Deep Learning

Russ Salakhutdinov

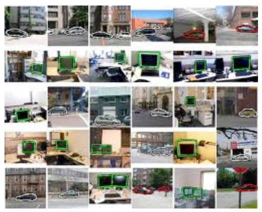
Department of Statistics and Computer Science
University of Toronto

Slides available at
http://www.utstat.toronto.edu/~rsalakhu/isbi.html

Mining for Structure

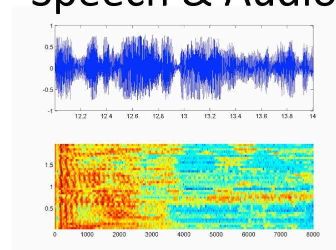
Massive increase in both computational power and the amount of data available from web, video cameras, laboratory measurements.

Images & Video

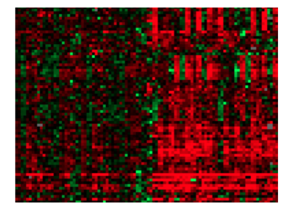


Text & Language

Speech & Audio

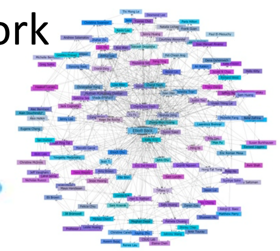


Gene Expression

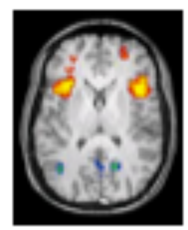


Product Recommendation

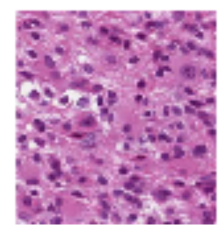
Relational Data/ Social Network



fMRI



Tumor region



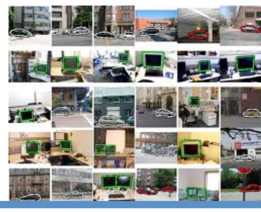
Mostly Unlabeled

- 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

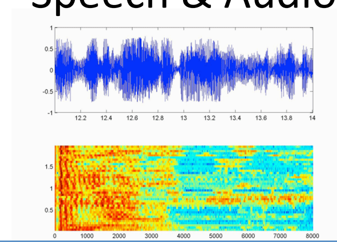
Massive increase in both computational power and the amount of data available from web, video cameras, laboratory measurements.

Images & Video

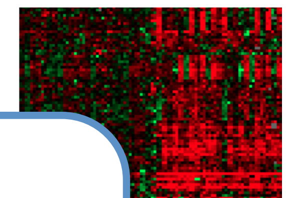


Text & Language

Speech & Audio



Gene Expression



Product Recomm

on

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

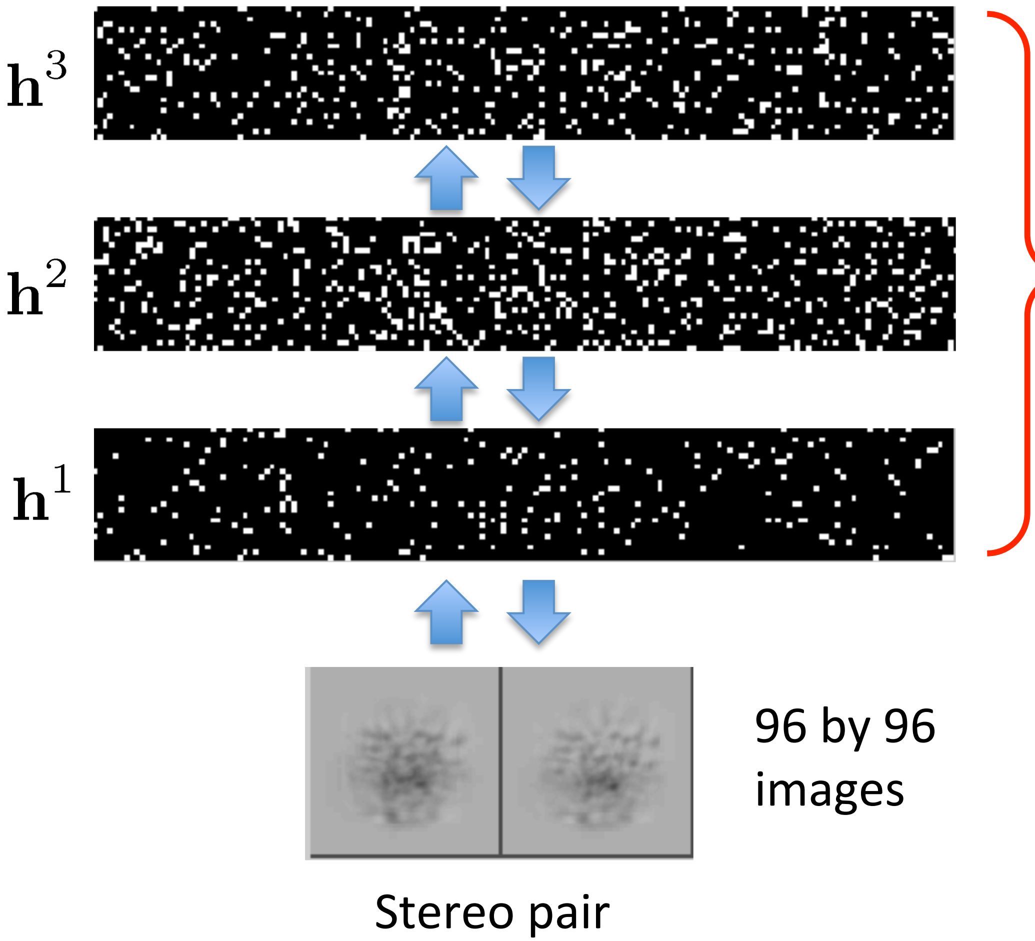
Mostly Unlabeled

- 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

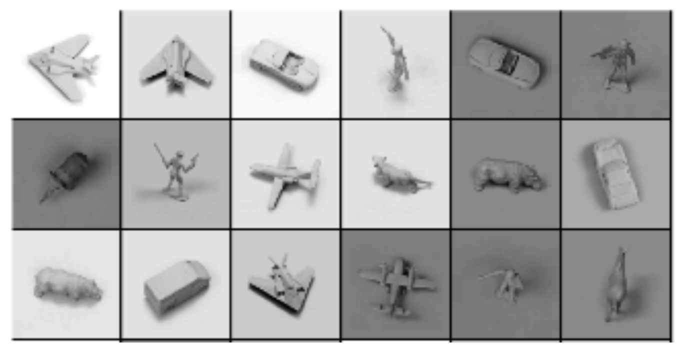
Deep Boltzmann Machine

Gaussian-Bernoulli Markov Random Field



12,000 Latent Variables

Model P(image)



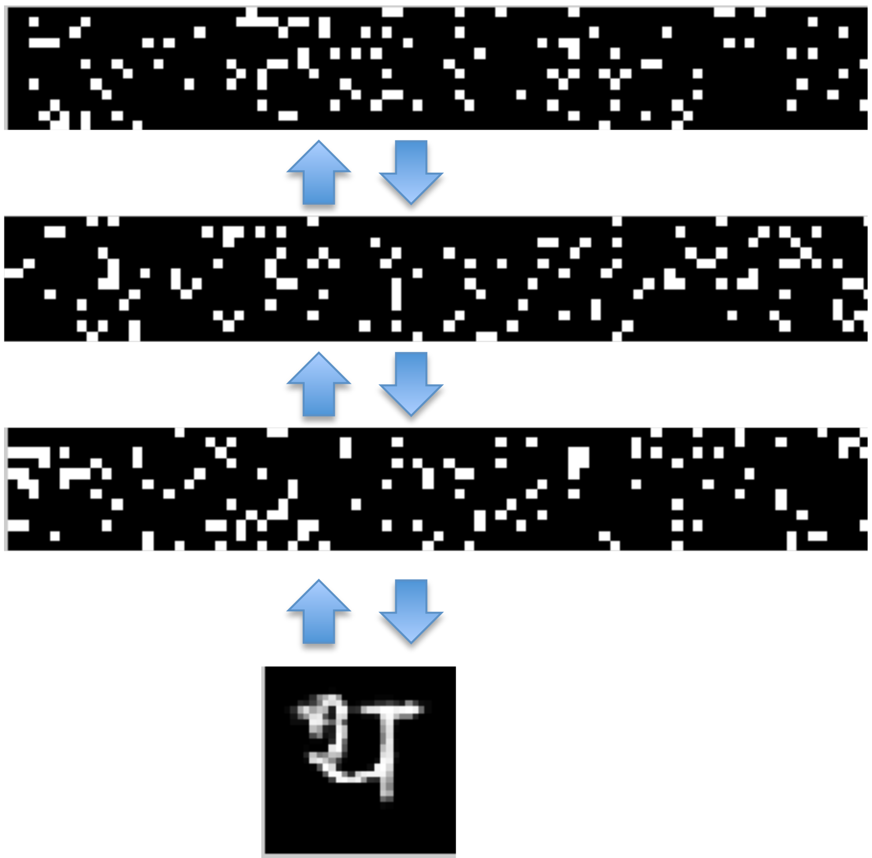
24,000 Training Images

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

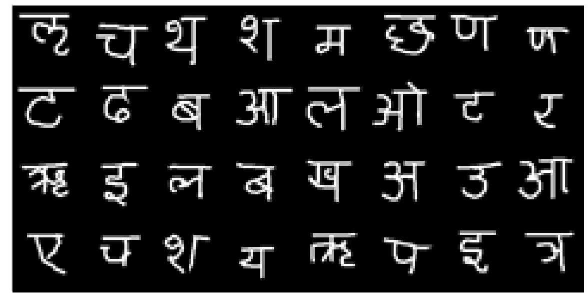
Deep Generative Model

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

Sanskrit



Model P(image)



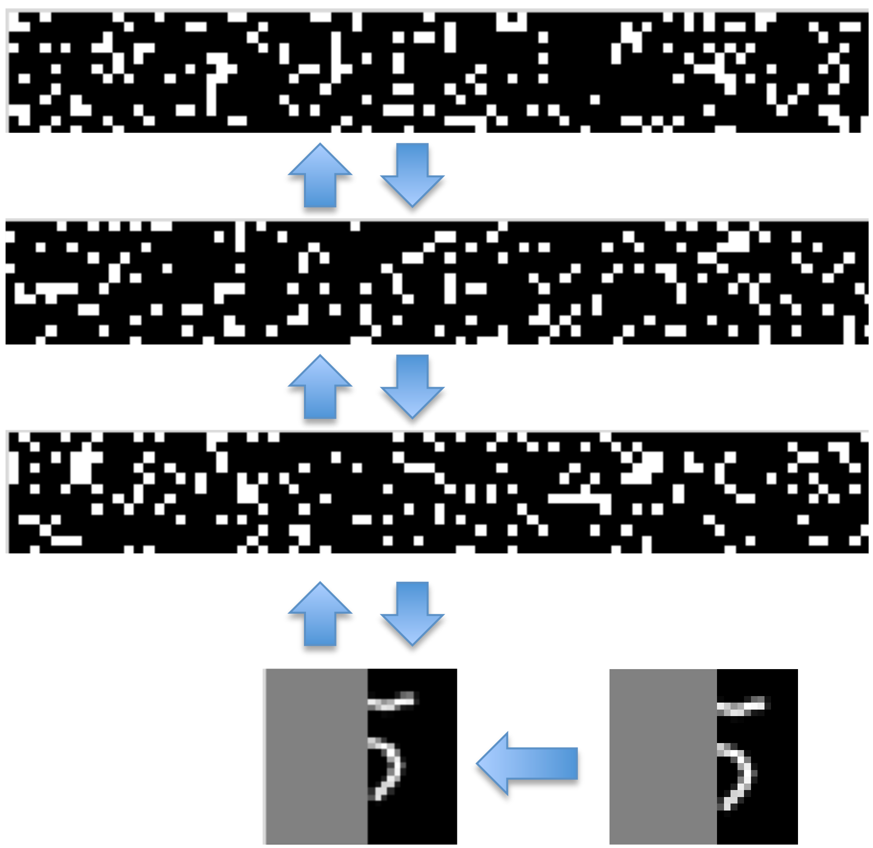
25,000 characters from 50 alphabets around the world.

- 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)



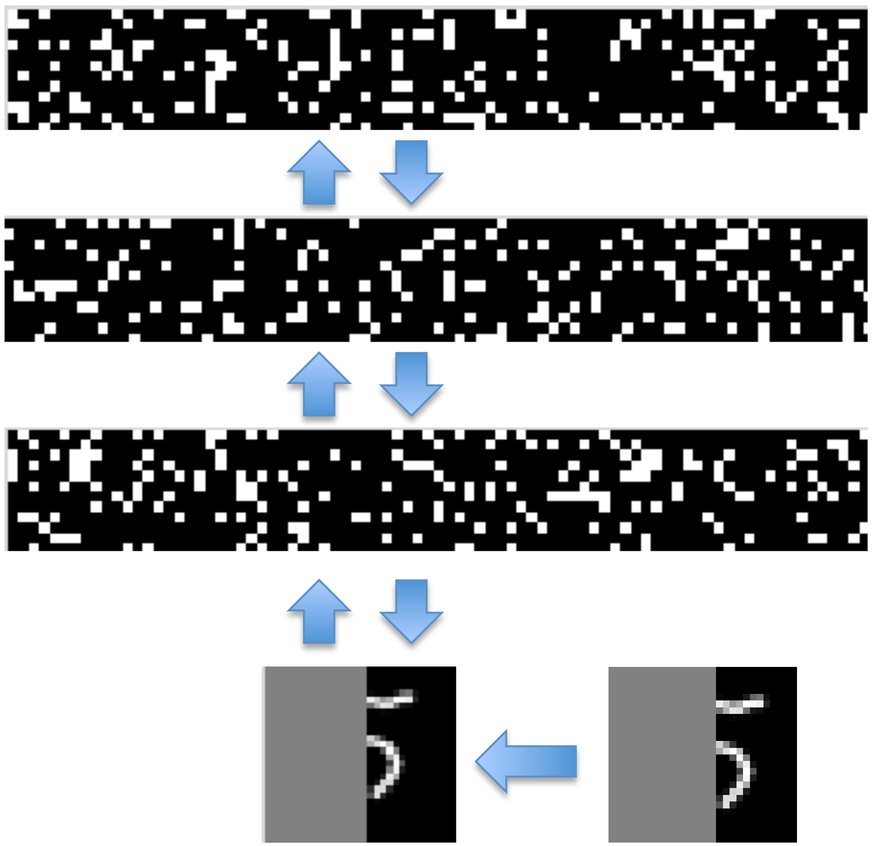
Conditional Simulation

P(image|partial image)

Bernoulli Markov Random Field

Deep Generative Model

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



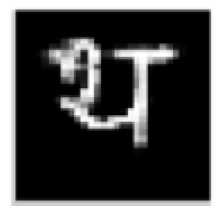
P(image|partial image)

Conditional Simulation

Why so difficult?

