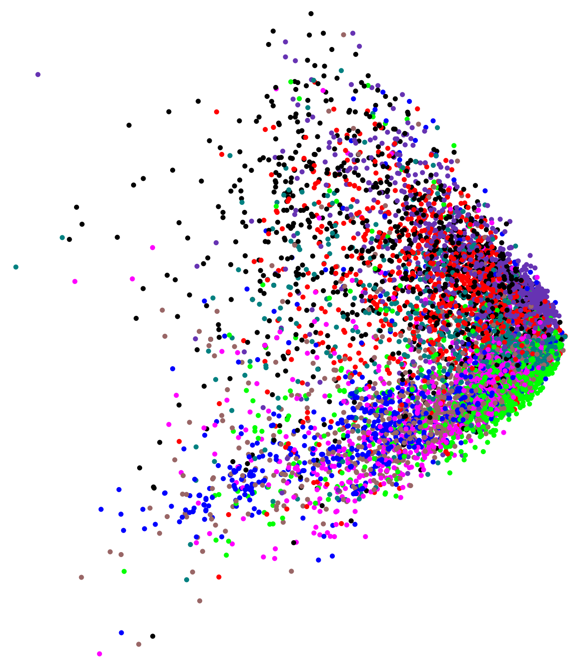


Information Retrieval

(Hinton and Salakhutdinov, Science 2006)



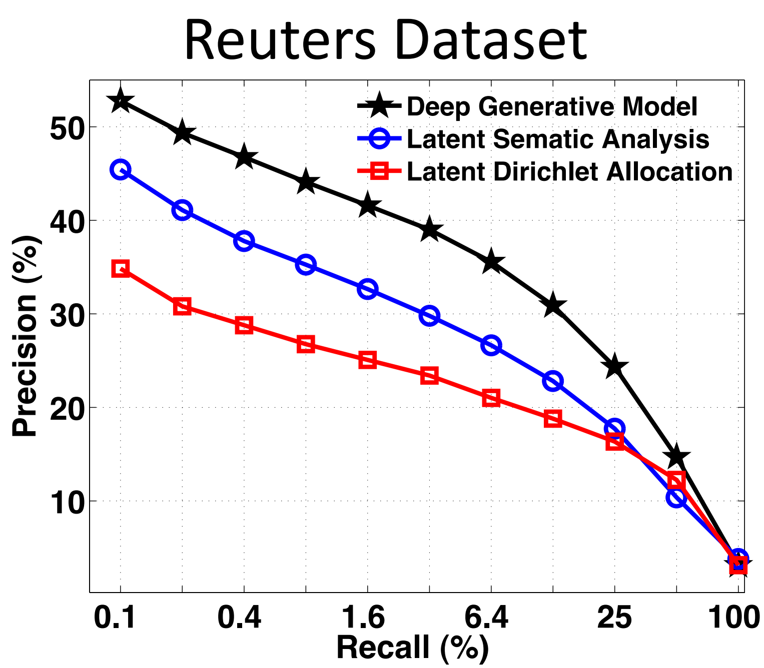
2-D LSA space



• The Reuters Corpus Volume II contains 804,414 newswire stories (randomly split into **402,207 training** and **402,207 test).**

• “Bag-of-words” representation: each article is represented as a vector containing the counts of the most frequently used 2000 words in the training set.

Information Retrieval

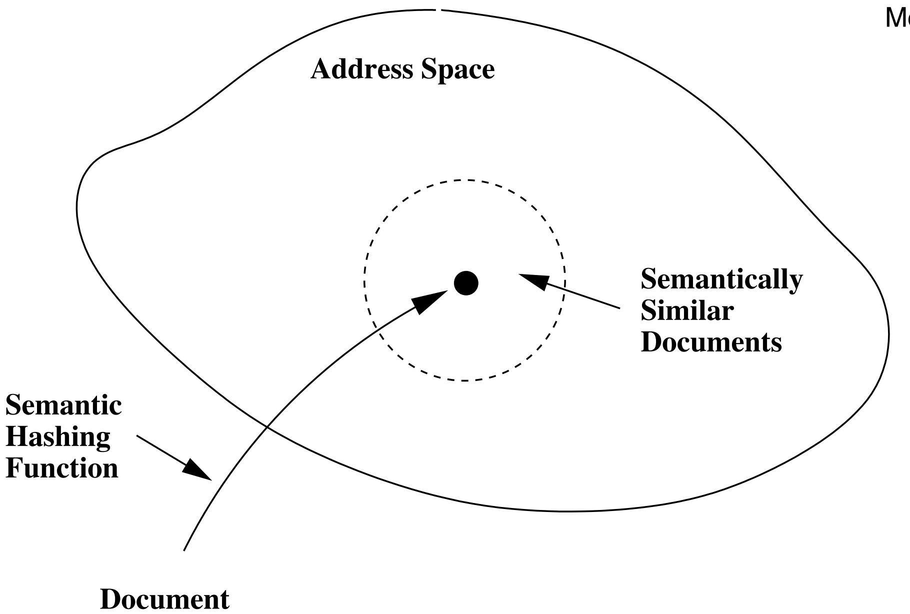


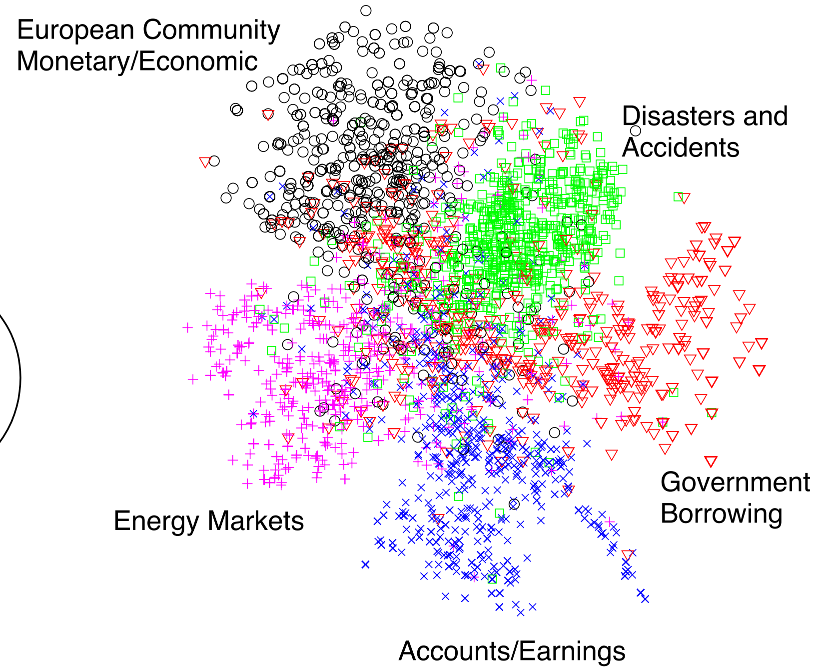
Reuters dataset: 804,414 newswire stories.

Deep generative model significantly outperforms LSA and LDA topic models

Semantic Hashing

(Salakhutdinov and Hinton, SIGIR 2007)

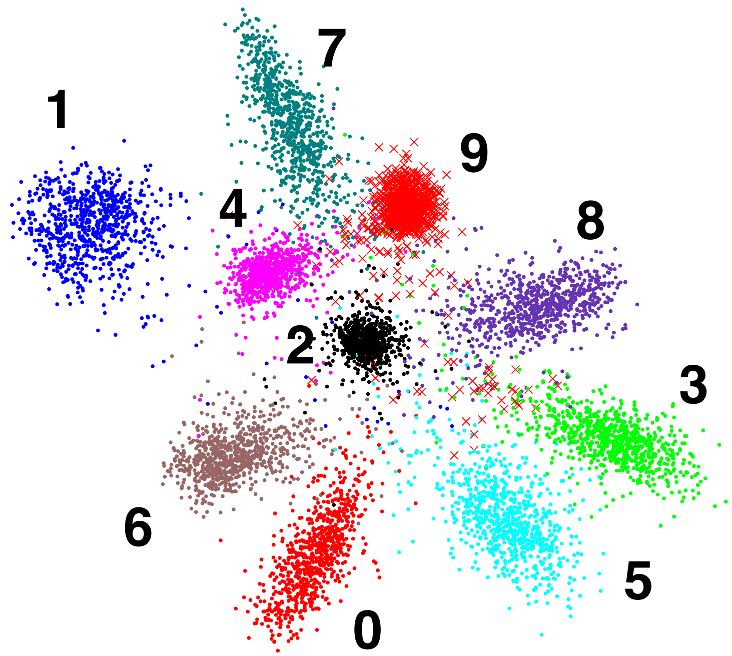


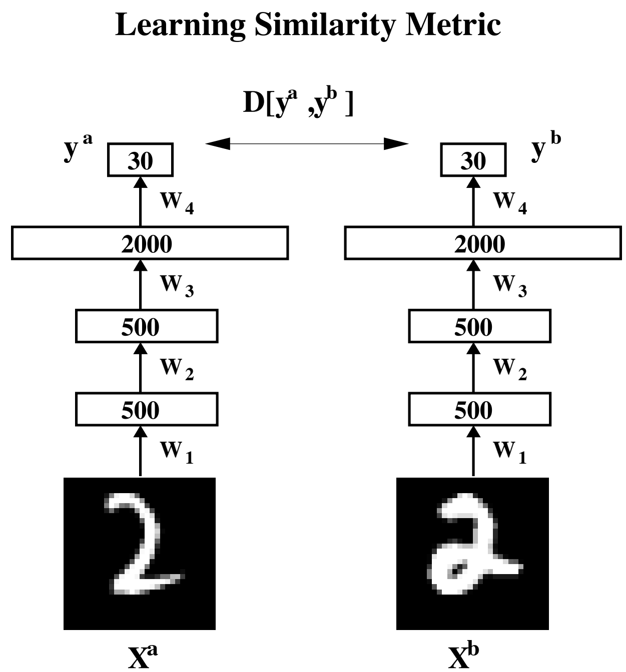


- Learn to map documents into **semantic 20-D binary codes.**
- Retrieve similar documents stored at the nearby addresses **with no search at all.**

Learning Similarity Measures

(Salakhutdinov and Hinton, AI and Statistics 2007)

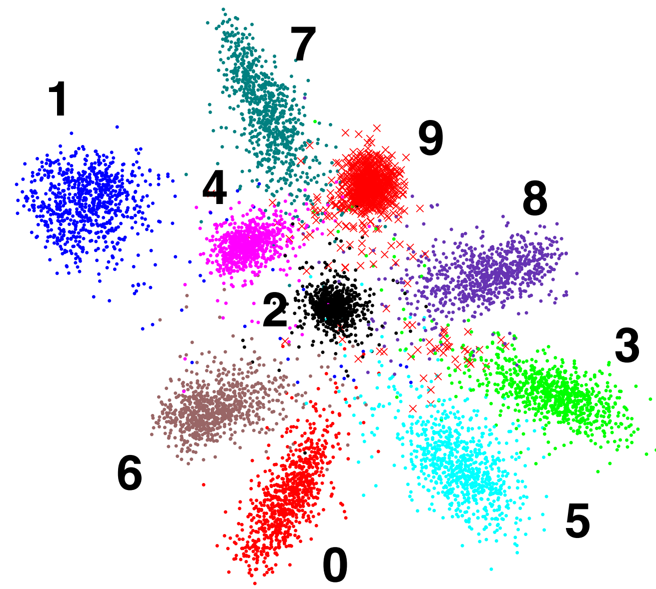


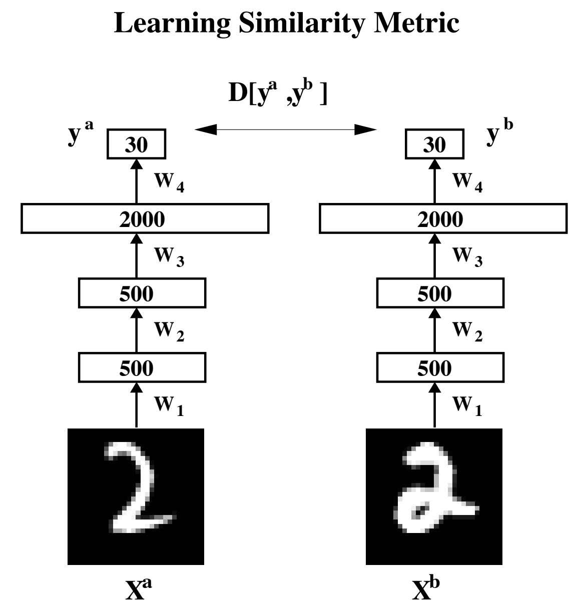


- Learn a nonlinear transformation of the input space.
- Optimize to make KNN perform well in the low-dimensional feature space

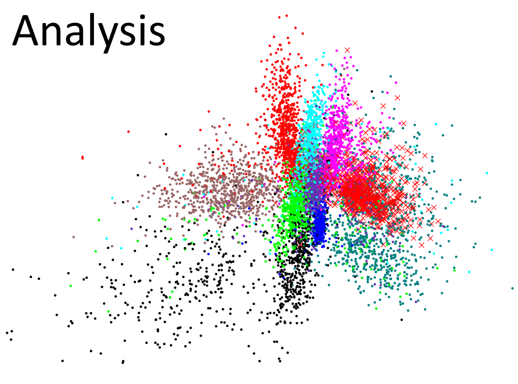
Compare to Other Approaches

(Salakhutdinov and Hinton, AI and Statistics 2007)

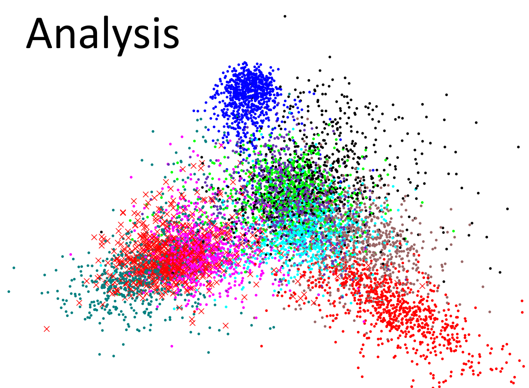




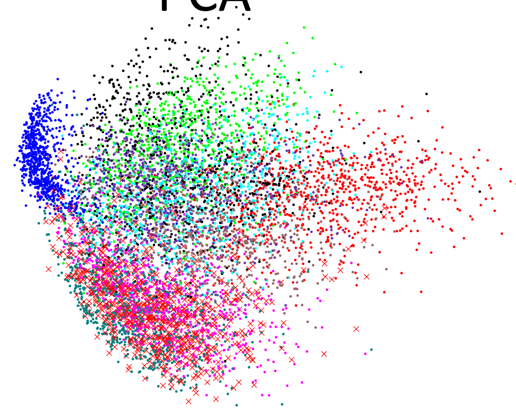
Neighborhood Component Analysis



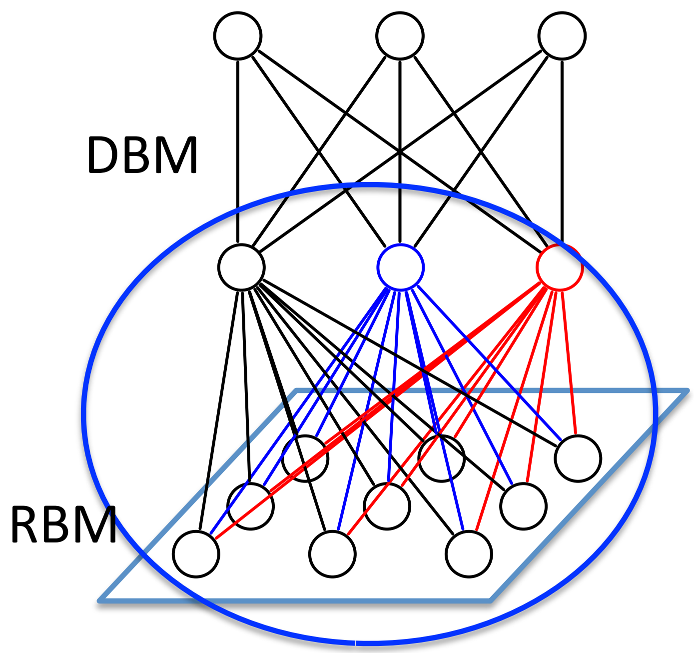
Linear Discriminant Analysis



PCA



Talk Roadmap

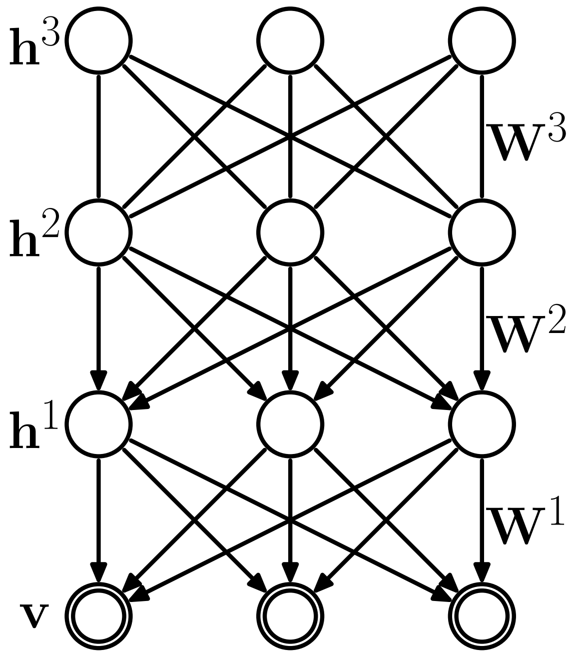


Part 1: Deep Networks

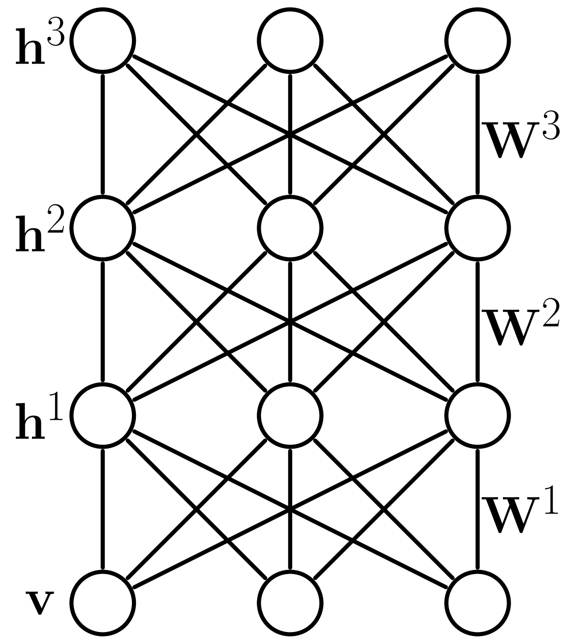
- Introduction: Graphical Models.
- Restricted Boltzmann Machines: Learning low-level features.
- Deep Belief Networks: Learning Part-based Hierarchies.
- Deep Boltzmann Machines.

DBNs vs. DBMs

Deep Belief Network



Deep Boltzmann Machine



DBNs are hybrid models:

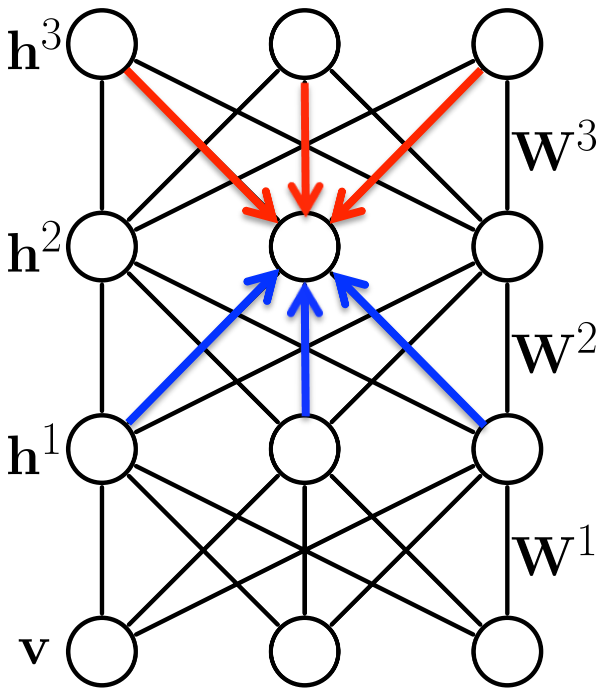
- Inference in DBNs is problematic due to **explaining away**.
- Only greedy pretrainig, **no joint optimization over all layers**.
- Approximate inference is feed-forward: **no bottom-up and top-down**.