28

28



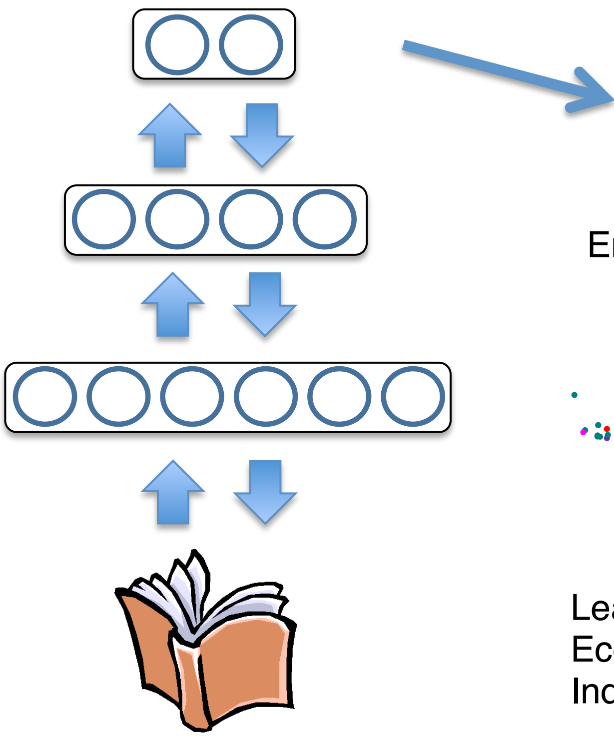
$2^{28 \times 28}$ possible images!

$\gg$ number of particles in the universe

Bernoulli Markov Random Field

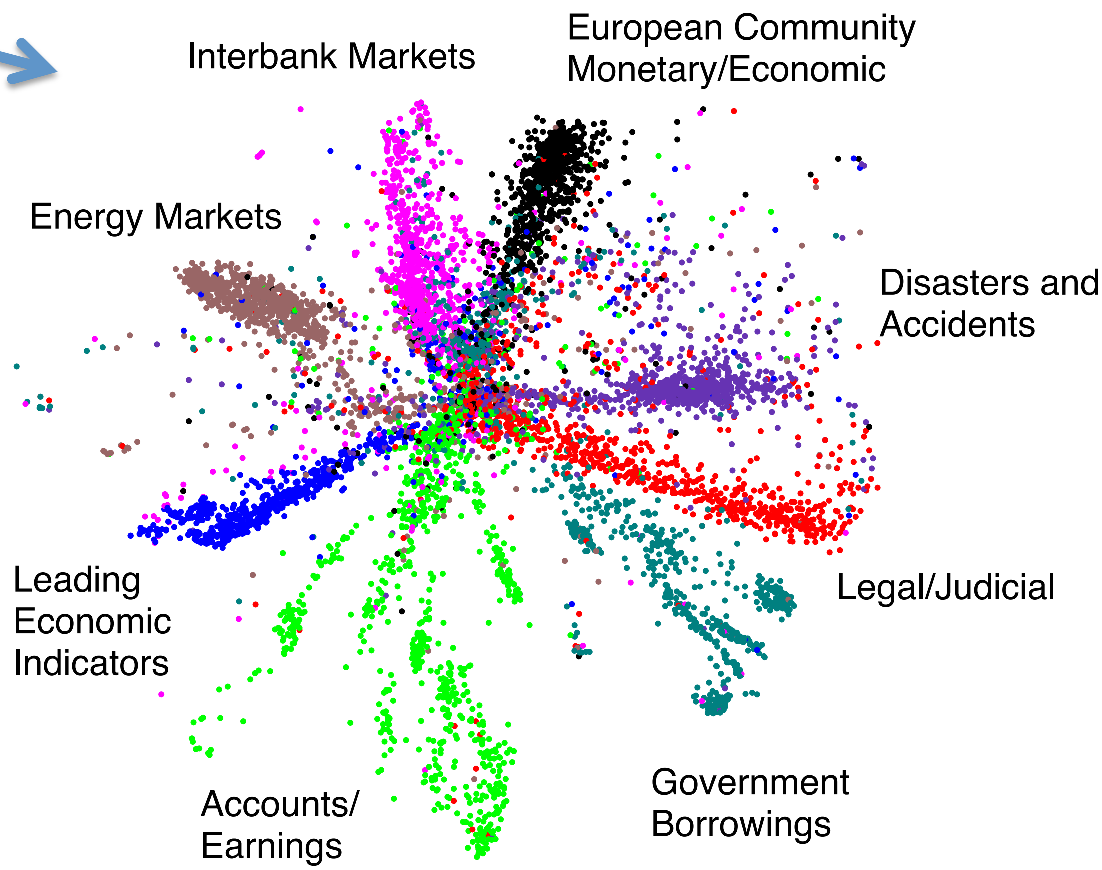
Deep Generative Model

Model P(document)



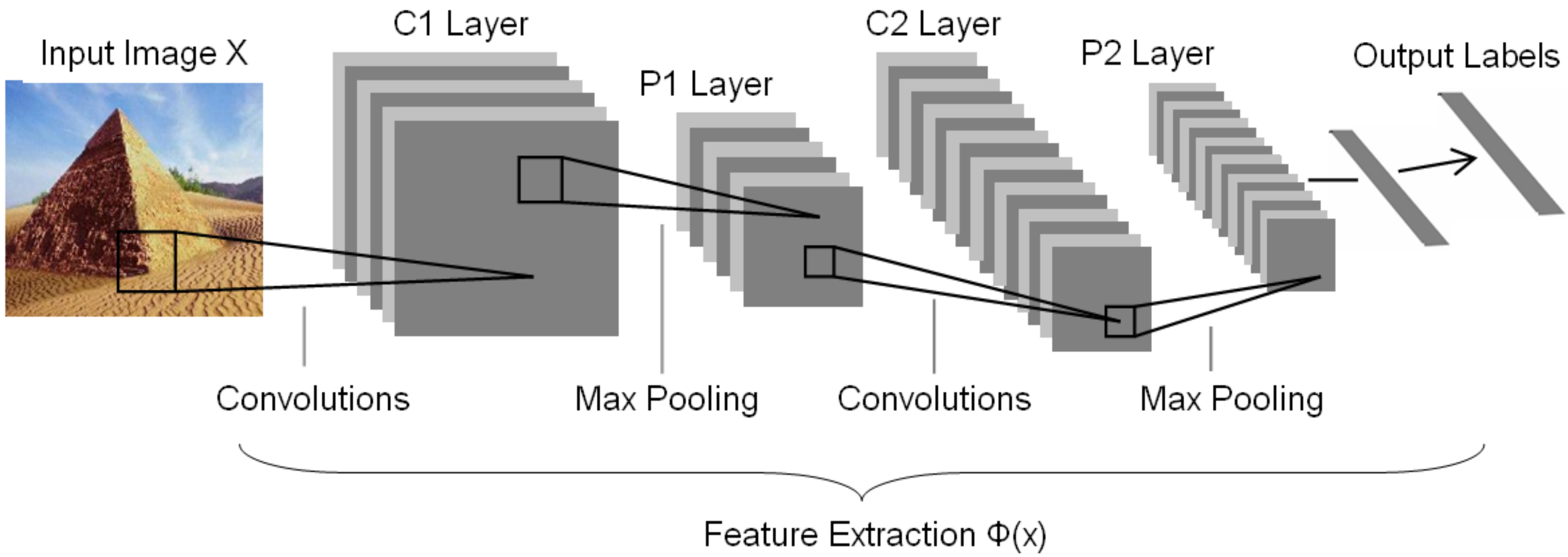
Bag of words

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



(Hinton & Salakhutdinov, Science 2006)

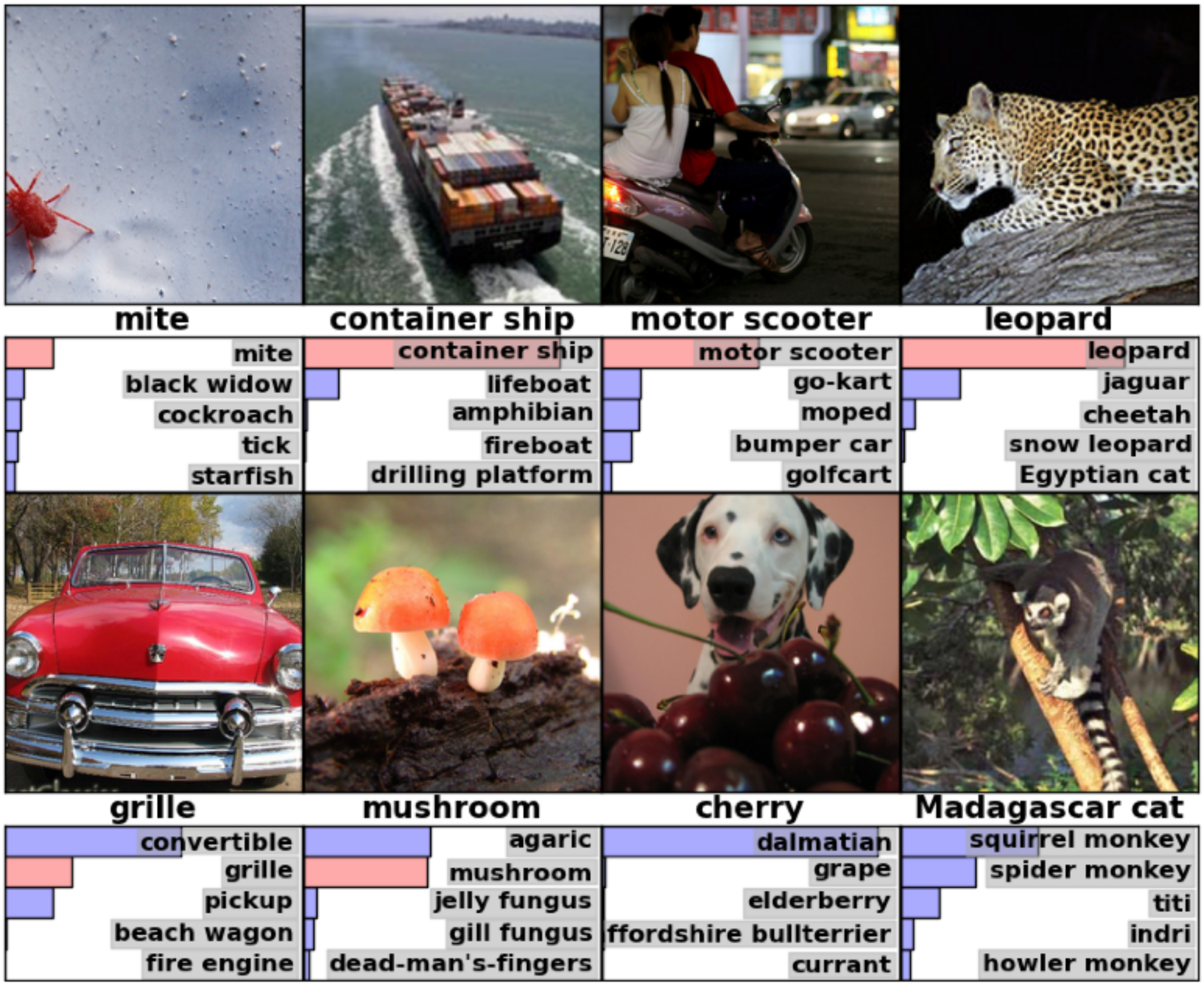
Convolutinal Deep Models for Image Recognition



- Learning multiple layers of representation.

(LeCun, 1992)

Convolutinal Deep Models for Image Recognition



(Krizhevsky et. al., NIPS 2012)

Predicting Roads from Satellite Images

(Mnih and Hinton, ICML 2012)

Predicting Roads from Satellite Images

(Mnih and Hinton, ICML 2012)

Talk Roadmap

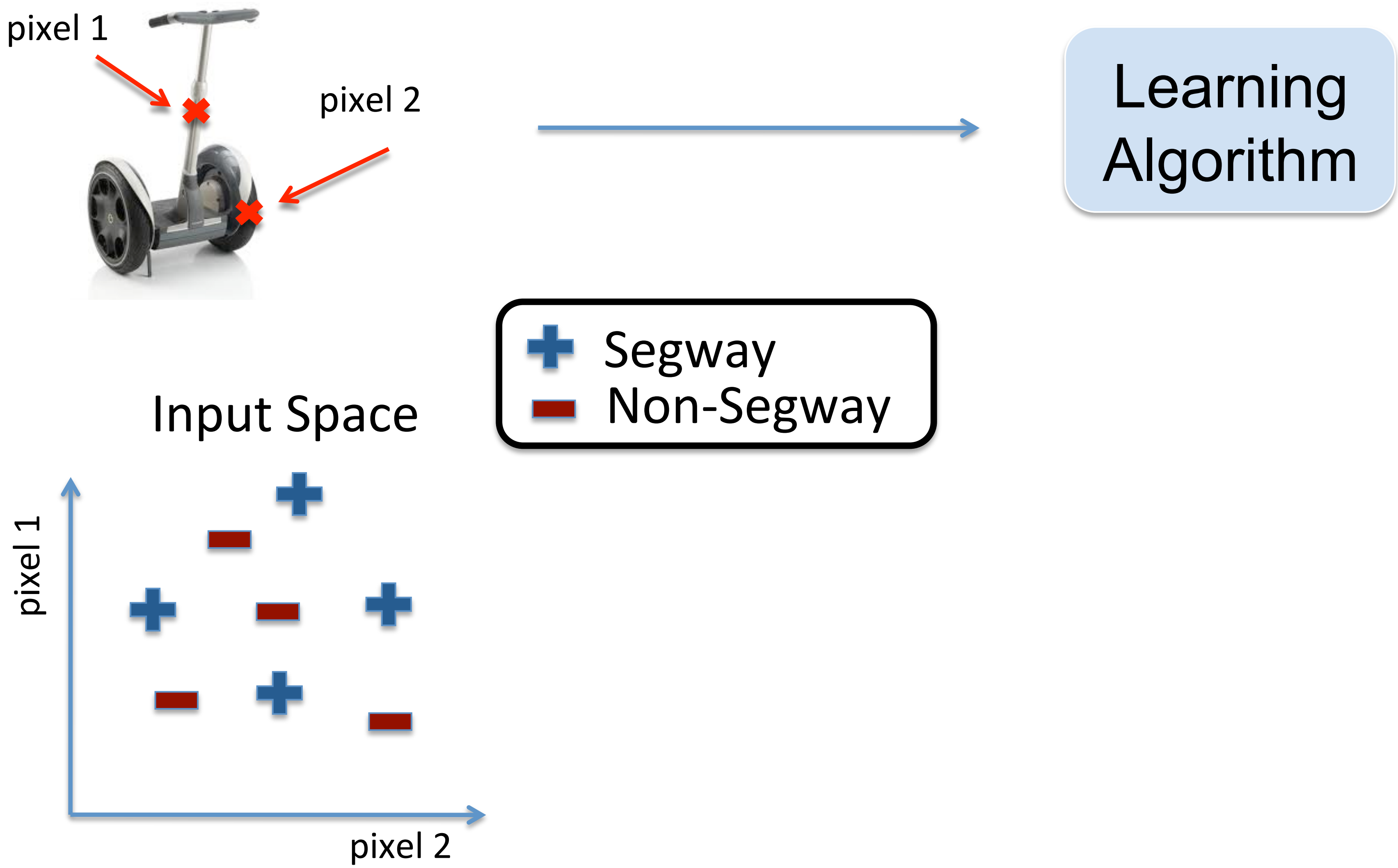
Part 1: Deep Networks

- Introduction, Sparse Coding, Autoencoders.
- Introduction to Graphical models
- Restricted Boltzmann Machines: Learning low-level features.
- Deep Belief Networks: Learning Part-based Hierarchies.