Introduce a new class of models called Deep Boltzmann Machines.

Mathematical Formulation

$$P_\theta(\mathbf{v}) = \frac{P^*(\mathbf{v})}{\mathcal{Z}(\theta)} = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}^1,\mathbf{h}^2,\mathbf{h}^3} \exp\left[\mathbf{v}^\top W^1 \mathbf{h}^1 + \underline{\mathbf{h}^{1\top} W^2 \mathbf{h}^2} + \underline{\mathbf{h}^{2\top} W^3 \mathbf{h}^3}\right]$$

Deep Boltzmann Machine



Input

$\theta = \{W^1, W^2, W^3\}$ model parameters

- Dependencies between hidden variables.
- All connections are undirected.
- Bottom-up and Top-down:

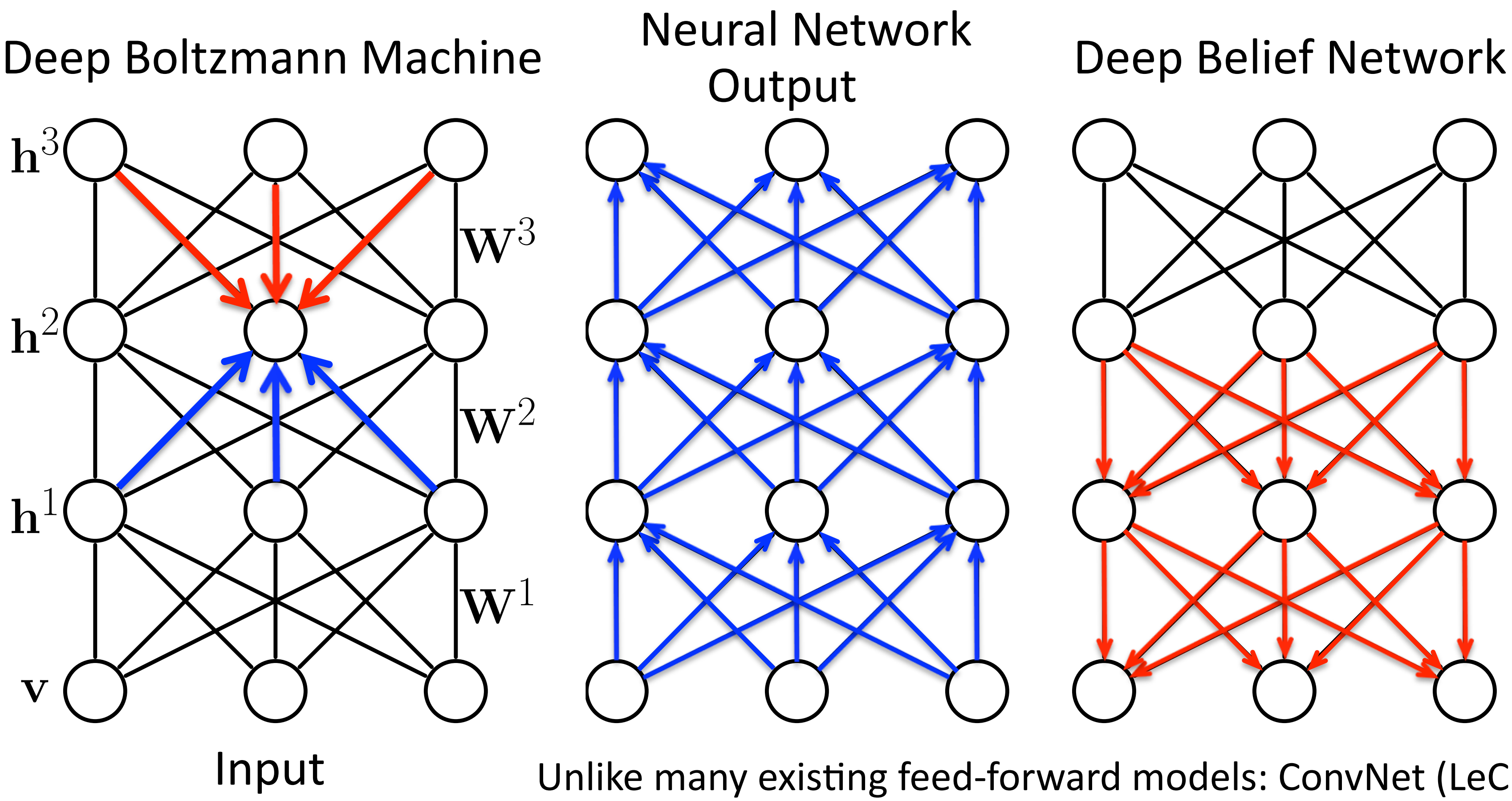
$$P(h_j^2 = 1 | \mathbf{h}^1, \mathbf{h}^3) = \sigma\left(\sum_k W_{kj}^3 h_k^3 + \sum_m W_{mj}^2 h_m^1\right)$$

Top-down   Bottom-up

Unlike many existing feed-forward models: ConvNet (LeCun), HMAX (Poggio et.al.), Deep Belief Nets (Hinton et.al.)

Mathematical Formulation

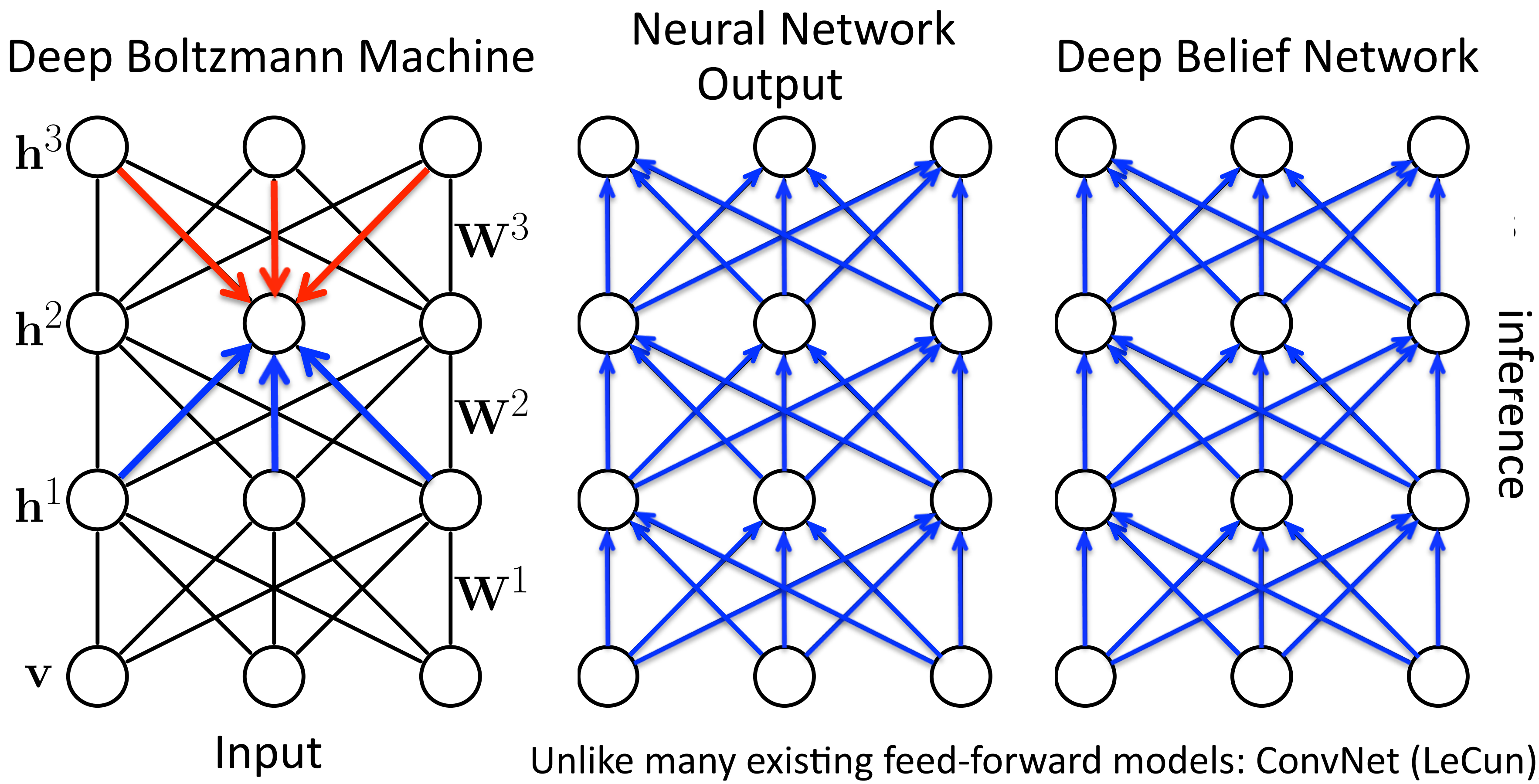
$$P_\theta(\mathbf{v}) = \frac{P^*(\mathbf{v})}{\mathcal{Z}(\theta)} = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}^1,\mathbf{h}^2,\mathbf{h}^3} \exp\left[\mathbf{v}^\top W^1 \mathbf{h}^1 + \mathbf{h}^{1\top} W^2 \mathbf{h}^2 + \mathbf{h}^{2\top} W^3 \mathbf{h}^3\right]$$



Unlike many existing feed-forward models: ConvNet (LeCun), HMAX (Poggio), Deep Belief Nets (Hinton)

Mathematical Formulation

$$P_\theta(\mathbf{v}) = \frac{P^*(\mathbf{v})}{\mathcal{Z}(\theta)} = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}^1,\mathbf{h}^2,\mathbf{h}^3} \exp\left[\mathbf{v}^\top W^1 \mathbf{h}^1 + \mathbf{h}^{1\top} W^2 \mathbf{h}^2 + \mathbf{h}^{2\top} W^3 \mathbf{h}^3\right]$$

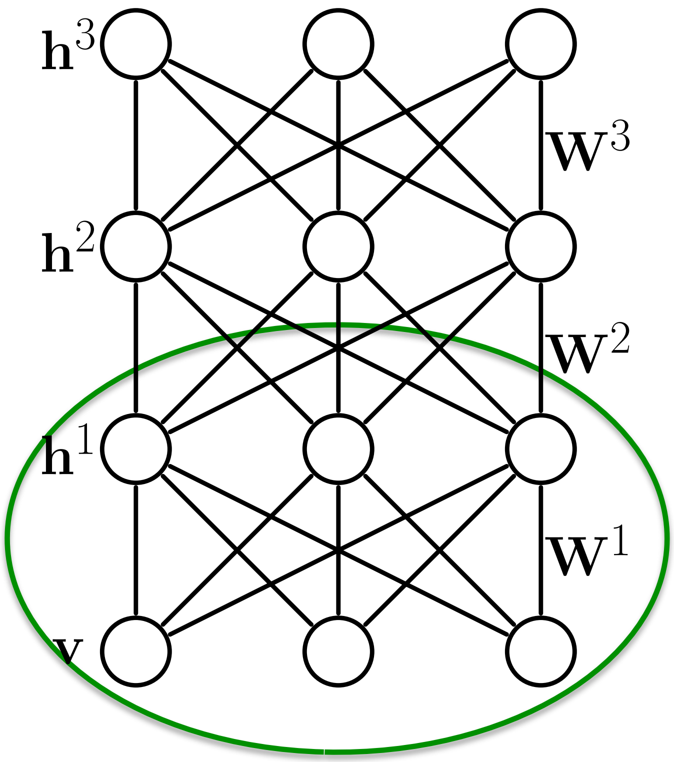


Unlike many existing feed-forward models: ConvNet (LeCun), HMAX (Poggio), Deep Belief Nets (Hinton)

Mathematical Formulation

$$P_\theta(\mathbf{v}) = \frac{P^*(\mathbf{v})}{\mathcal{Z}(\theta)} = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}^1,\mathbf{h}^2,\mathbf{h}^3} \exp\left[\mathbf{v}^\top W^1 \mathbf{h}^1 + \mathbf{h}^{1\top} W^2 \mathbf{h}^2 + \mathbf{h}^{2\top} W^3 \mathbf{h}^3\right]$$

Deep Boltzmann Machine



$\theta = \{W^1, W^2, W^3\}$ model parameters

- Dependencies between hidden variables.

Maximum likelihood learning:

$$\frac{\partial \log P_\theta(\mathbf{v})}{\partial W^1} = \mathrm{E}_{P_{data}}[\mathbf{v}\mathbf{h}^{1\top}] - \mathrm{E}_{P_\theta}[\mathbf{v}\mathbf{h}^{1\top}]$$

Problem: Both expectations are intractable!

Learning rule for undirected graphical models: MRFs, CRFs, Factor graphs.

Previous Work

Many approaches for learning Boltzmann machines have been proposed over the last 20 years:

- Hinton and Sejnowski (1983),
- Peterson and Anderson (1987)
- Galland (1991)
- Kappen and Rodriguez (1998)
- Lawrence, Bishop, and Jordan (1998)
- Tanaka (1998)
- Welling and Hinton (2002)
- Zhu and Liu (2002)
- Welling and Teh (2003)
- Yasuda and Tanaka (2009)

Real-world applications – thousands of hidden and observed variables with millions of parameters.

Many of the previous approaches were not successful for learning general Boltzmann machines with **hidden variables.**

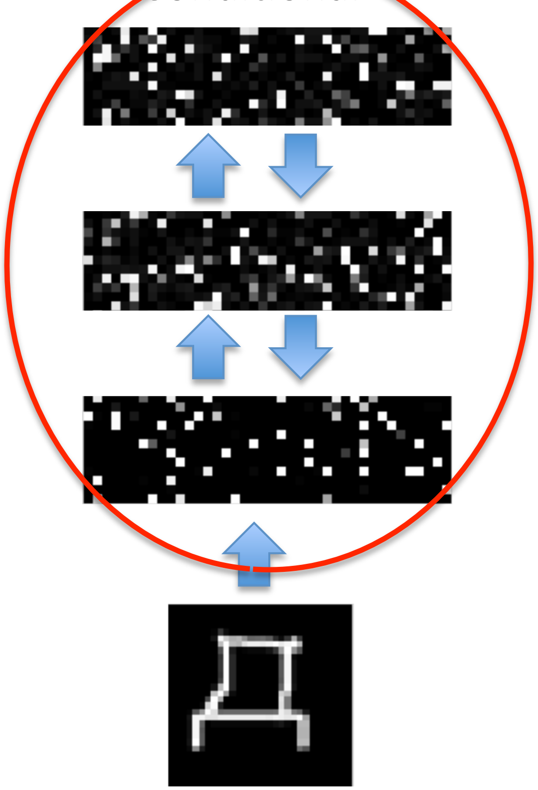
Algorithms based on Contrastive Divergence, Score Matching, Pseudo-Likelihood, Composite Likelihood, MCMC-MLE, Piecewise Learning, cannot handle multiple layers of hidden variables.

New Learning Algorithm

(Salakhutdinov, 2008; NIPS 2009)