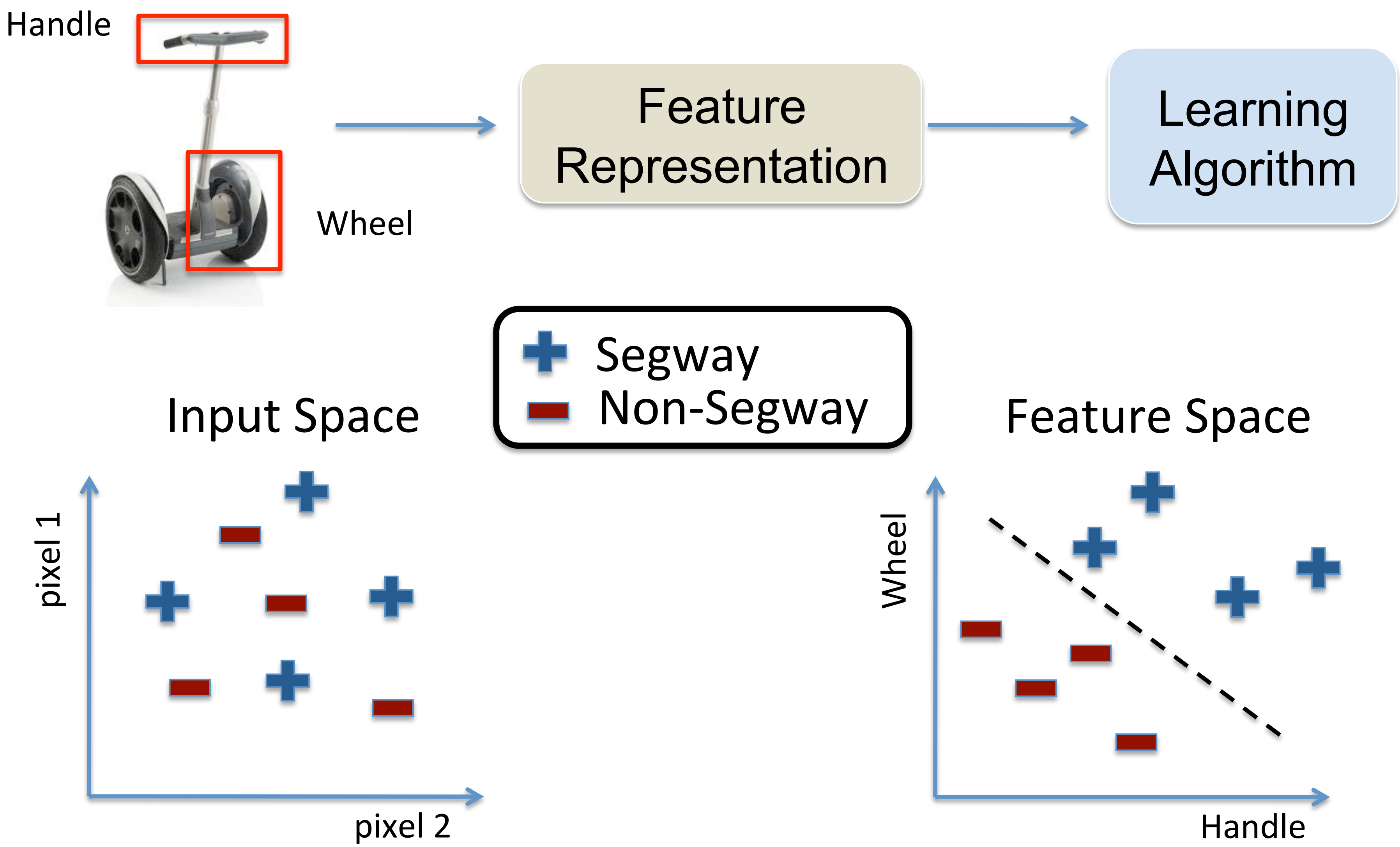
Part 2: Advanced Deep Models.

- Deep Boltzmann Machines
- Multimodal Learning

Learning Feature Representations



Learning Feature Representations

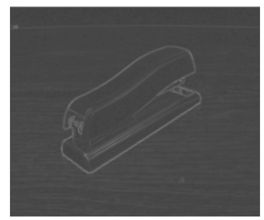


How is computer perception done?

Input Data → Low-level features → Learning Algorithm

Object detection

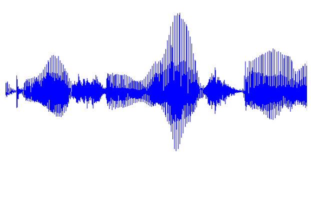
Image



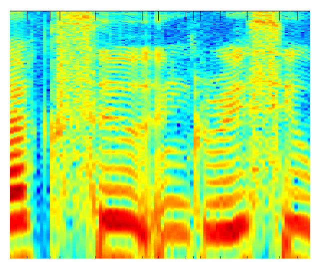
Low-level vision features

Recognition

Audio classification



Audio

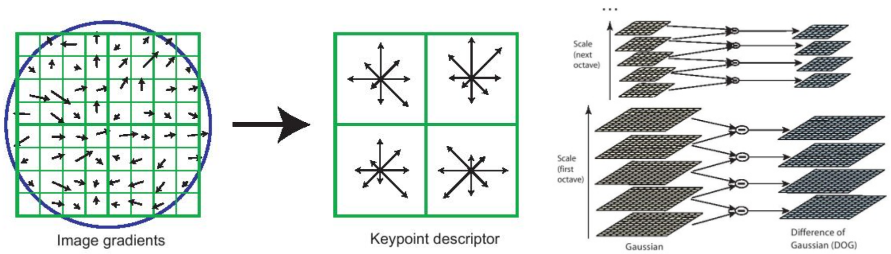


Low-level audio features

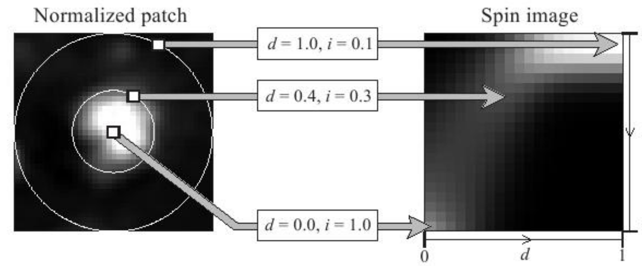
Speaker identification

Slide Credit: Honglak Lee

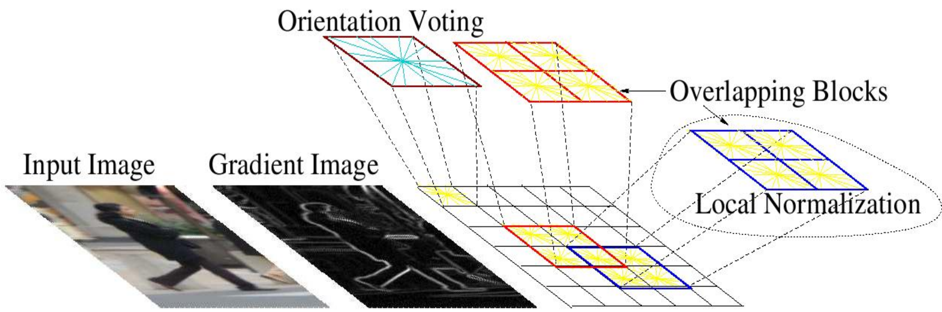
Computer vision features



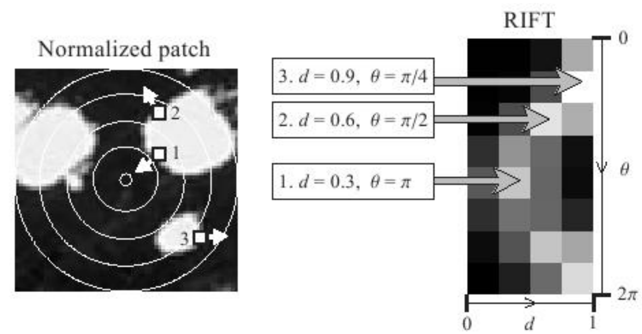
SIFT



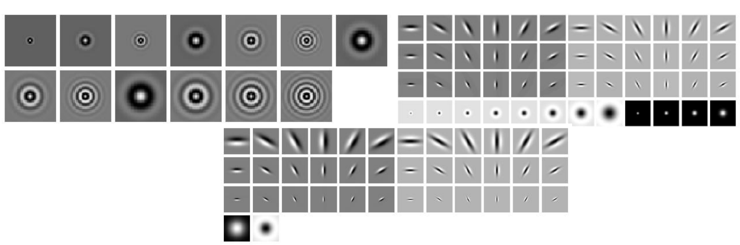
Spin image



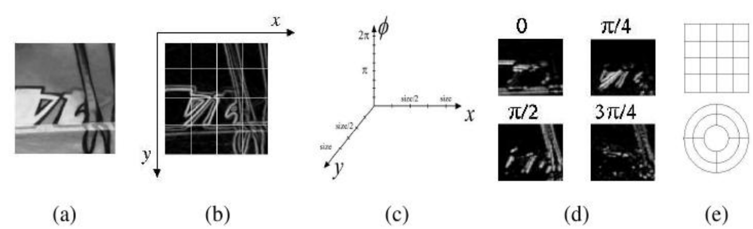
HoG



RIFT



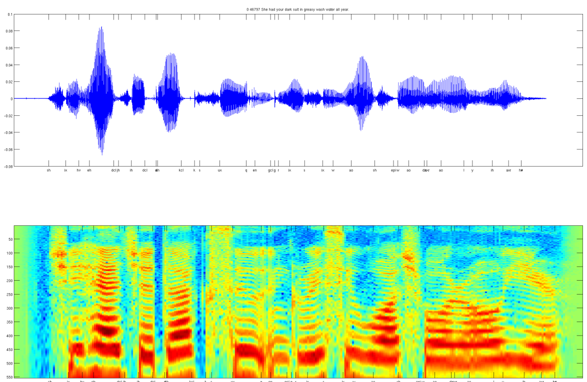
Textons



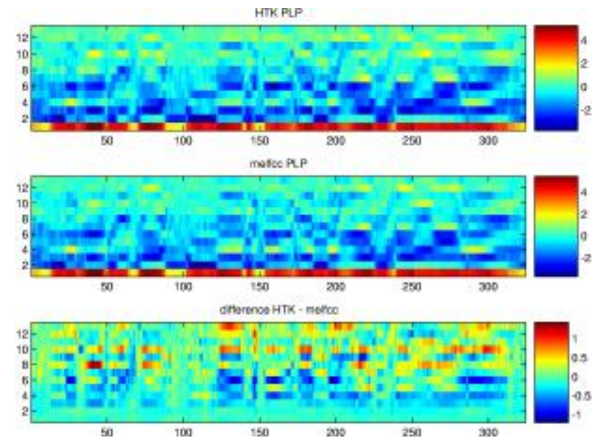
GLOH

Slide Credit: Honglak Lee

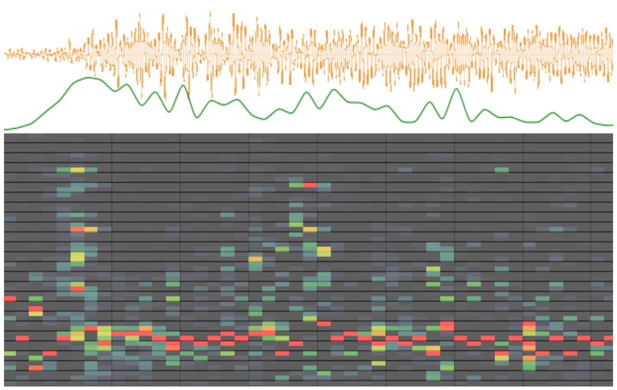
Audio features



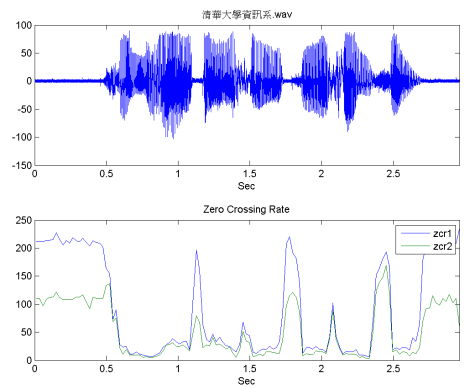
Spectrogram



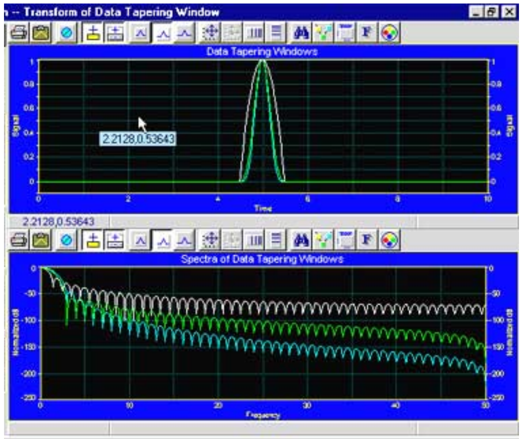
MFCC



Flux

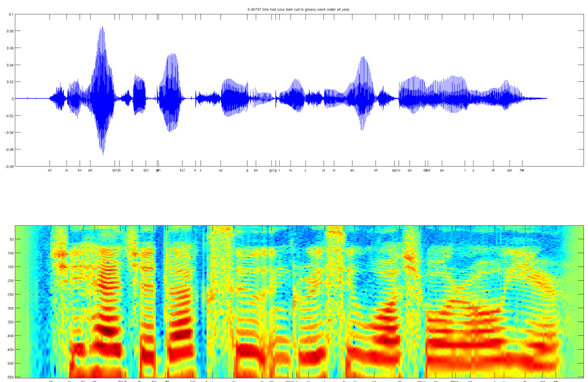


ZCR

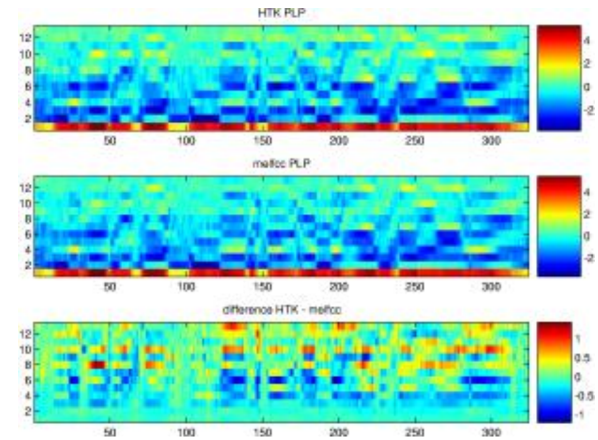


Rolloff

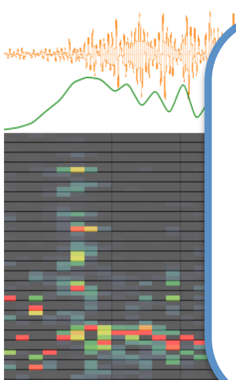
Audio features

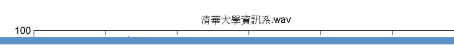


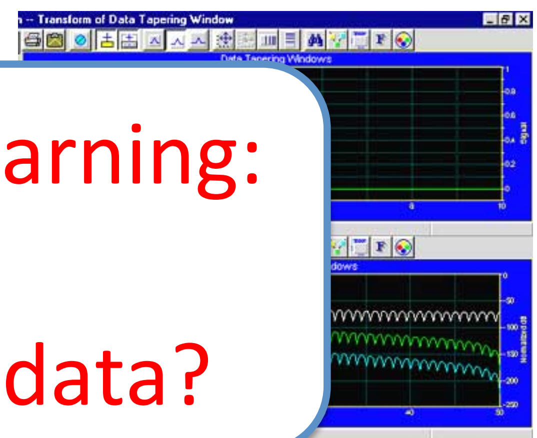
Spectrogram



MFCC







Unsupervised Feature Learning:
Can we learn meaningful
features from unlabeled data?