Posterior Inference

Conditional



Approximate conditional

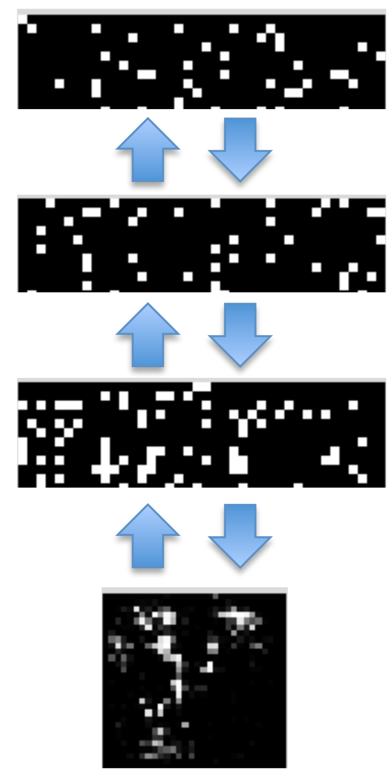
$$P_{data}(\mathbf{h}|\mathbf{v})$$

Simulate from the Model

Approximate the joint distribution

$$P_{model}(\mathbf{h}, \mathbf{v})$$

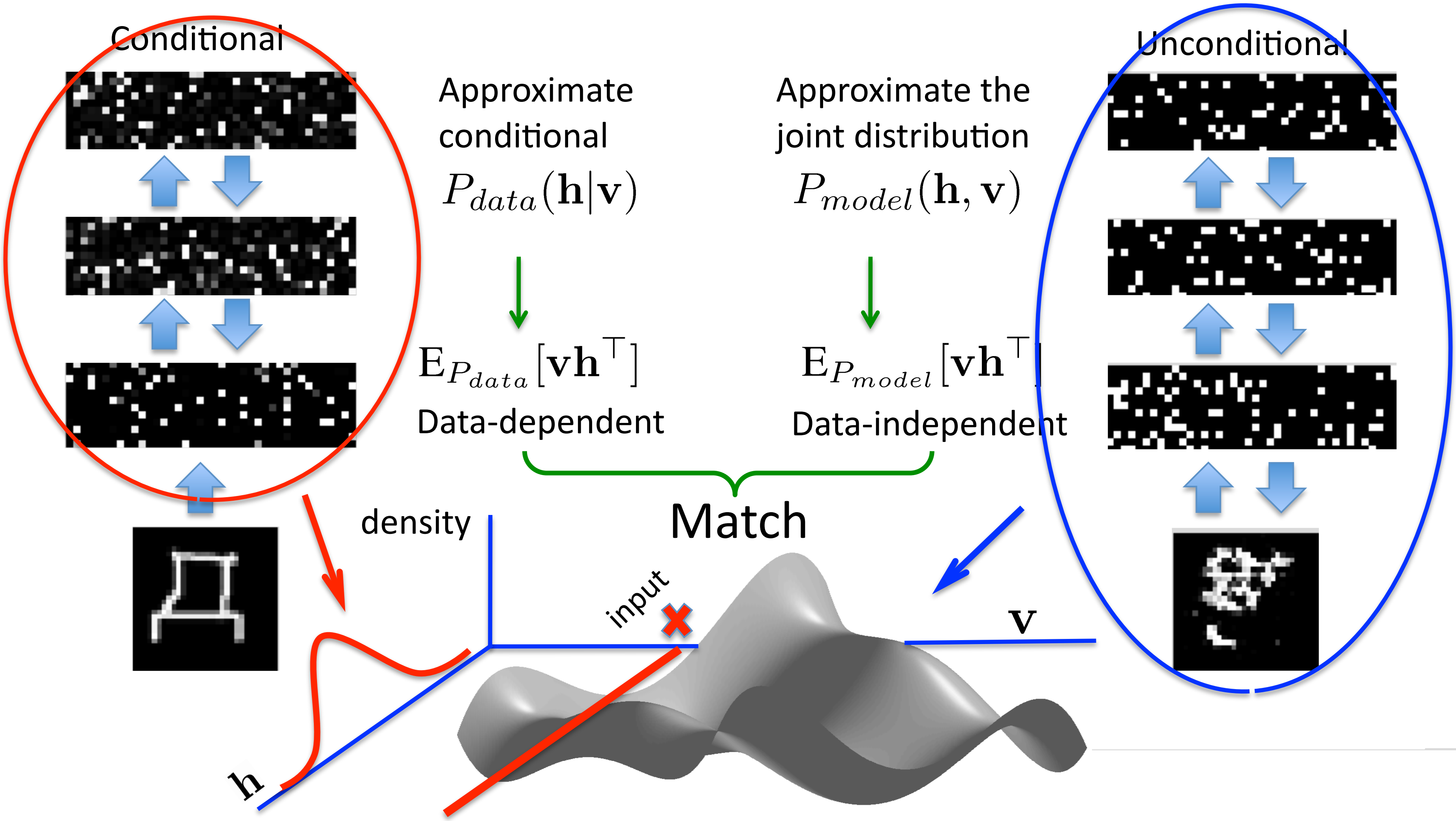
Unconditional



New Learning Algorithm

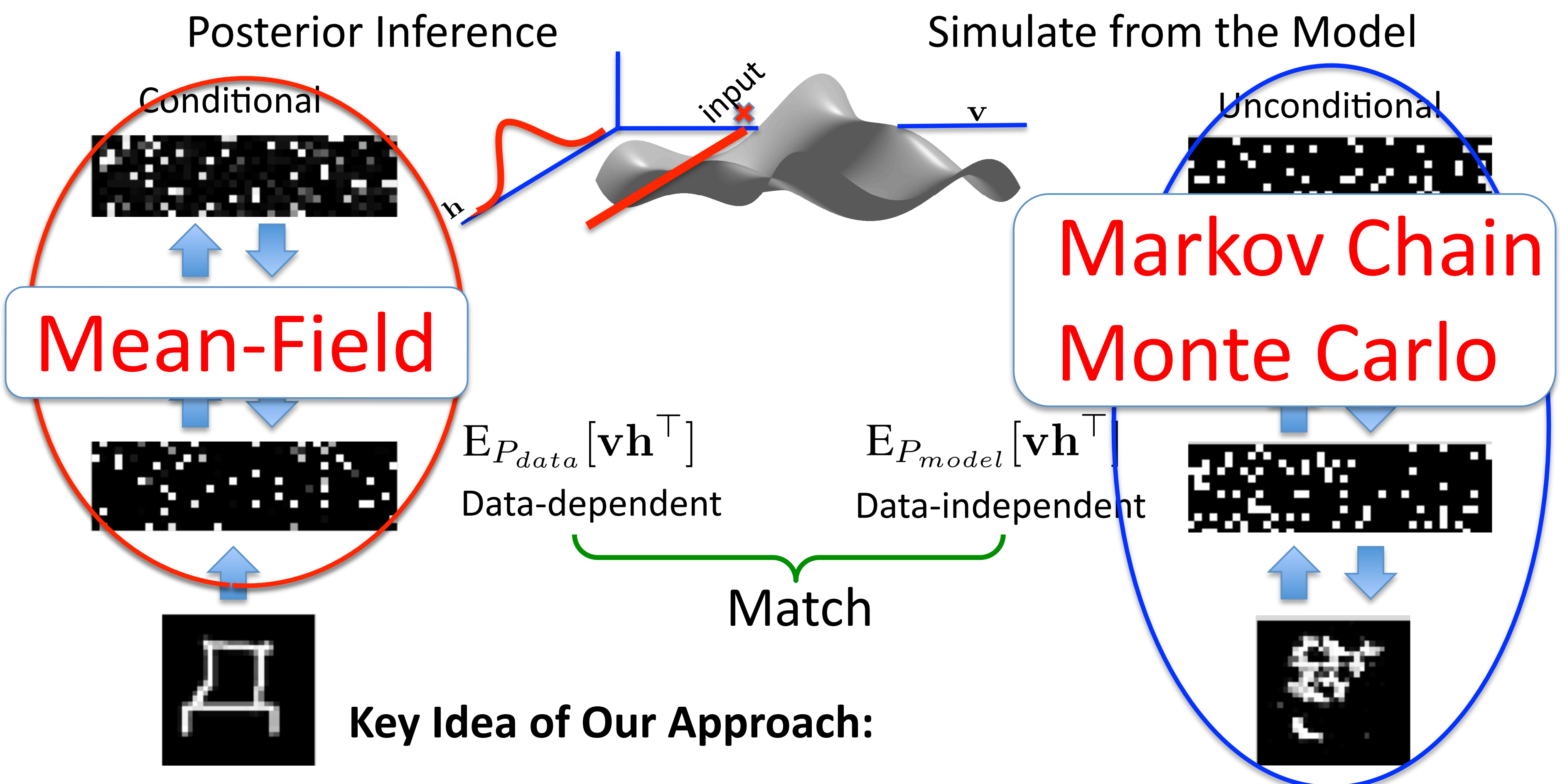
(Salakhutdinov, 2008; NIPS 2009)

Posterior Inference | Simulate from the Model



New Learning Algorithm

(Salakhutdinov, 2008; NIPS 2009)



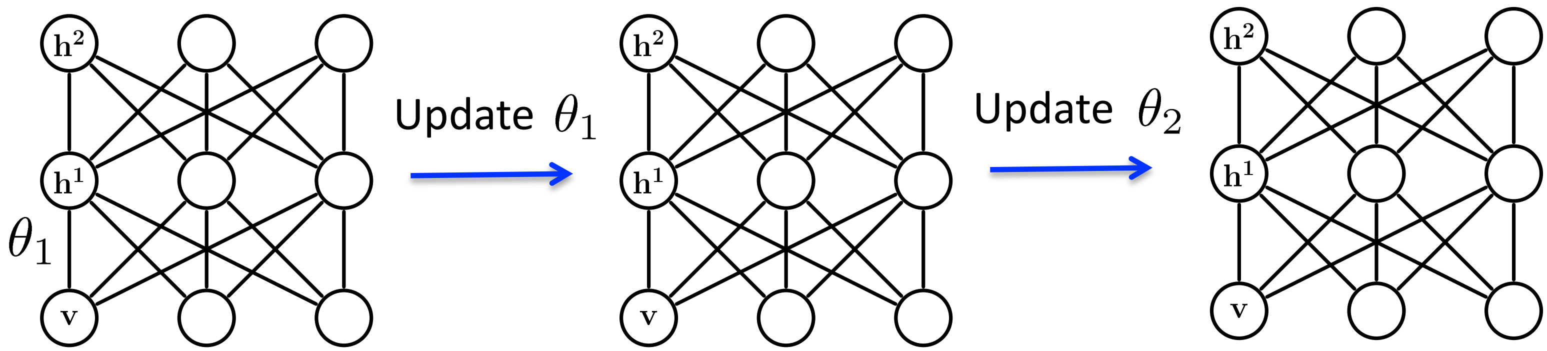
Key Idea of Our Approach:

Data-dependent: **Variational Inference**, mean-field theory

Data-independent: **Stochastic Approximation**, MCMC based

Stochastic Approximation

Time t=1 t=2 t=3



$\mathbf{x}_1 \sim \mathrm{T}_{\theta_1}(\mathbf{x}_1 \leftarrow \mathbf{x}_0)$ $\quad$ $\mathbf{x}_2 \sim \mathrm{T}_{\theta_2}(\mathbf{x}_2 \leftarrow \mathbf{x}_1)$ $\quad$ $\mathbf{x}_3 \sim \mathrm{T}_{\theta_3}(\mathbf{x}_3 \leftarrow \mathbf{x}_2)$

Update θ_t and $\mathbf{x}_t$ sequentially, where $\mathbf{x} = \{\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2\}$

- Generate $\mathbf{x}_t \sim \mathrm{T}_{\theta_t}(\mathbf{x}_t \leftarrow \mathbf{x}_{t-1})$ by simulating from a Markov chain that leaves P_{θ_t} invariant (e.g. Gibbs or M-H sampler)
- Update θ_t by replacing intractable $\mathrm{E}_{P_{\theta_t}}[\mathbf{v}\mathbf{h}^\top]$ with a point estimate $[\mathbf{v}_t \mathbf{h}_t^\top]$

In practice we simulate several Markov chains in parallel.

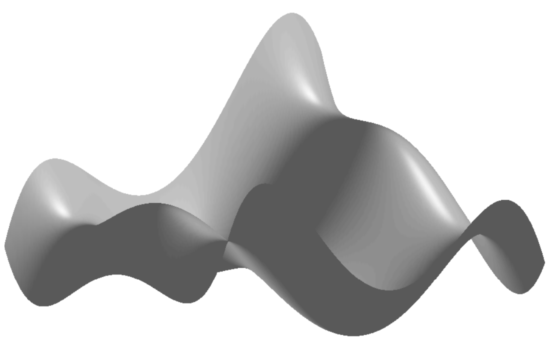
Robbins and Monro, Ann. Math. Stats, 1957
L. Younes, Probability Theory 1989

Stochastic Approximation

Update rule decomposes:

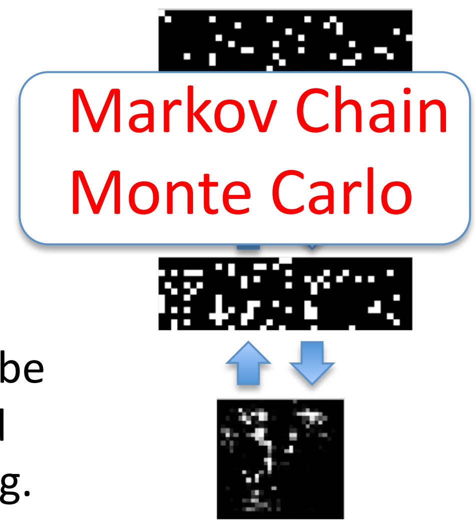
$$\theta_{t+1} = \theta_t + \alpha_t \underbrace{\left(\mathrm{E}_{P_{data}}[\mathbf{v}\mathbf{h}^\top] - \mathrm{E}_{P_{\theta_t}}[\mathbf{v}\mathbf{h}^\top] \right)}_{\text{True gradient}} + \alpha_t \underbrace{\left(\mathrm{E}_{P_{\theta_t}}[\mathbf{v}\mathbf{h}^\top] - \frac{1}{M}\sum_{m=1}^{M} \mathbf{v}_t^{(m)} \mathbf{h}_t^{(m)\top} \right)}_{\text{Noise term } \epsilon_t}$$

Almost sure convergence guarantees as learning rate $\alpha_t \rightarrow 0$



Salakhutdinov,
ICML 2010

Problem: High-dimensional data: the energy landscape is highly multimodal



Key insight: The transition operator can be any valid transition operator – Tempered Transitions, Parallel/Simulated Tempering.