Flux

ZCR

Rolloff

Sparse Coding

- Sparse coding (Olshausen & Field, 1996). Originally developed to explain early visual processing in the brain (edge detection).

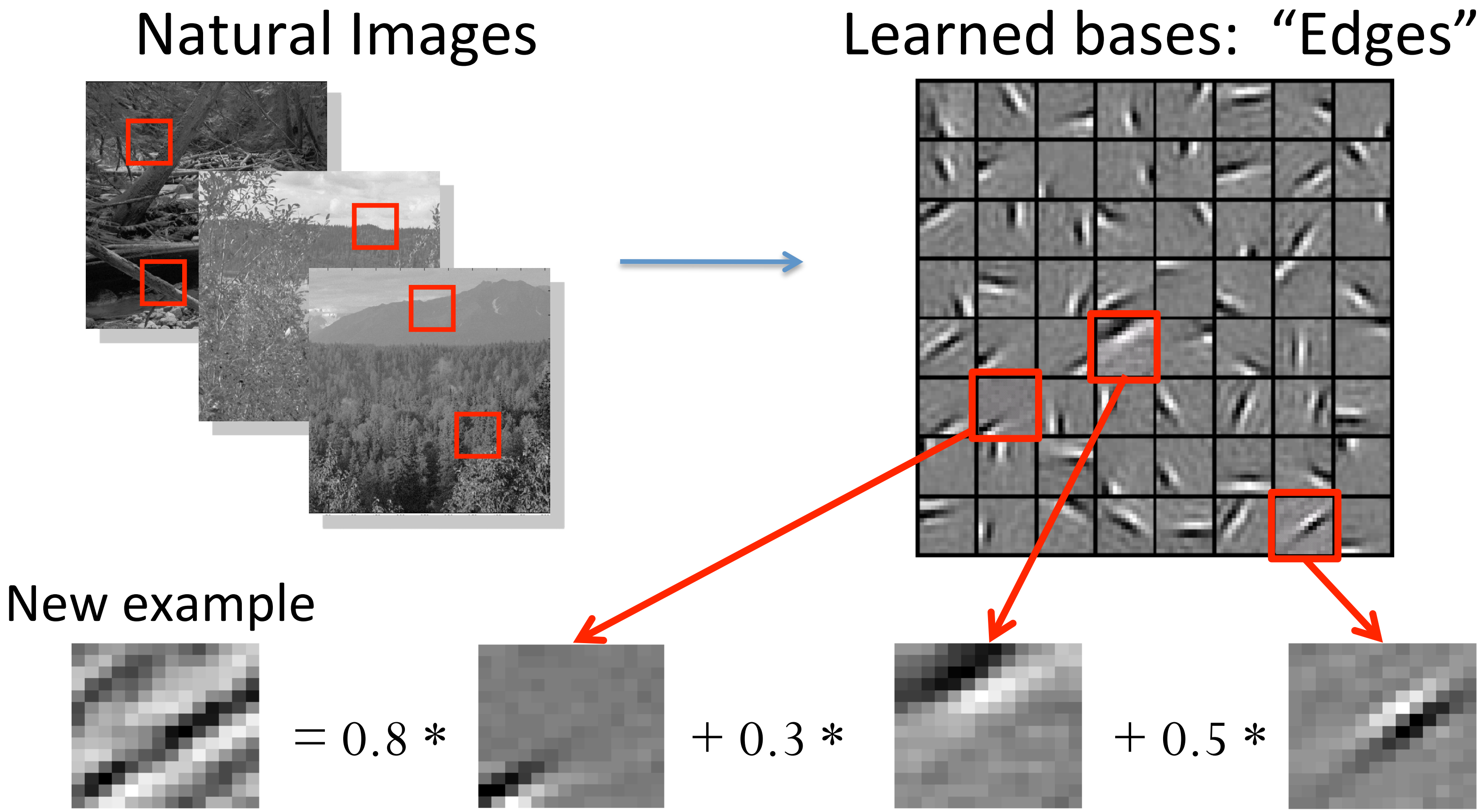
- Objective: Given a set of input data vectors $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, learn a dictionary of bases $\{\phi_1, \phi_2, ..., \phi_K\}$, such that:

$$\mathbf{x}_n = \sum_{k=1}^{K} a_{nk} \phi_k,$$

Sparse: mostly zeros

- Each data vector is represented as a sparse linear combination of bases.

Sparse Coding



$$x = 0.8 * \phi_{36} + 0.3 * \phi_{42} + 0.5 * \phi_{65}$$

[0, 0, … **0.8**, …, **0.3**, …, **0.5**, …] = coefficients (feature representation)

Slide Credit: Honglak Lee

Sparse Coding: Training

- Input image patches: $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N \in \mathbb{R}^D$
- Learn dictionary of bases: $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, ..., \boldsymbol{\phi}_K \in \mathbb{R}^D$

$$\min_{\mathbf{a},\boldsymbol{\phi}} \underbrace{\sum_{n=1}^{N} \left\| \mathbf{x}_n - \sum_{k=1}^{K} a_{nk} \boldsymbol{\phi}_k \right\|_2^2}_{\text{Reconstruction error}} + \lambda \underbrace{\sum_{n=1}^{N} \sum_{k=1}^{K} |a_{nk}|}_{\text{Sparsity penalty}}$$

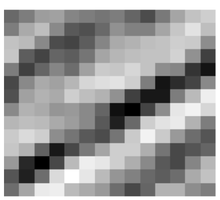
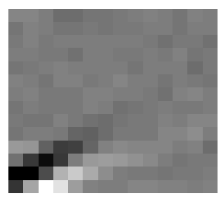
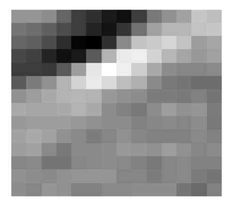
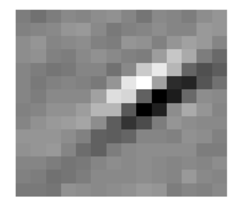
- Alternating Optimization:

1. Fix dictionary of bases $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, ..., \boldsymbol{\phi}_K$ and solve for activations **a** (a standard Lasso problem).
2. Fix activations **a**, optimize the dictionary of bases (convex QP problem).

Sparse Coding: Testing Time

- Input: a new image patch x* , and K learned bases $\phi_1, \phi_2, ..., \phi_K$
- Output: sparse representation **a** of an image patch x*.

$$\min_{\mathbf{a}} \left\| \mathbf{x}^* - \sum_{k=1}^{K} a_k \phi_k \right\|_2^2 + \lambda \sum_{k=1}^{K} |a_k|$$

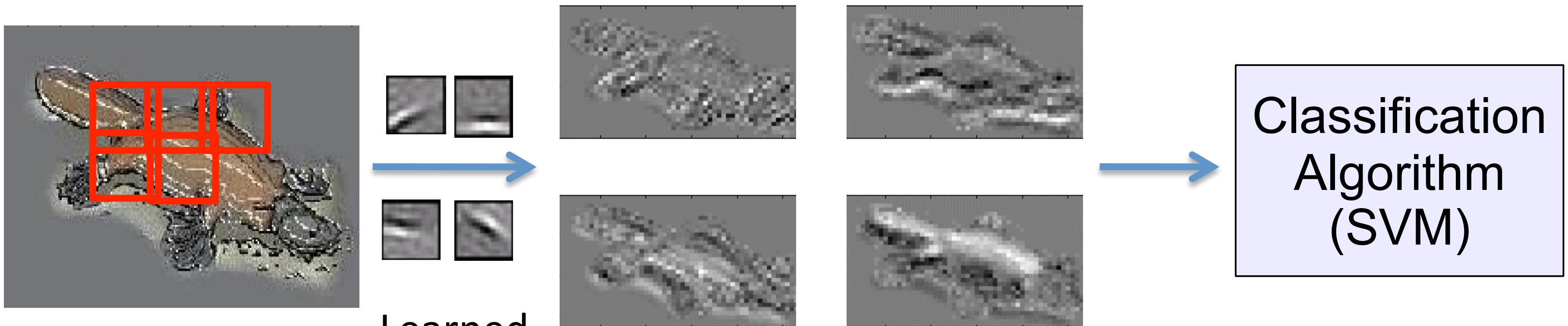
 $= 0.8 *$  $+ 0.3 *$  $+ 0.5 *$ 

$$x^* = 0.8 * \phi_{36} + 0.3 * \phi_{42} + 0.5 * \phi_{65}$$

[0, 0, … **0.8**, …, **0.3**, …, **0.5**, …] = coefficients (feature representation)

Image Classification

Evaluated on Caltech101 object category dataset.

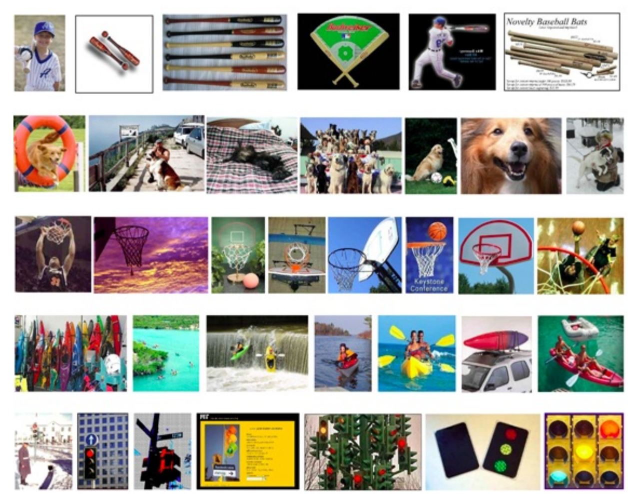


Input Image

Learned bases

Features (coefficients)

9K images, 101 classes

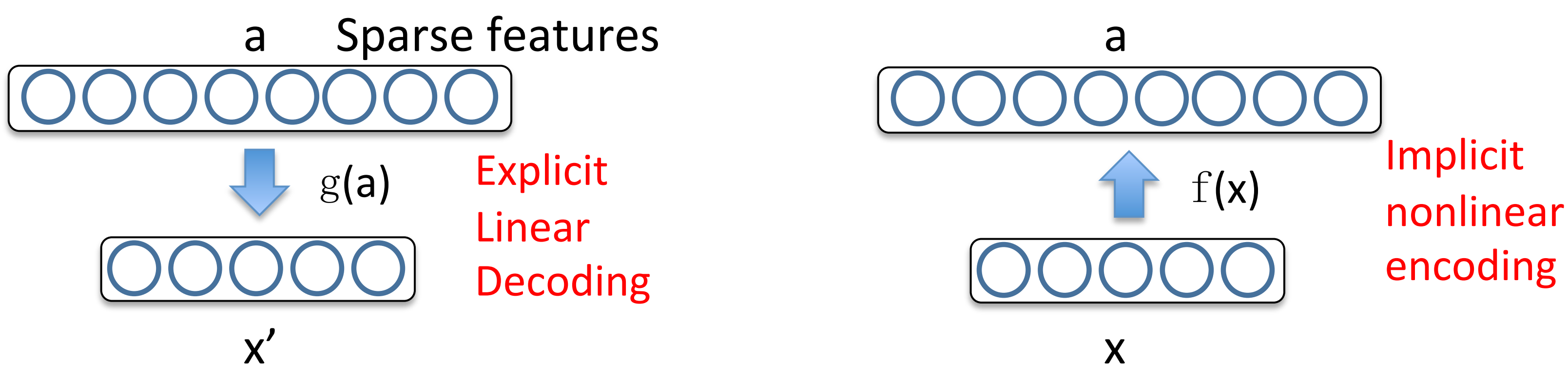


Algorithm	Accuracy
Baseline (Fei-Fei et al., 2004)	16%
PCA	37%
Sparse Coding	**47%**

Lee et al., NIPS 2006

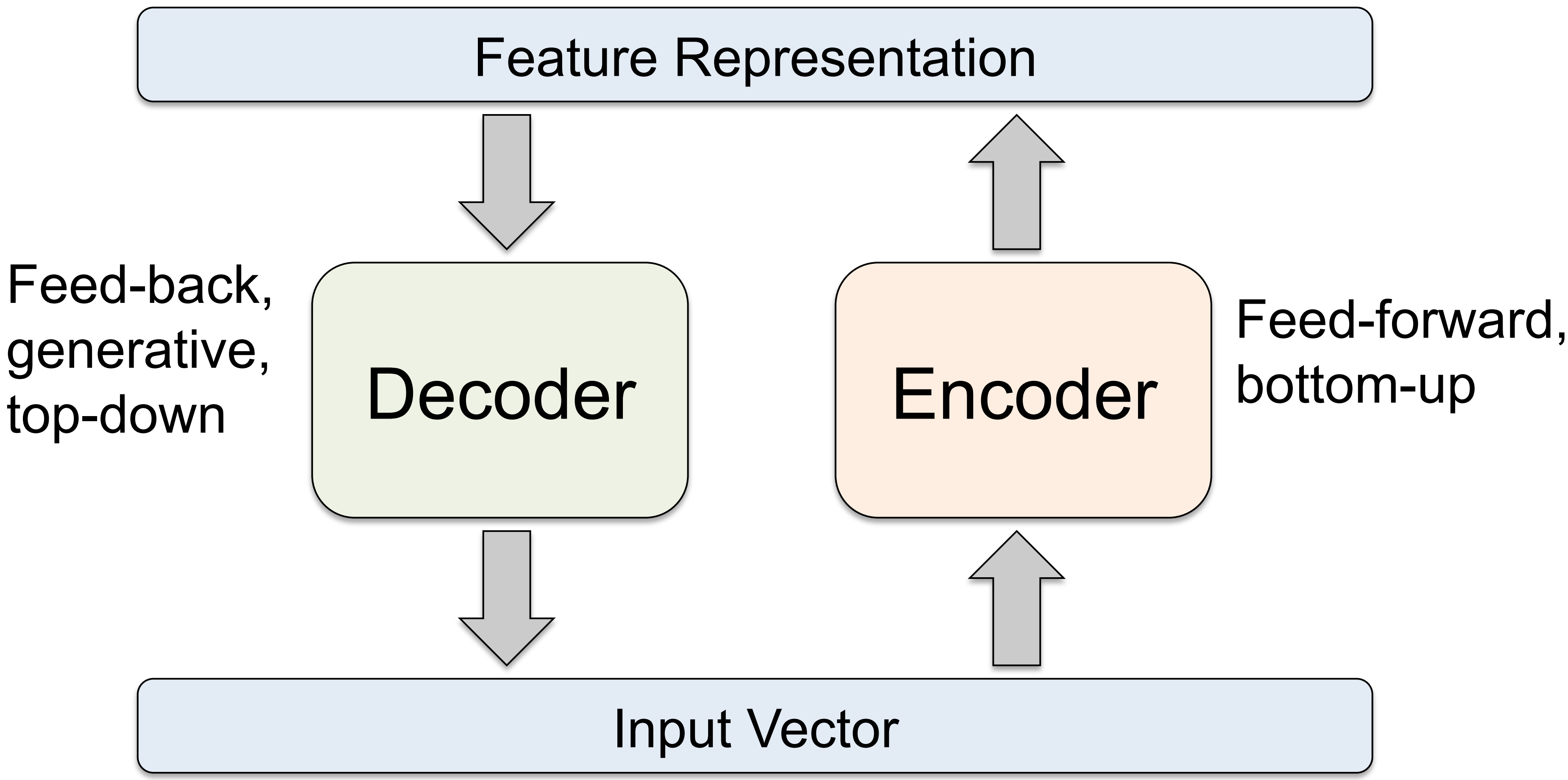
Interpreting Sparse Coding

$$\min_{\mathbf{a},\boldsymbol{\phi}} \sum_{n=1}^{N} \left\| \mathbf{x}_n - \sum_{k=1}^{K} a_{nk} \phi_k \right\|_2^2 + \lambda \sum_{n=1}^{N} \sum_{k=1}^{K} |a_{nk}|$$