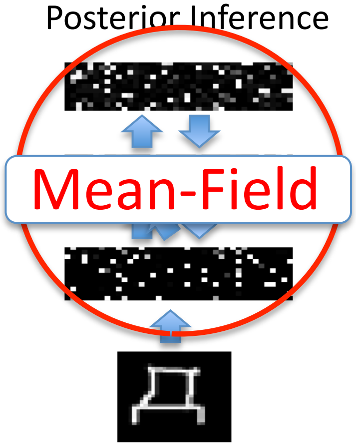
Connections to the theory of stochastic approximation and adaptive MCMC.

Variational Inference

(Salakhutdinov, 2008; Salakhutdinov & Larochelle, AI & Statistics 2010)

Approximate intractable distribution $P_\theta(\mathbf{h}|\mathbf{v})$ with simpler, tractable distribution $Q_\mu(\mathbf{h}|\mathbf{v})$:

$$\log P_\theta(\mathbf{v}) = \log \sum_{\mathbf{h}} P_\theta(\mathbf{h},\mathbf{v}) = \log \sum_{\mathbf{h}} Q_\mu(\mathbf{h}|\mathbf{v}) \frac{P_\theta(\mathbf{h},\mathbf{v})}{Q_\mu(\mathbf{h}|\mathbf{v})}$$



$$\geq \sum_{\mathbf{h}} Q_\mu(\mathbf{h}|\mathbf{v}) \log \frac{P_\theta(\mathbf{h},\mathbf{v})}{Q_\mu(\mathbf{h}|\mathbf{v})}$$

$$\underbrace{= \sum_{\mathbf{h}} Q_\mu(\mathbf{h}|\mathbf{v}) \log \underbrace{P^*_\theta(\mathbf{h},\mathbf{v})}_{\mathbf{v}^\top W^1 \mathbf{h}^1 + \mathbf{h}^{1\top} W^2 \mathbf{h}^2 + \mathbf{h}^{2\top} W^3 \mathbf{h}^3} - \log \mathcal{Z}(\theta) + \sum_{\mathbf{h}} Q_\mu(\mathbf{h}|\mathbf{v}) \log \frac{1}{Q_\mu(\mathbf{h}|\mathbf{v})}}_{\text{Variational Lower Bound}}$$

$$= \log P_\theta(\mathbf{v}) - \mathrm{KL}\big(Q_\mu(\mathbf{h}|\mathbf{v}) || P_\theta(\mathbf{h}|\mathbf{v})\big)$$

$$\mathrm{KL}(Q||P) = \int Q(x) \log \frac{Q(x)}{P(x)} dx$$

Minimize KL between approximating and true distributions with respect to variational parameters μ .

Variational Inference

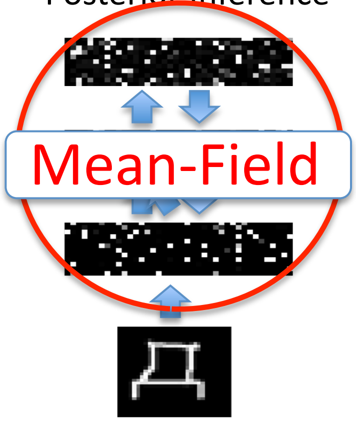
(Salakhutdinov, 2008; Salakhutdinov & Larochelle, AI & Statistics 2010)

Approximate intractable distribution $P_\theta(\mathbf{h}|\mathbf{v})$ with simpler, tractable distribution $Q_\mu(\mathbf{h}|\mathbf{v})$:

$$\mathrm{KL}(Q||P) = \int Q(x)\log\frac{Q(x)}{P(x)}dx$$

$$\log P_\theta(\mathbf{v}) \geq \underbrace{\log P_\theta(\mathbf{v}) - \mathrm{KL}\big(Q_\mu(\mathbf{h}|\mathbf{v})||P_\theta(\mathbf{h}|\mathbf{v})\big)}_{\text{Variational Lower Bound}}$$

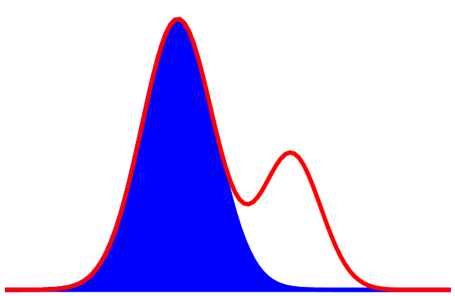
Posterior Inference



Mean-Field: Choose a fully factorized distribution:

$$Q_\mu(\mathbf{h}|\mathbf{v}) = \prod_{j=1}^{F} q(h_j|\mathbf{v}) \;\text{ with }\; q(h_j = 1|\mathbf{v}) = \mu_j$$

Variational Inference: Maximize the lower bound w.r.t. Variational parameters μ.



Nonlinear fixed-point equations:

$$\mu_j^{(1)} = \sigma\Big(\sum_i W_{ij}^1 v_i + \sum_k W_{jk}^2 \mu_k^{(2)}\Big)$$

$$\mu_k^{(2)} = \sigma\Big(\sum_j W_{jk}^2 \mu_j^{(1)} + \sum_m W_{km}^3 \mu_m^{(3)}\Big)$$

$$\mu_m^{(3)} = \sigma\Big(\sum_k W_{km}^3 \mu_k^{(2)}\Big)$$

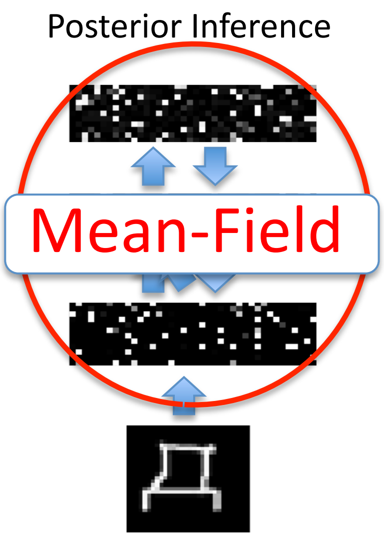
Variational Inference

(Salakhutdinov, 2008; Salakhutdinov & Larochelle, AI & Statistics 2010)

Approximate intractable distribution $P_\theta(\mathbf{h}|\mathbf{v})$ with simpler, tractable distribution $Q_\mu(\mathbf{h}|\mathbf{v})$:

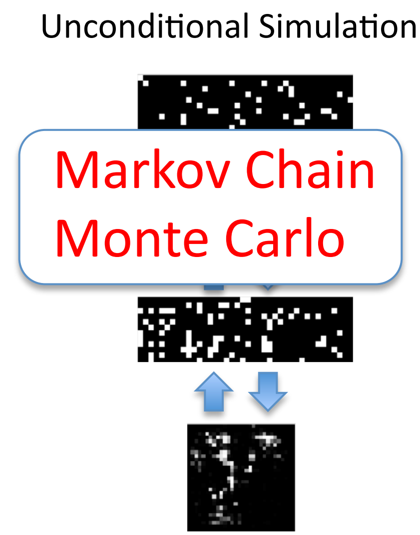
$$\mathrm{KL}(Q||P) = \int Q(x)\log\frac{Q(x)}{P(x)}dx$$

$$\log P_\theta(\mathbf{v}) \geq \underbrace{\log P_\theta(\mathbf{v}) - \mathrm{KL}\big(Q_\mu(\mathbf{h}|\mathbf{v})||P_\theta(\mathbf{h}|\mathbf{v})\big)}_{\text{Variational Lower Bound}}$$



1. Variational Inference: Maximize the lower bound w.r.t. variational parameters

2. MCMC: Apply stochastic approximation to update model parameters



Almost sure convergence guarantees to an asymptotically stable point.

Variational Inference

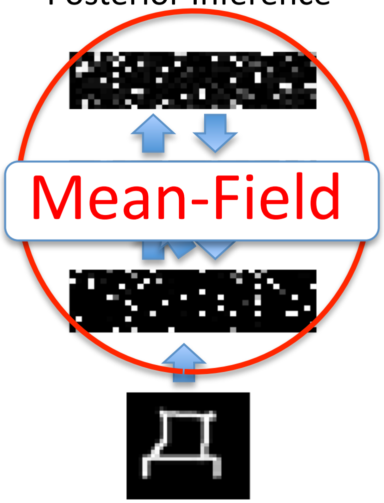
(Salakhutdinov, 2008; Salakhutdinov & Larochelle, AI & Statistics 2010)

Approximate intractable distribution $P_\theta(\mathbf{h}|\mathbf{v})$ with simpler, tractable distribution $Q_\mu(\mathbf{h}|\mathbf{v})$:

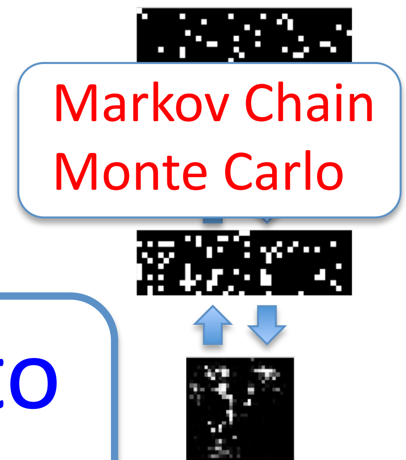
$$\mathrm{KL}(Q||P) = \int Q(x) \log \frac{Q(x)}{P(x)} dx$$

$$\log P_\theta(\mathbf{v}) \geq \underbrace{\log P_\theta(\mathbf{v}) - \mathrm{KL}\big(Q_\mu(\mathbf{h}|\mathbf{v}) || P_\theta(\mathbf{h}|\mathbf{v})\big)}_{\text{Variational Lower Bound}}$$

Posterior Inference



Unconditional Simulation



1. V... wer bou...

Fast Inference

2. M... to u...

Learning can scale to millions of examples

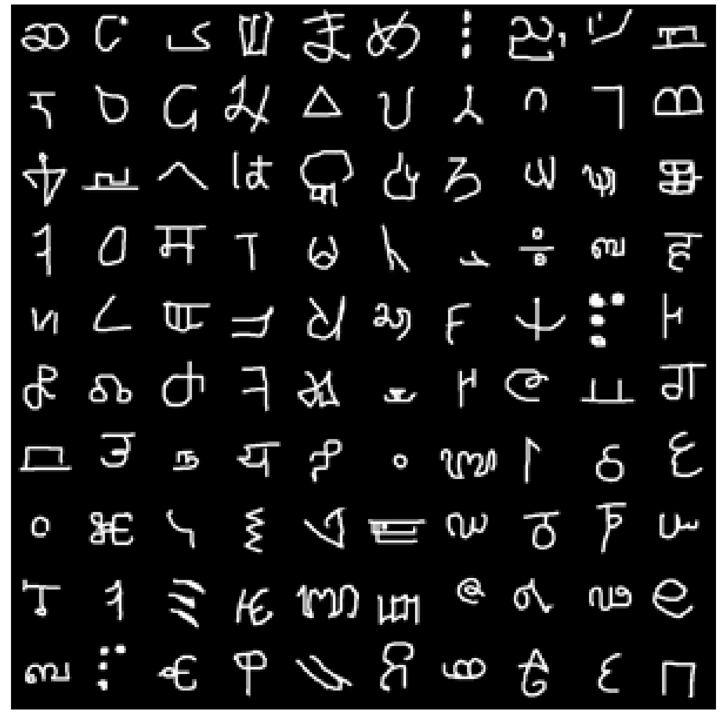
Almost sure convergence guarantees to an asymptotically stable point.

Good Generative Model?

Handwritten Characters

Good Generative Model?

Handwritten Characters



Good Generative Model?

Handwritten Characters

Simulated

Real Data

Good Generative Model?

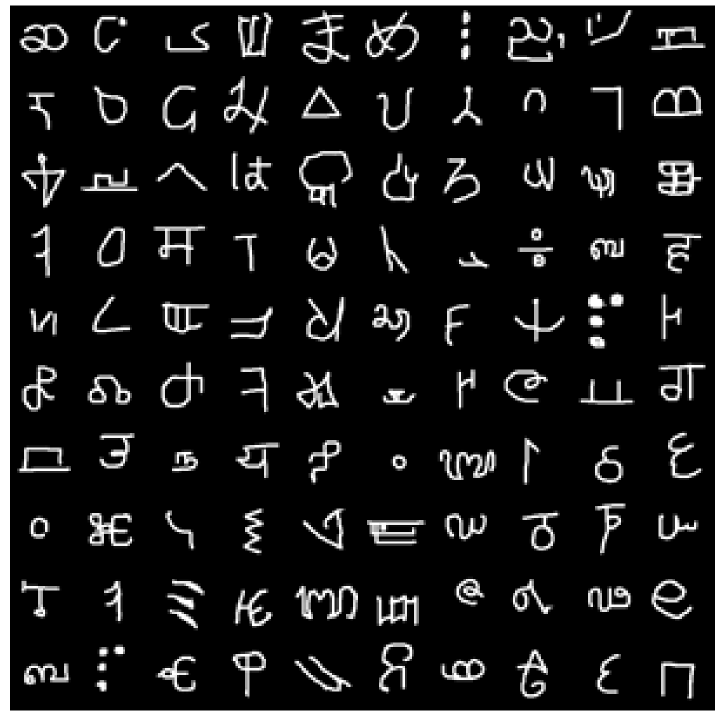
Handwritten Characters

Real Data

Simulated

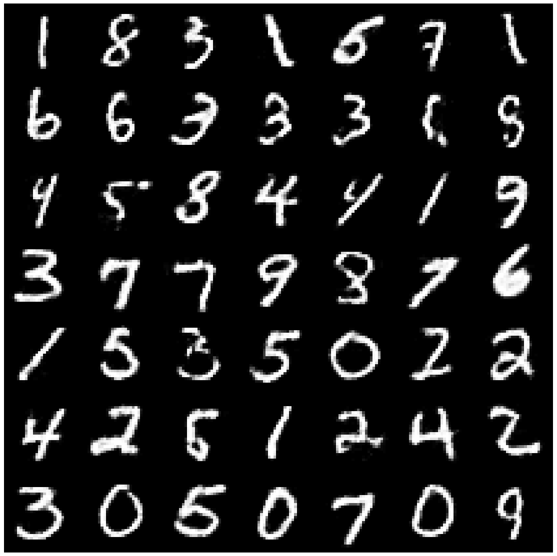
Good Generative Model?

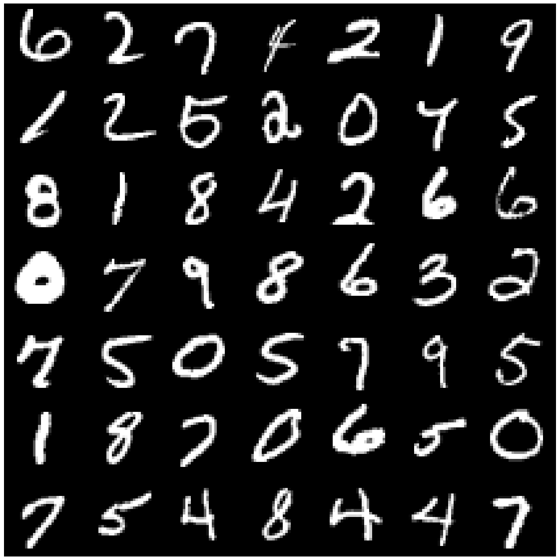
Handwritten Characters



Good Generative Model?

MNIST Handwritten Digit Dataset





Handwriting Recognition

MNIST Dataset
60,000 examples of 10 digits

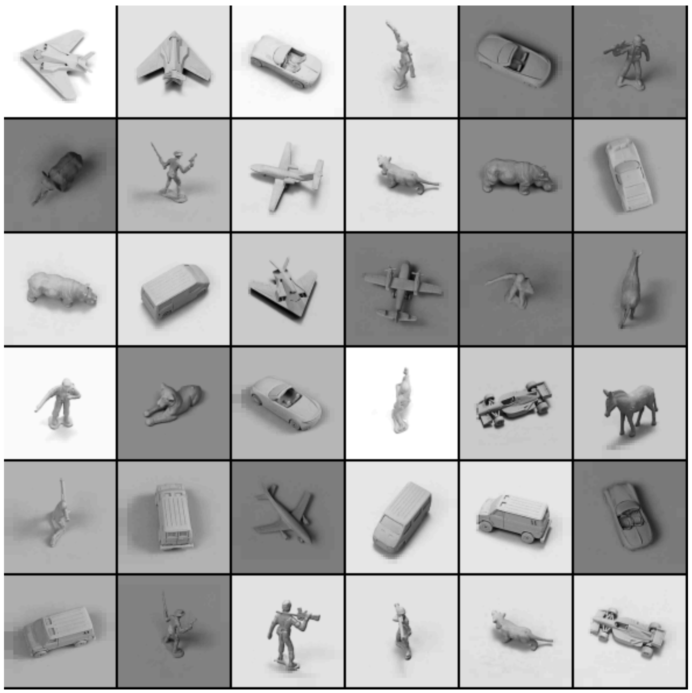
Learning Algorithm	Error
Logistic regression	12.0%
K-NN	3.09%
Neural Net (Platt 2005)	1.53%
SVM (Decoste et.al. 2002)	1.40%
Deep Autoencoder (Bengio et. al. 2007)	1.40%
Deep Belief Net (Hinton et. al. 2006)	1.20%
DBM	**0.95%**

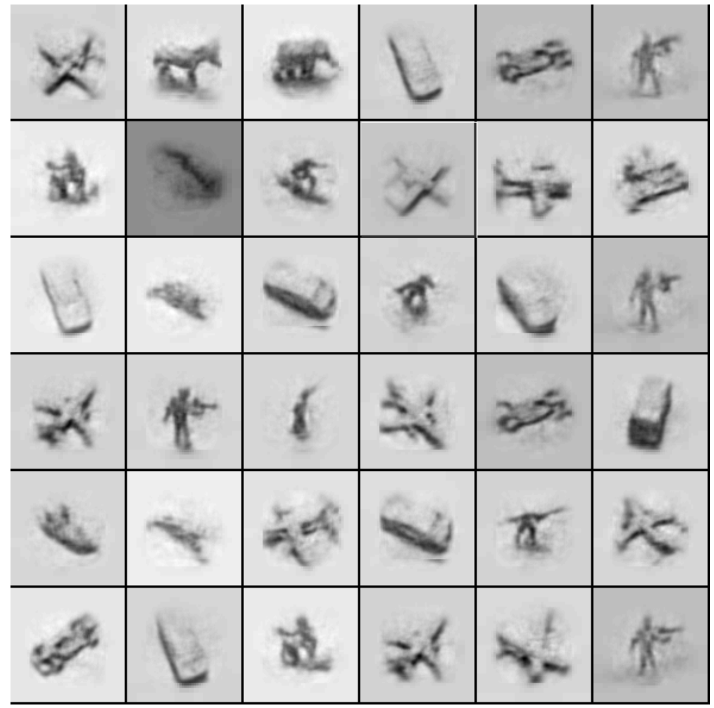
Optical Character Recognition
42,152 examples of 26 English letters

Learning Algorithm	Error
Logistic regression	22.14%
K-NN	18.92%
Neural Net	14.62%
SVM (Larochelle et.al. 2009)	9.70%
Deep Autoencoder (Bengio et. al. 2007)	10.05%
Deep Belief Net (Larochelle et. al. 2009)	9.68%
DBM	**8.40%**

Permutation-invariant version.