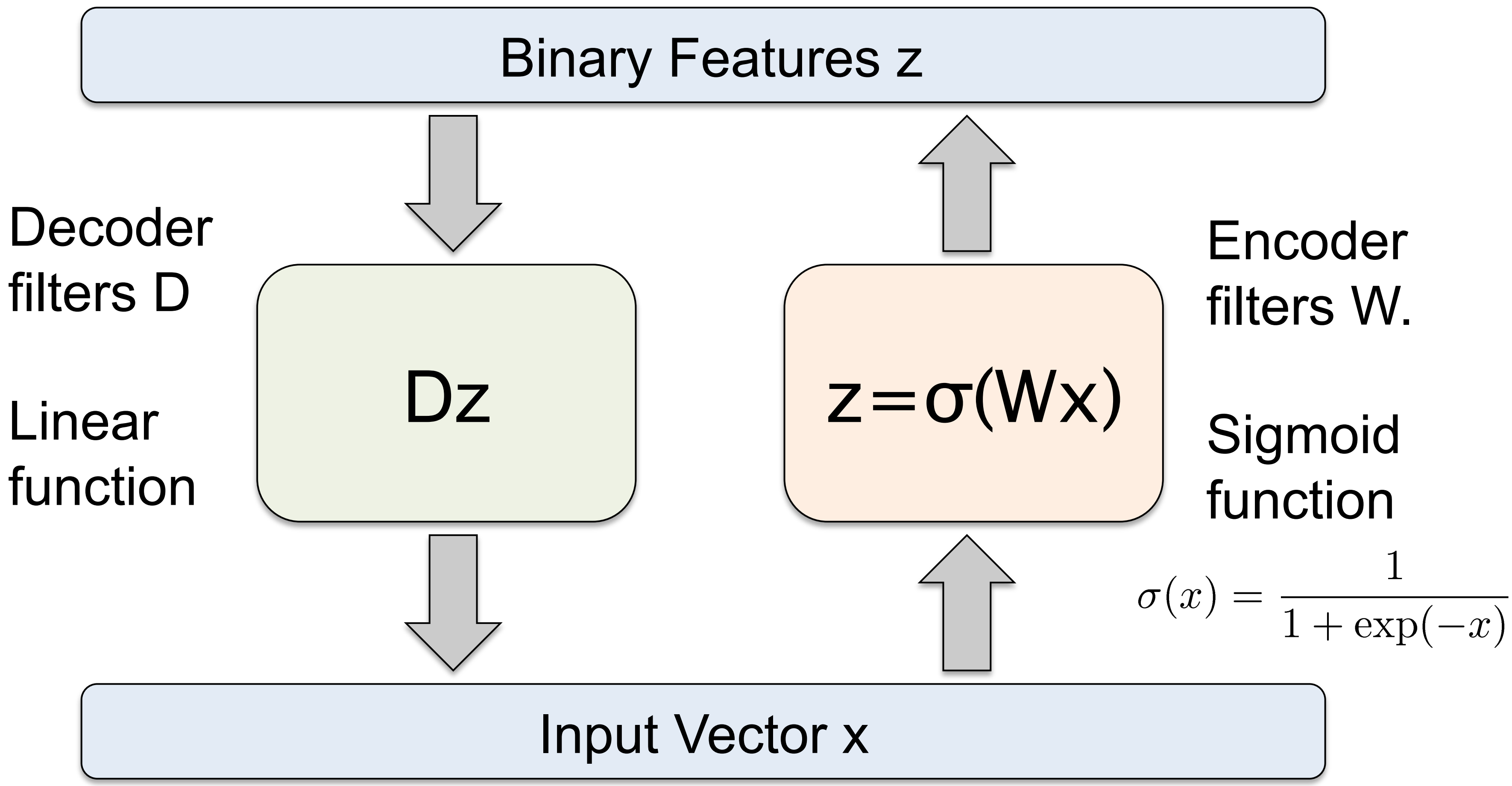
- Sparse, over-complete representation **a.**
- Encoding **a** = f(**x**) is implicit and nonlinear function of **x**.
- Reconstruction (or decoding) **x'** = g(**a**) is linear and explicit.

Autoencoder

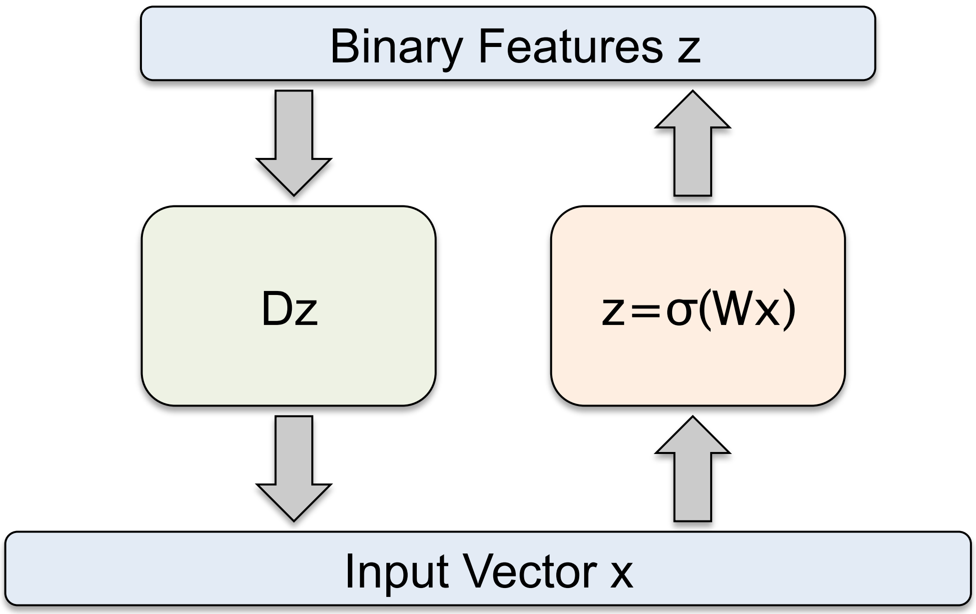


- Details of what goes insider the encoder and decoder matter!
- Need constraints to avoid learning an identity.

Autoencoder



Autoencoder



- An autoencoder with D inputs, D outputs, and K hidden units, with K<D.

- Given an input x, its reconstruction is given by:

$$y_j(\mathbf{x}, W, D) = \underbrace{\sum_{k=1}^{K} D_{jk}}_{\text{Decoder}} \sigma \underbrace{\left(\sum_{i=1}^{D} W_{ki} x_i \right)}_{\text{Encoder}}, \quad j = 1, .., D.$$

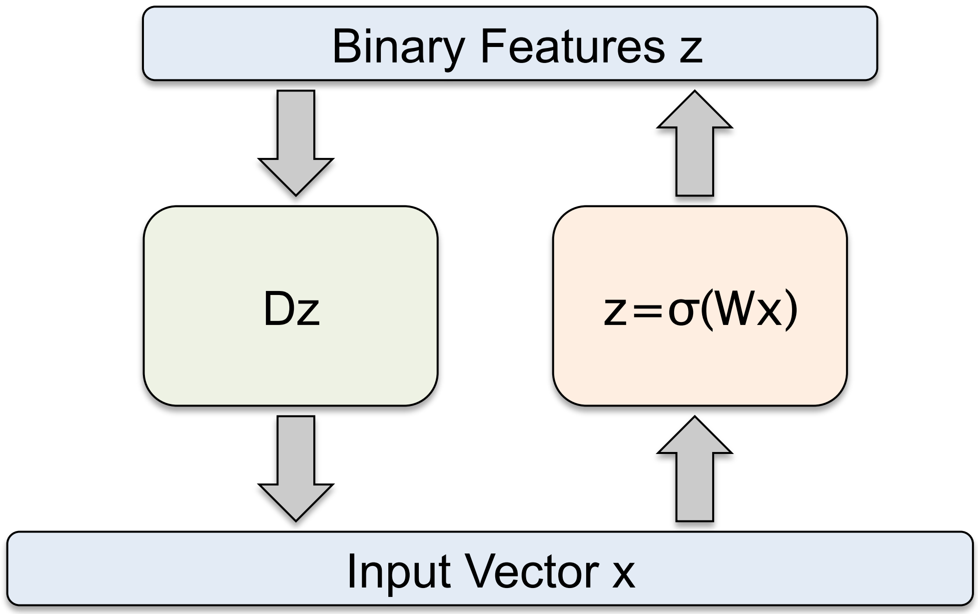
Decoder

$$y_j = \sum_{k=1}^{K} D_{jk} z_k$$

Encoder

$$z_k = \sigma \left(\sum_{i=1}^{D} W_{ki} x_i \right)$$

Autoencoder

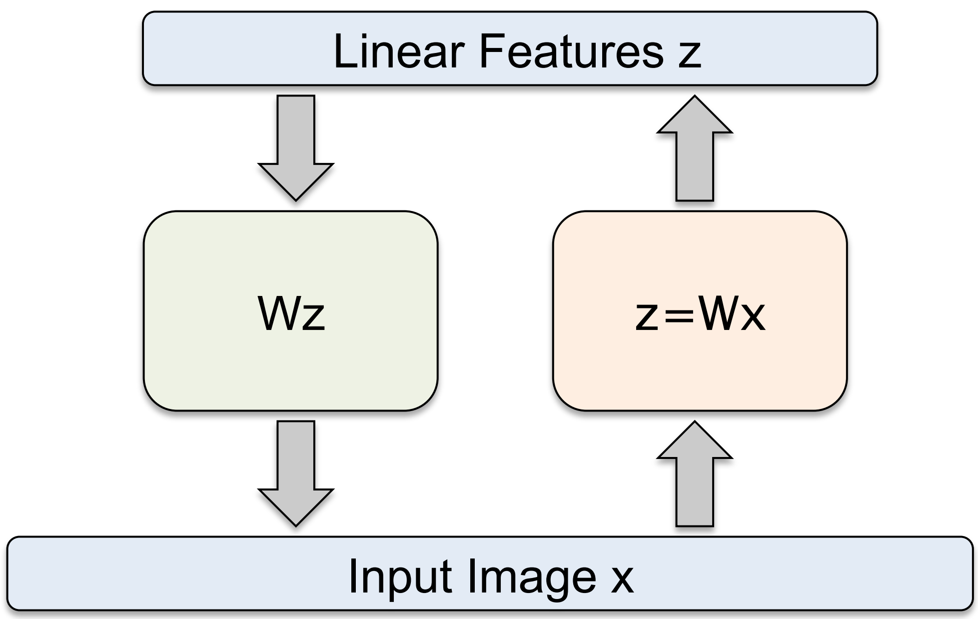


- An autoencoder with D inputs, D outputs, and K hidden units, with K<D.

- We can determine the network parameters W and D by minimizing the reconstruction error:

$$E(W, D) = \frac{1}{2} \sum_{n=1}^{N} ||y(\mathbf{x}_n, W, D) - \mathbf{x}_n||^2.$$

Autoencoder

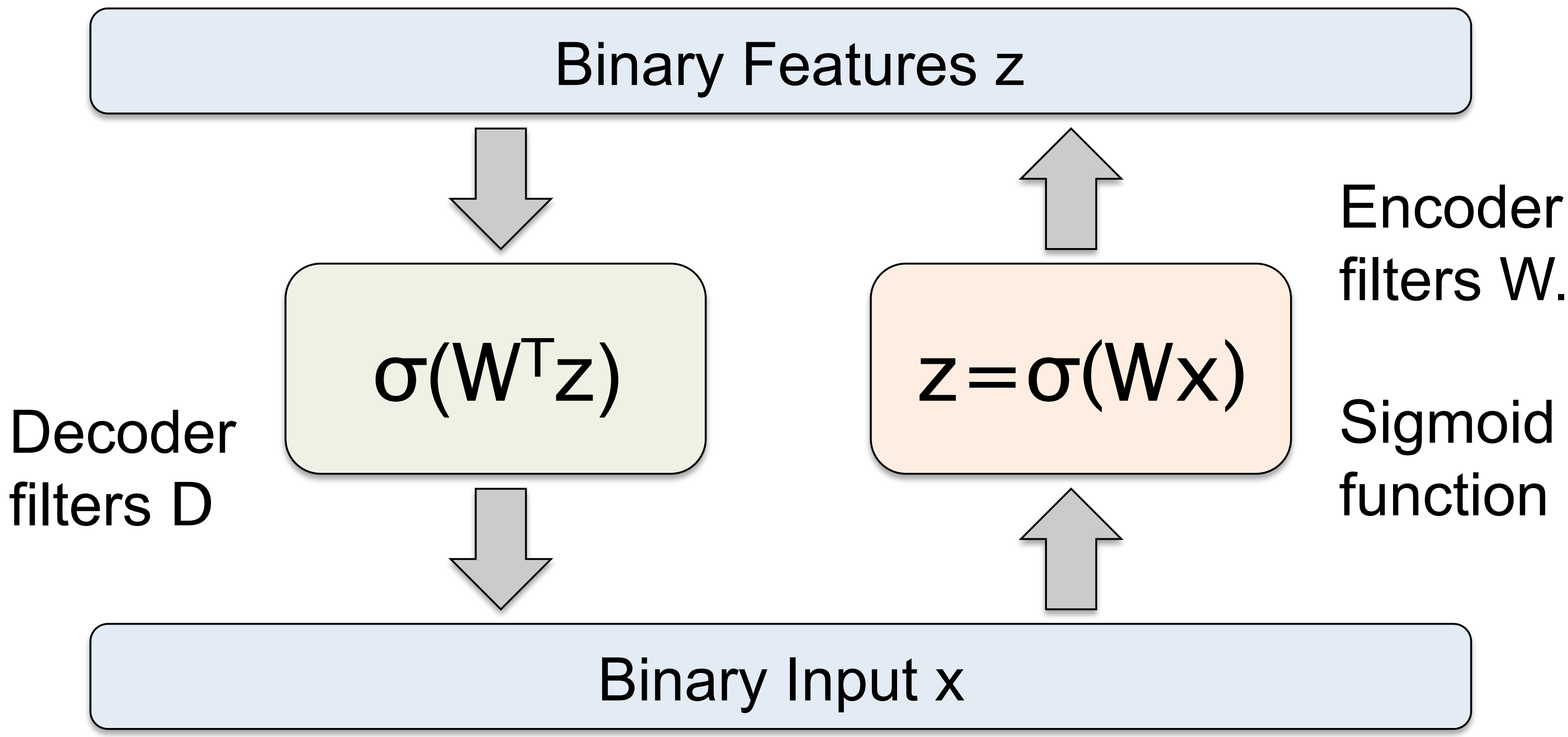


- If the hidden and output layers are linear, it will learn hidden units that are a linear function of the data and minimize the squared error.

- The K hidden units will span the same space as the first k principal components. The weight vectors may not be orthogonal.

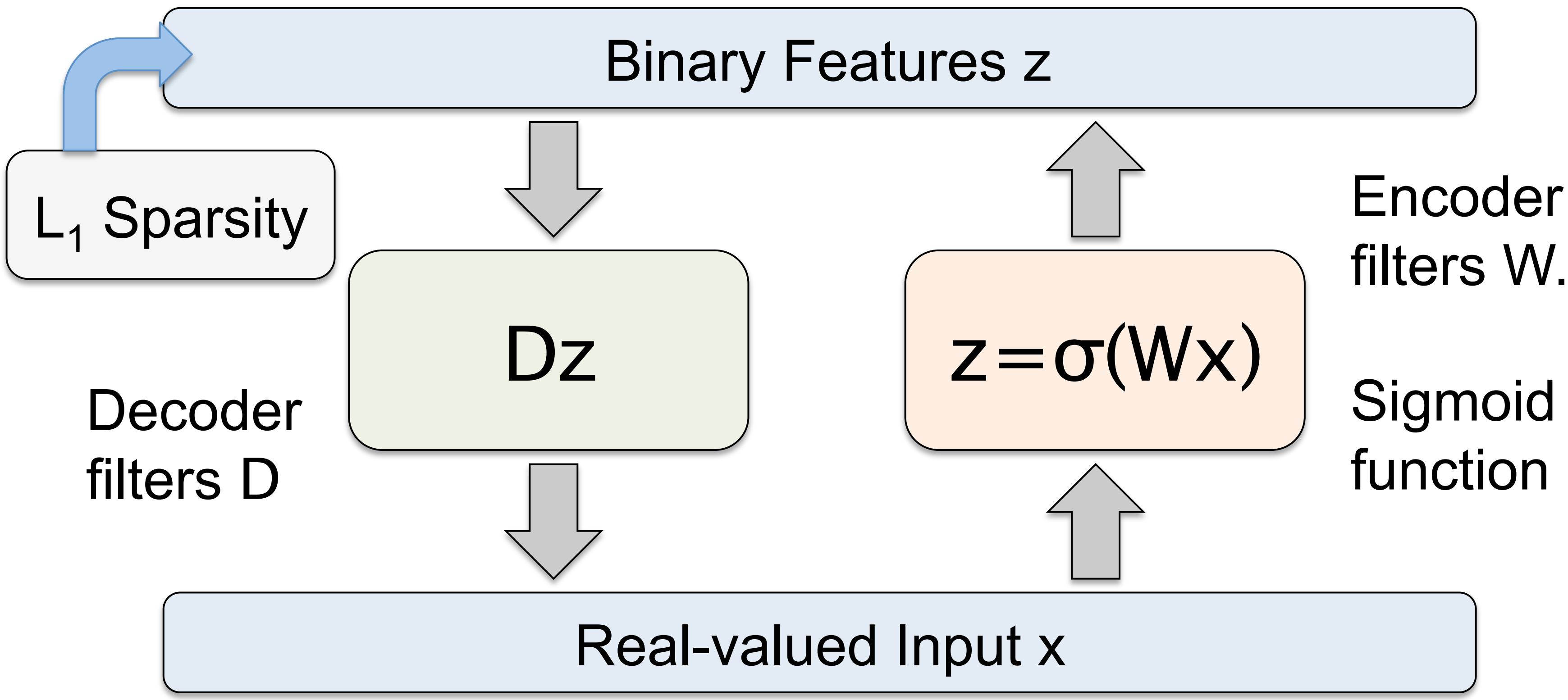
- With nonlinear hidden units, we have a nonlinear generalization of PCA.

Another Autoencoder Model



- Need additional constraints to avoid learning an identity.
- Relates to Restricted Boltzmann Machines (later).

Predictive Sparse Decomposition



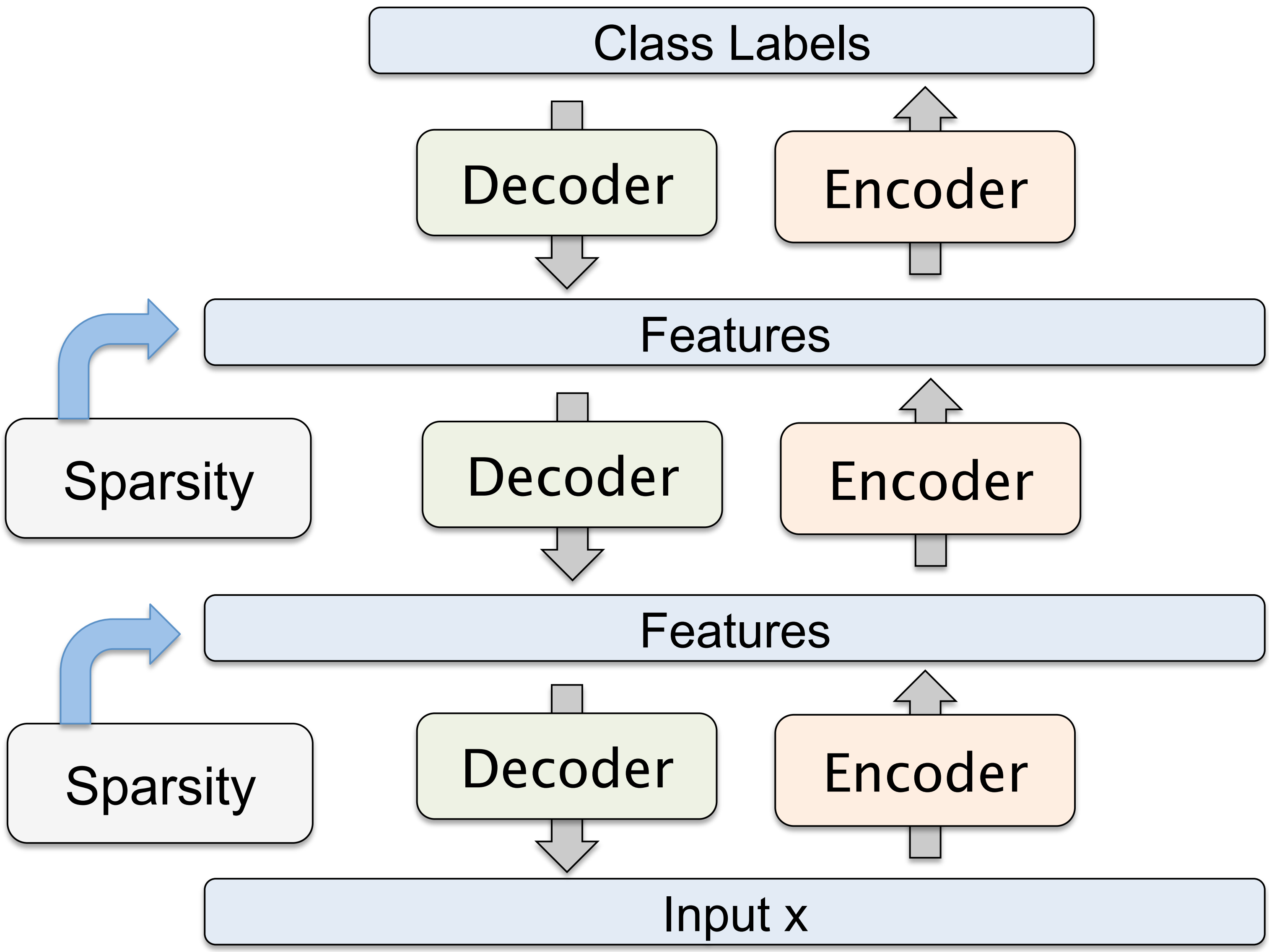
At training time

$$\min_{D,W,\mathbf{z}} ||D\mathbf{z}-\mathbf{x}||_2^2 + \lambda|\mathbf{z}|_1 + ||\sigma(W\mathbf{x})-\mathbf{z}||_2^2$$

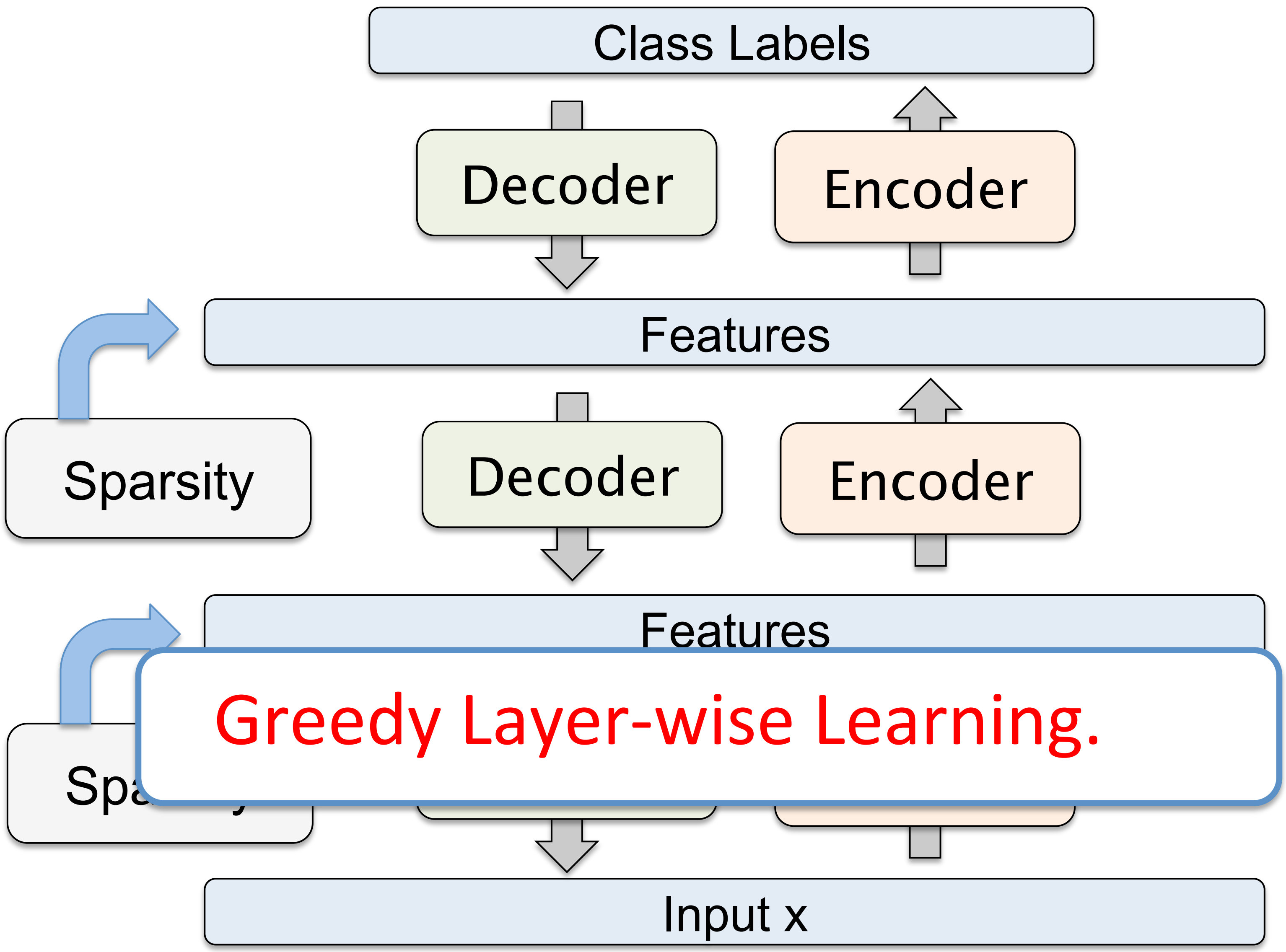
Decoder   Encoder

Kavukcuoglu et al., '09

Stacked Autoencoders



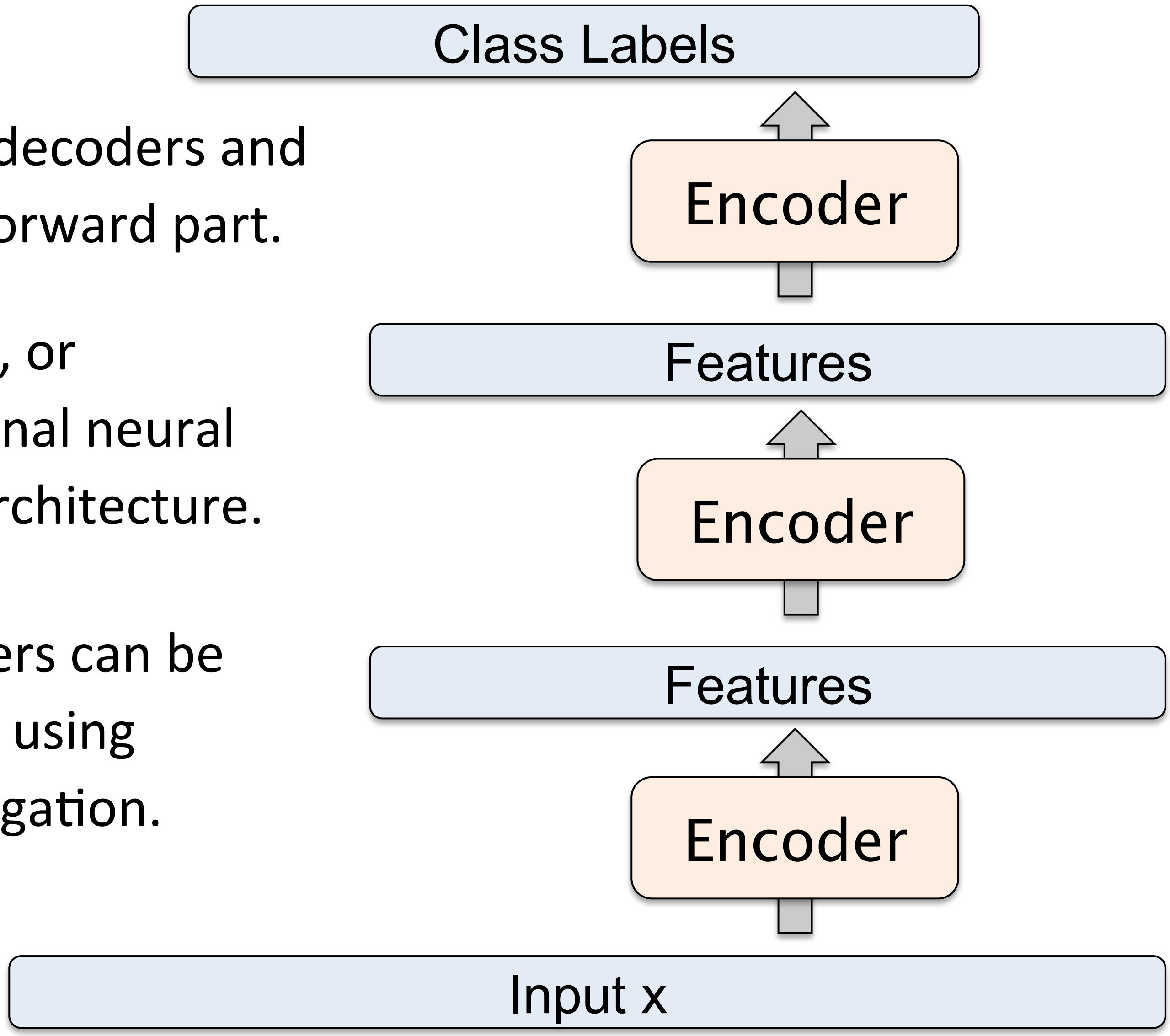
Stacked Autoencoders



Greedy Layer-wise Learning.