Generative Model of 3-D Objects



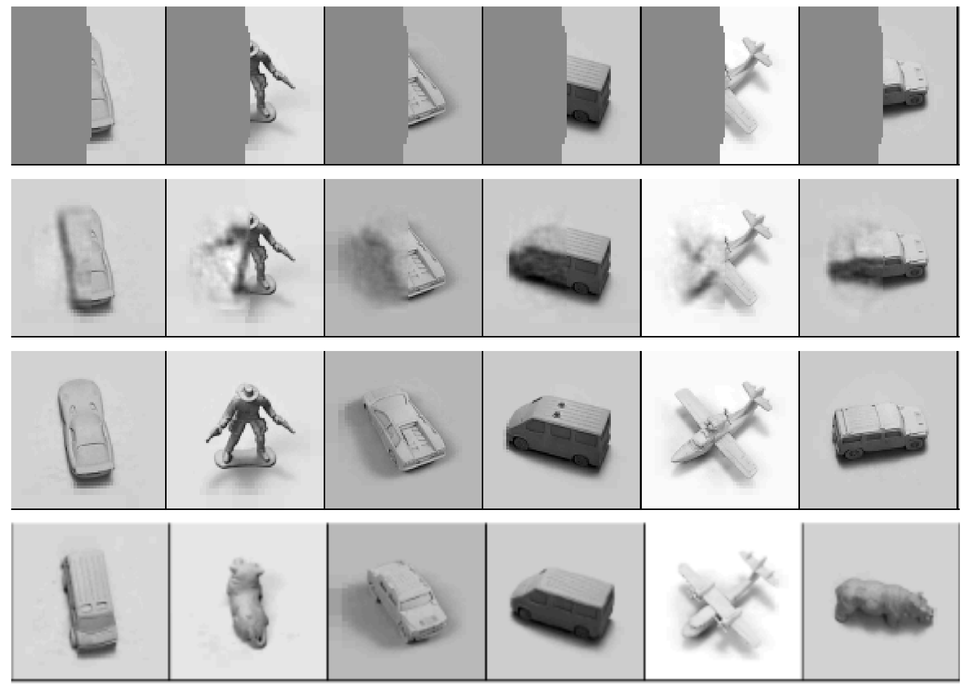


24,000 examples, 5 object categories, 5 different objects within each category, 6 lightning conditions, 9 elevations, 18 azimuths.

3-D Object Recognition

Learning Algorithm	Error
Logistic regression	22.5%
K-NN (LeCun 2004)	18.92%
SVM (Bengio & LeCun 2007)	11.6%
Deep Belief Net (Nair & Hinton 2009)	9.0%
DBM	**7.2%**

Pattern Completion

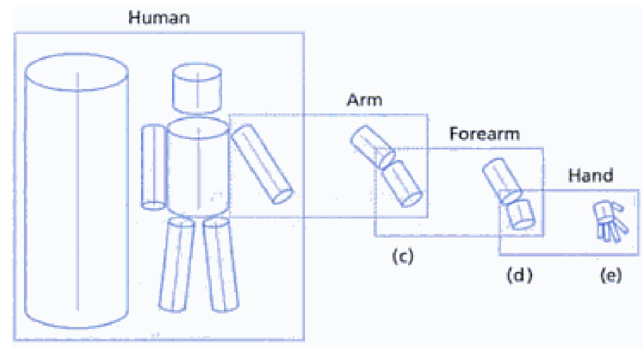


Permutation-invariant version.

Learning Hierarchical Representations

Deep Boltzmann Machines:

Learning Hierarchical Structure in Features: edges, combination of edges.



- Performs well in many application domains
- Combines bottom and top-down
- Fast Inference: fraction of a second
- Learning scales to millions of examples

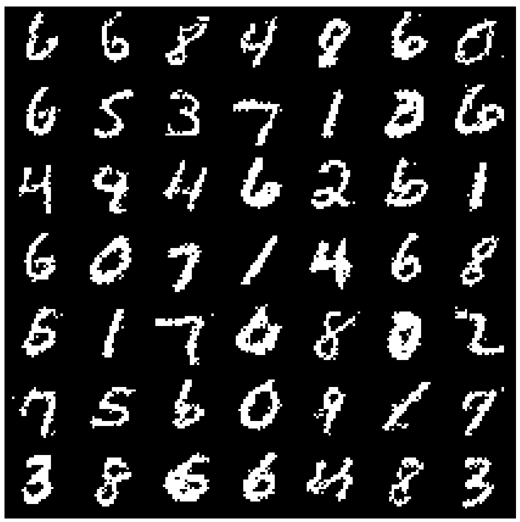
Many examples, few categories

Next: Few examples, many categories – One-Shot Learning

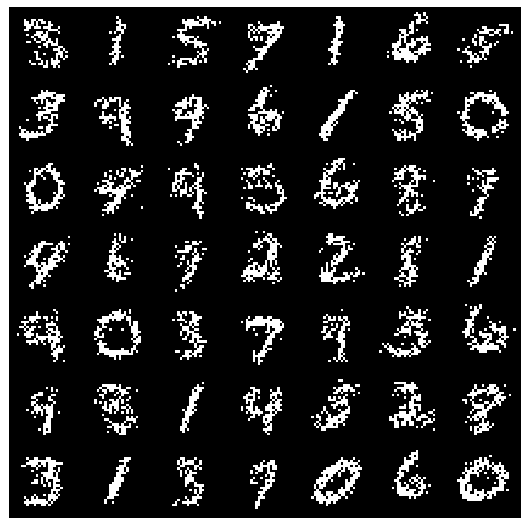
Model Selection

How to choose the number of layers and the number of hidden units?

More generally, how can we choose between models?



DBM samples



Mixture of Bernoulli's

Goal: Compare $P(\mathbf{v})$ on the validation $P(\mathbf{v}) = P(\mathbf{v})^*/\mathcal{Z}$

Need an estimate of Partition Function $\mathcal{Z}$

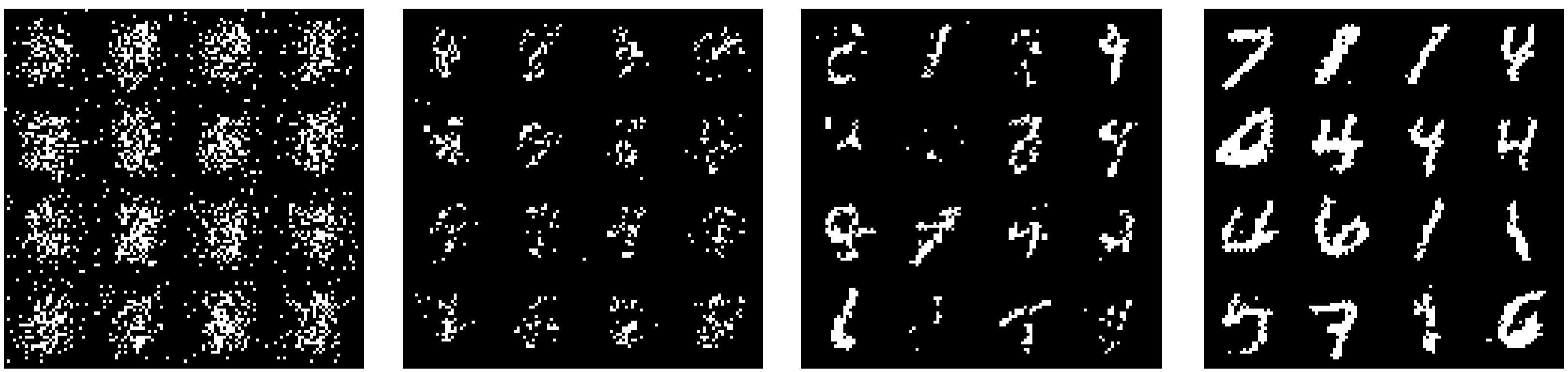
Model Selection

(Salakhutdinov & Murray, ICML 2008, Salakhutdinov 2008)

We have developed an MCMC-based algorithm based on Annealed Importance Sampling to estimate partition function of a DBM model.

$$P_\theta(\mathbf{v};\beta) = \frac{1}{\mathcal{Z}(\beta)} P_\theta(\mathbf{v})^\beta \pi(\mathbf{v})^{(1-\beta)}$$

$\beta = 0$ $\beta = 0.5$ $\beta = 0.8$ $\beta = 1.0$

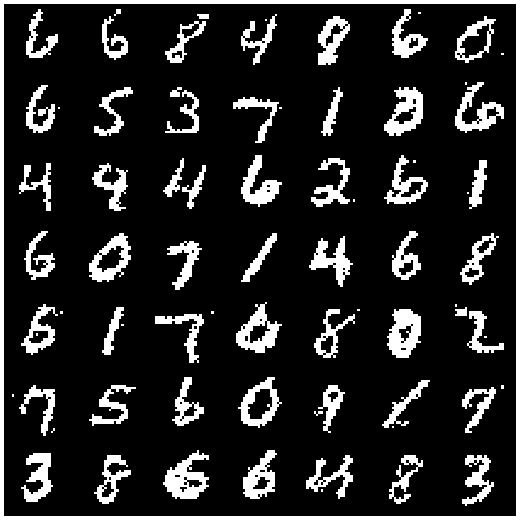


$$\frac{\mathcal{Z}(1)}{\mathcal{Z}(0)} = \frac{\mathcal{Z}(\beta_1)}{\mathcal{Z}(0)} \cdot \frac{\mathcal{Z}(\beta_2)}{\mathcal{Z}(\beta_1)} \cdot \frac{\mathcal{Z}(\beta_3)}{\mathcal{Z}(\beta_2)} \cdot \frac{\mathcal{Z}(\beta_4)}{\mathcal{Z}(\beta_3)} \cdot \frac{\mathcal{Z}(1)}{\mathcal{Z}(\beta_4)}$$

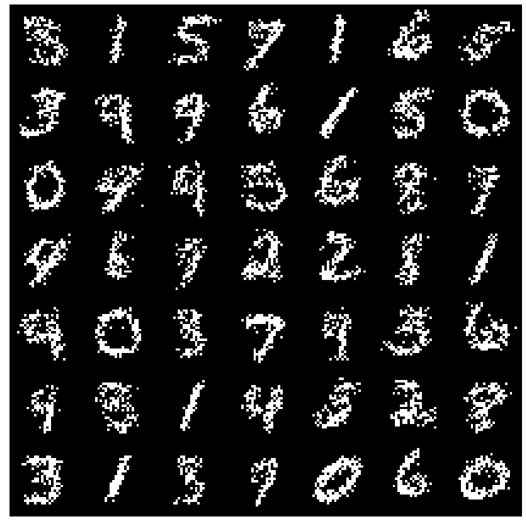
Annealing, or Tempering: $1/\beta =$ "temperature"

Model Selection

(Salakhutdinov & Murray, ICML 2008, Salakhutdinov 2008)



DBM samples



Mixture of Bernoulli's

MoB, test log-probability: -137.64 nats/digit

DBM, test log-probability: -85.97 nats/digit

Difference of over 50 nats is striking!

Thank you

Code for learning RBMs, DBNs, and DBMs is available at:
http://www.utstat.toronto.edu/~rsalakhu/