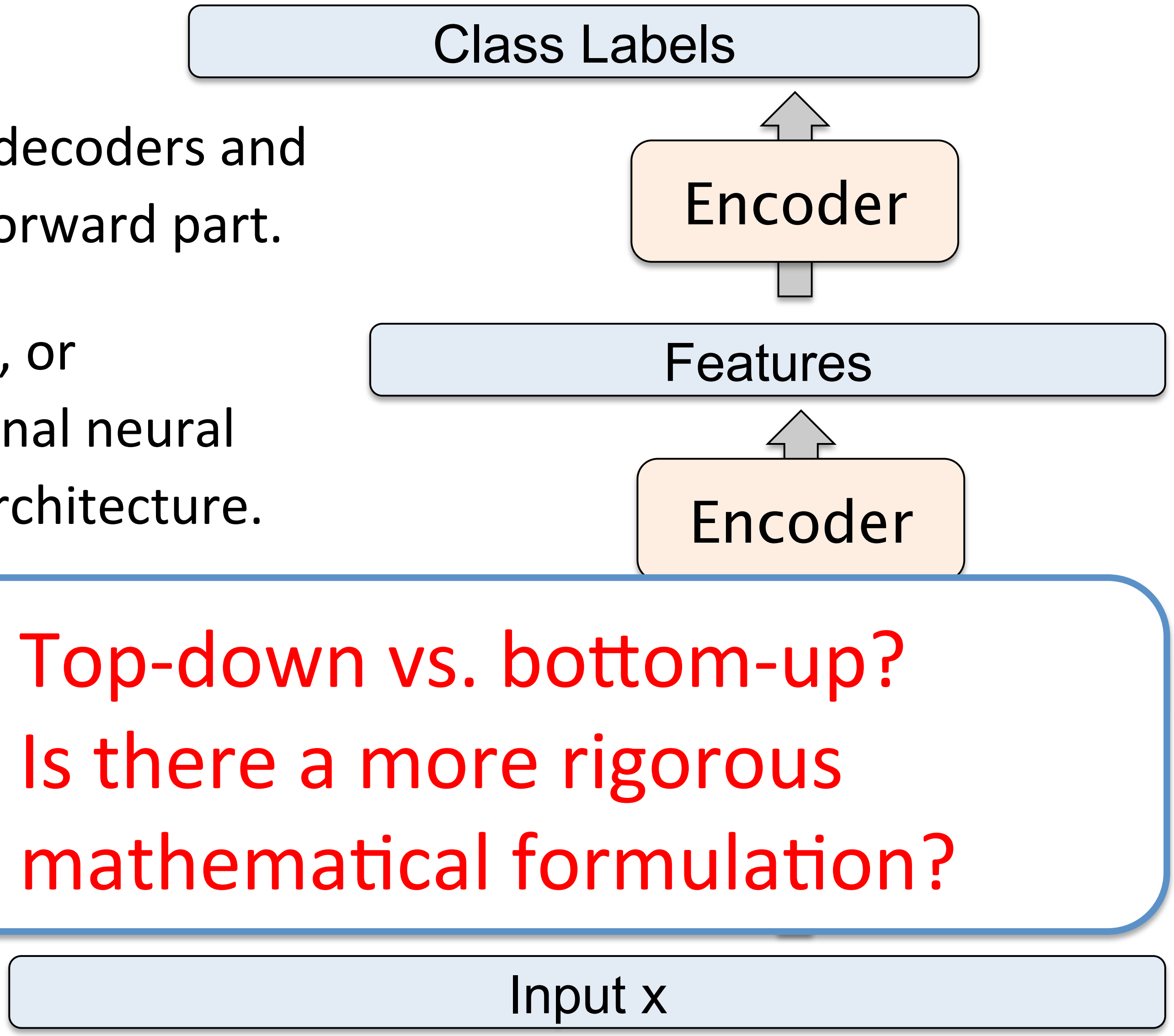
Stacked Autoencoders

- Remove decoders and use feed-forward part.
- Standard, or convolutional neural network architecture.
- Parameters can be fine-tuned using backpropagation.



Stacked Autoencoders

- Remove decoders and use feed-forward part.
- Standard, or convolutional neural network architecture.
- Param
fine-tu
backpr



Top-down vs. bottom-up?
Is there a more rigorous mathematical formulation?

Talk Roadmap

Part 1: Deep Networks

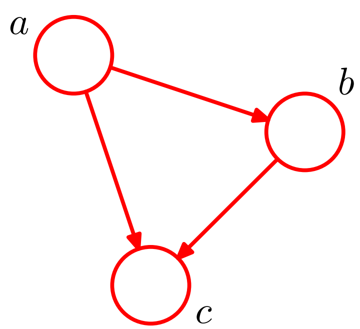
- Introduction, Sparse Coding, Autoencoders.
- Introduction to Graphical models.
- Restricted Boltzmann Machines: Learning low-level features.
- Deep Belief Networks: Learning Part-based Hierarchies.

Part 2: Deep Boltzmann Machines.

- Inference and Learning
- Advanced Deep Models

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.

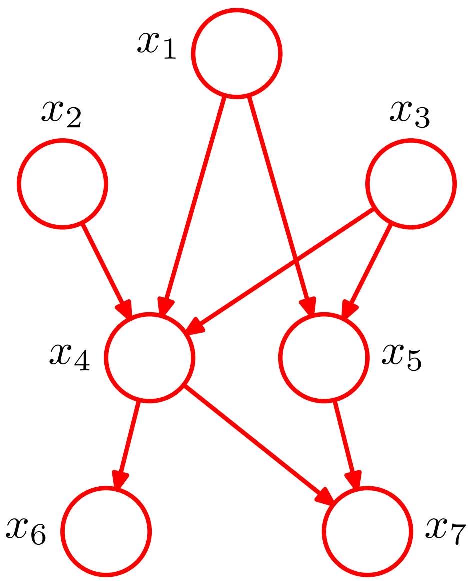
Two type of graphical models:

- **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.**

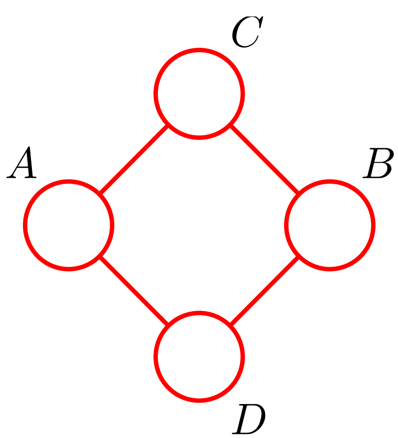
$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k|\mathrm{pa}_k)$$

- 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*.

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:

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

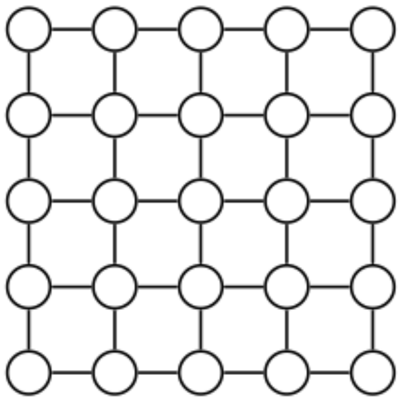
where E(x) is called an energy function.

Boltzmann distribution

- 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.

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:

$$\frac{\partial L(\theta)}{\partial \theta_{ij}} = \frac{1}{N}\sum_n [x_i^{(n)} x_j^{(n)}] - \sum_{\mathbf{x}} [x_i x_j P_\theta(\mathbf{x})] = \mathrm{E}_{P_{data}}[x_i x_j] - \mathrm{E}_{P_\theta}[x_i x_j]$$

Difficult to compute: exponentially many configurations

Talk Roadmap

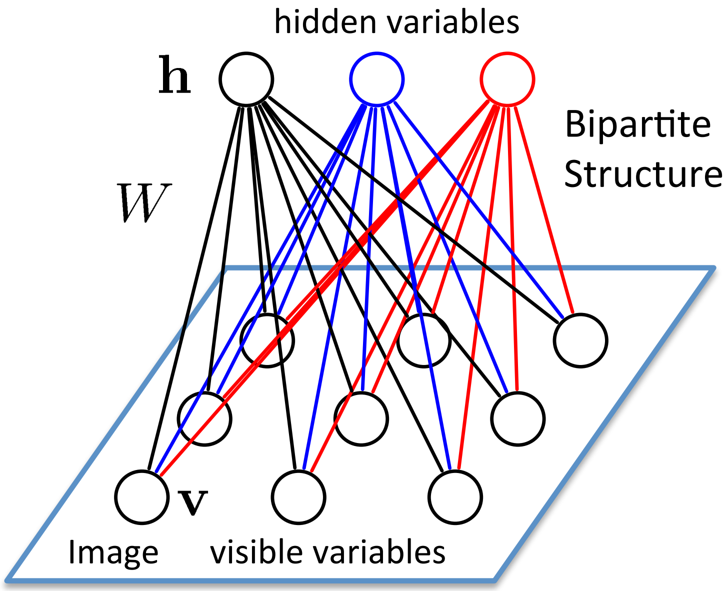
Part 1: Deep Networks

- Introduction, Sparse Coding, Autoencoders.
- Introduction to Graphical models.
- Restricted Boltzmann Machines: Learning low-level features.
- Deep Belief Networks: Learning Part-based Hierarchies.

Part 2: Deep Boltzmann Machines.

- Inference and Learning
- Advanced Deep Models

Restricted Boltzmann Machines



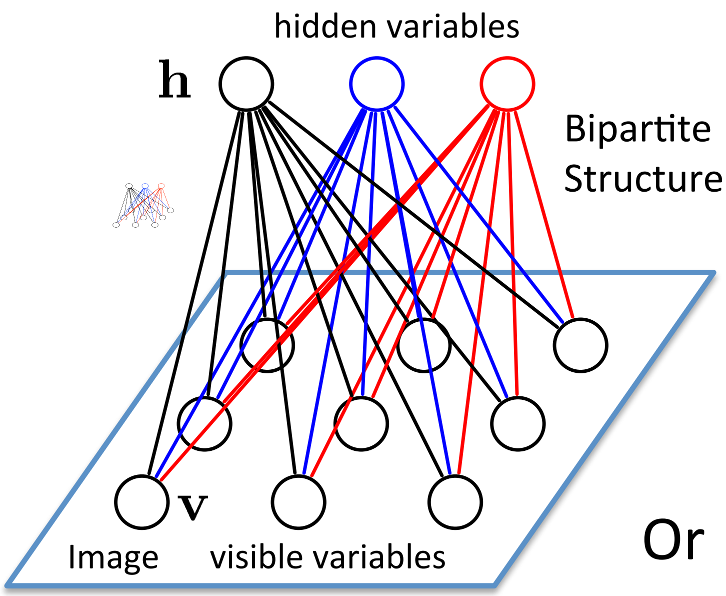
- Undirected bipartite graphical model
- Stochastic binary visible variables:
  $\mathbf{v} \in \{0,1\}^D$
- Stochastic binary hidden variables:
  $\mathbf{h} \in \{0,1\}^F$

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.

Restricted Boltzmann Machines



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

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

Or

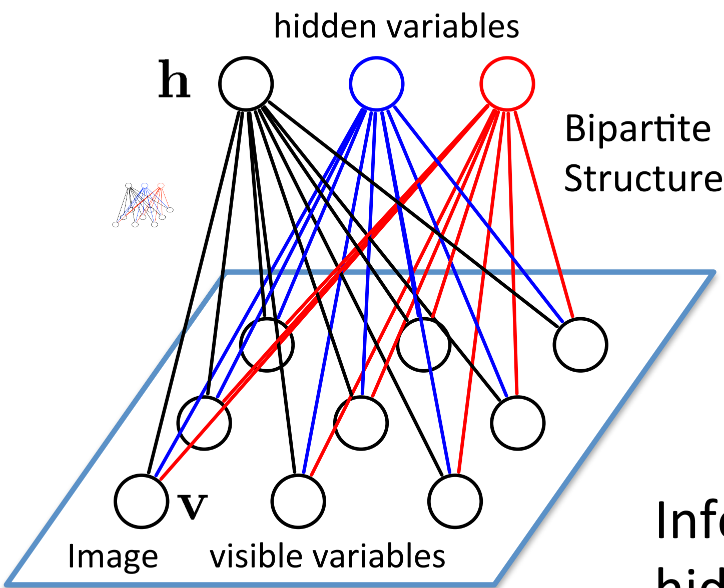
Pair-wise **Unary**

$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(\sum_{i=1}^{D}\sum_{j=1}^{F} W_{ij} v_i h_j + \sum_{i=1}^{D} v_i b_i + \sum_{j=1}^{F} h_j a_j\right)$$

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

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

Restricted Boltzmann Machines



Restricted: No interaction between hidden variables

Inferring the distribution over the hidden variables is easy:

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

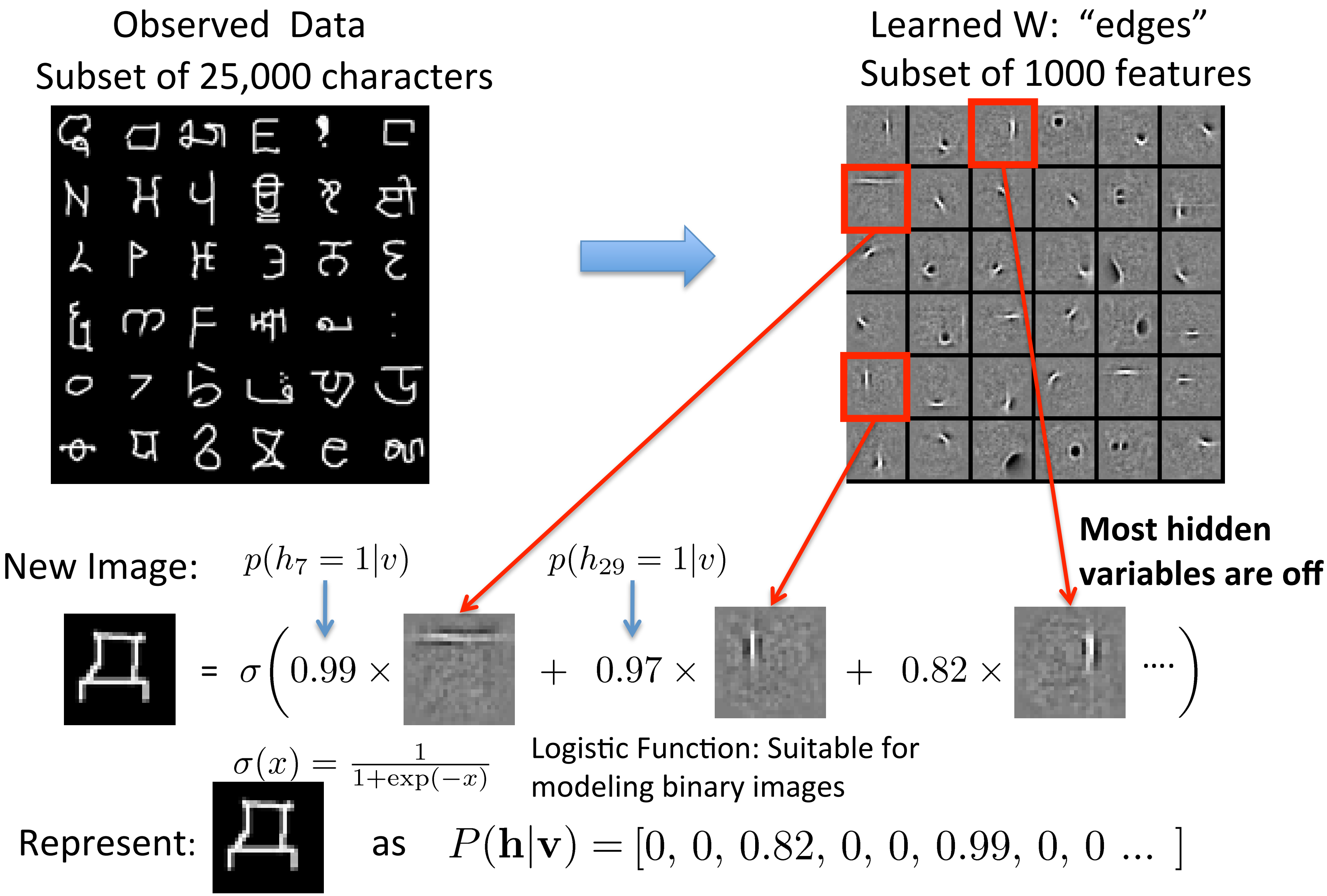
Factorizes: Easy to compute

Similarly:

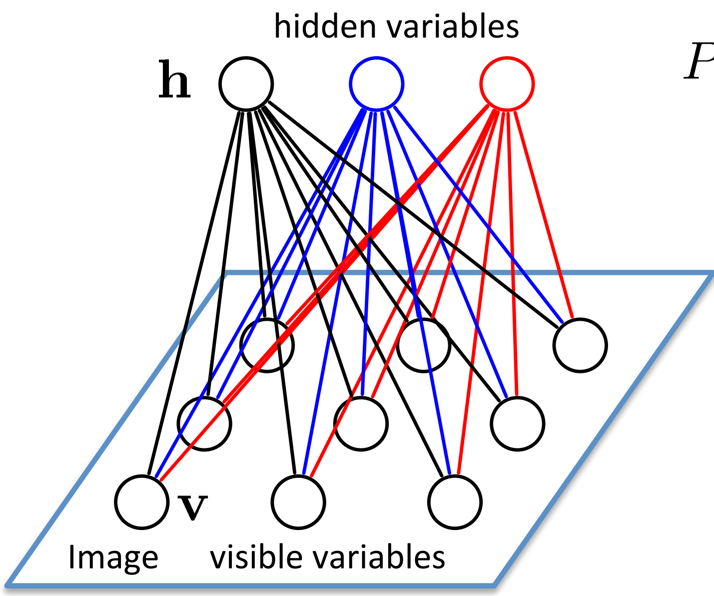
$$P(\mathbf{v}|\mathbf{h}) = \prod_i P(v_i|\mathbf{h}) \quad P(v_i = 1|\mathbf{h}) = \frac{1}{1 + \exp(-\sum_j W_{ij} h_j - b_i)}$$

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

Learning Features



Model Learning



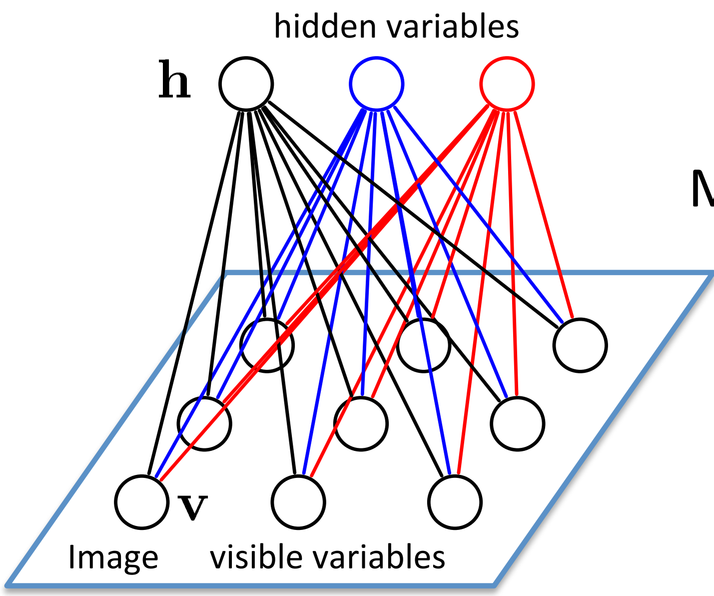
$$P_\theta(\mathbf{v}) = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}} \exp\left[\mathbf{v}^\top W\mathbf{h} + \mathbf{a}^\top\mathbf{h} + \mathbf{b}^\top\mathbf{v}\right]$$

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:

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

Model Learning



Maximize (penalized) log-likelihood objective:

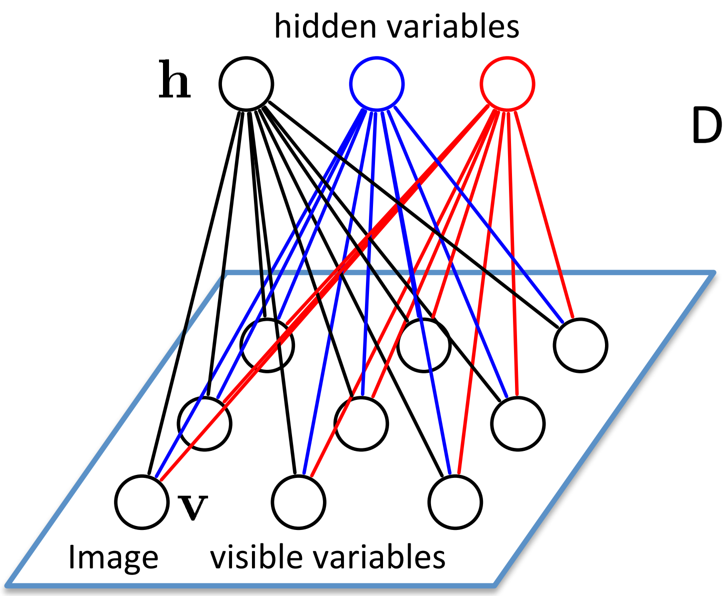
$$L(\theta) = \frac{1}{N}\sum_{n=1}^{N} \log P_\theta(\mathbf{v}^{(n)}) - \underbrace{\frac{\lambda}{N}||W||_F^2}_{\text{Regularization}}$$

Derivative of the log-likelihood:

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

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

Model Learning



Derivative of the log-likelihood:

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

$$\sum_{\mathbf{v},\mathbf{h}} v_i h_j P_\theta(\mathbf{v}, \mathbf{h})$$

Easy to compute exactly

Difficult to compute: exponentially many configurations.

Use MCMC

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

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

Approximate maximum likelihood learning

Approximate Learning

- An approximation to the gradient of the log-likelihood objective:

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

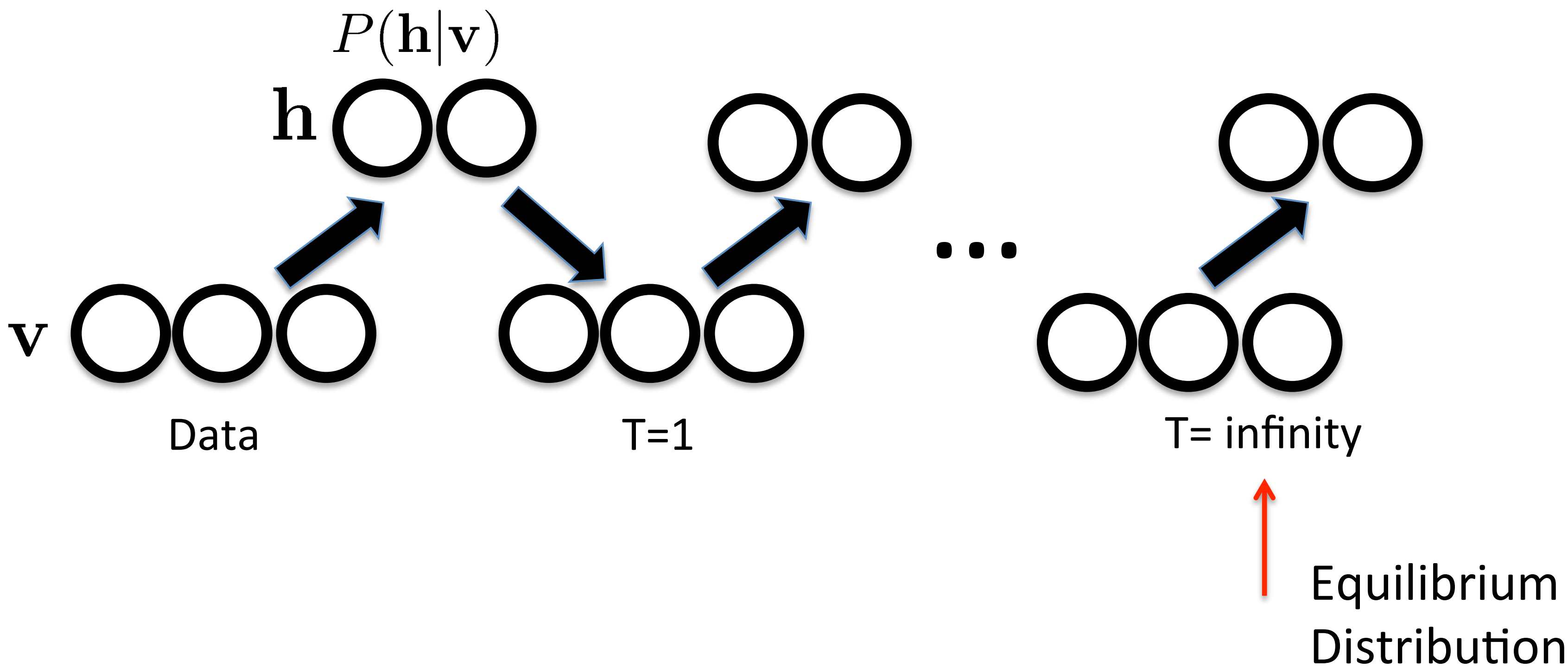
$$\sum_{\mathbf{v},\mathbf{h}} v_i h_j P_\theta(\mathbf{v}, \mathbf{h})$$

- Replace the average over all possible input configurations by samples.
- Run MCMC chain (Gibbs sampling) starting from the observed examples.

> - Initialize $v^0 = v$
> - Sample h^0 from $P(h \mid v^0)$
> - For t=1:T
>   - Sample v^t from $P(v \mid h^{t-1})$
>   - Sample h^t from $P(h \mid v^t)$

Approximate ML Learning for RBMs

Run Markov chain (alternating Gibbs Sampling):

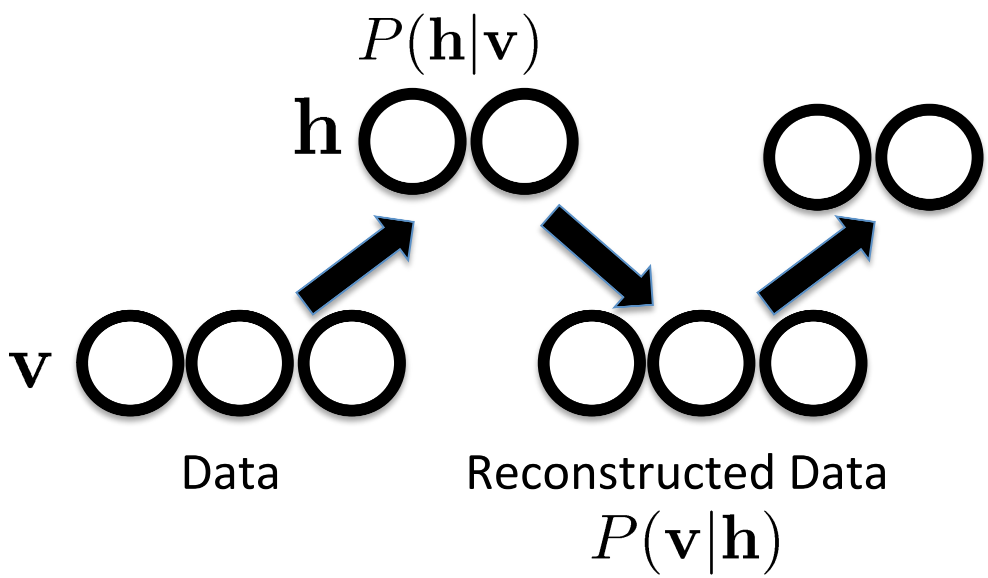


$$P(\mathbf{h}|\mathbf{v}) = \prod_j P(h_j|\mathbf{v}) \quad P(h_j = 1|\mathbf{v}) = \frac{1}{1 + \exp(-\sum_i W_{ij} v_i - a_j)}$$

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

Contrastive Divergence

A quick way to learn RBM:



- Start with a training vector on the visible units.
- Update all the hidden units in parallel.
- Update the all the visible units in parallel to get a "reconstruction".
- Update the hidden units again.

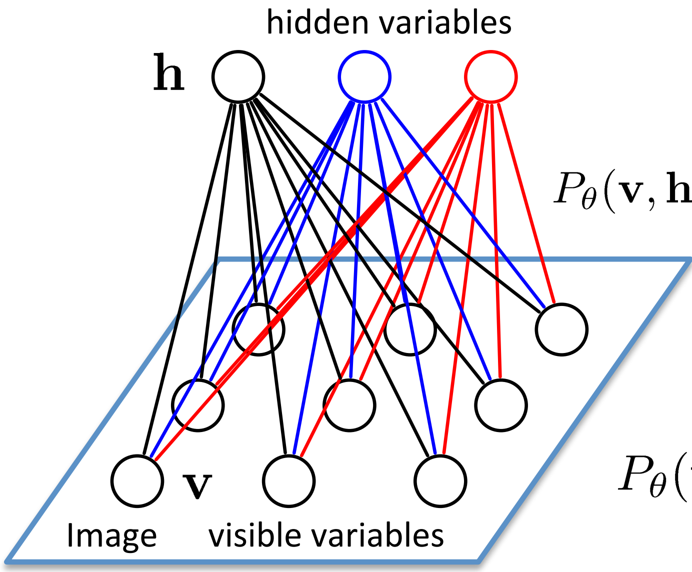
Update model parameters:

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

Implementation: ~10 lines of Matlab code.

Hinton, Neural Computation 2002

RBMs for Real-valued Data



Pair-wise **Unary**

$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(\sum_{i=1}^{D}\sum_{j=1}^{F} W_{ij} h_j \frac{v_i}{\sigma_i} + \sum_{i=1}^{D} \frac{(v_i - b_i)^2}{2\sigma_i^2} + \sum_{j=1}^{F} a_j h_j \right)$$

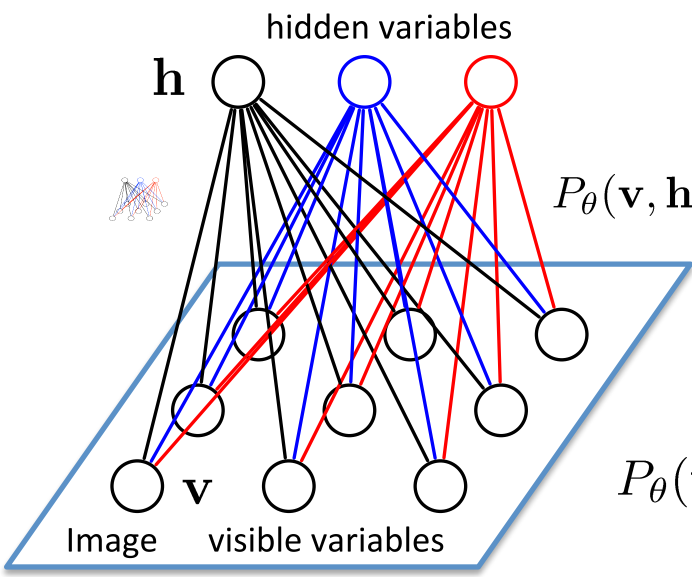
$$\theta = \{W, a, b\}$$

$$P_\theta(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^{D} P_\theta(v_i|\mathbf{h}) = \prod_{i=1}^{D} \mathcal{N}\left(b_i + \sum_{j=1}^{F} W_{ij} h_j, \sigma_i^2 \right)$$

Gaussian-Bernoulli RBM:

- Stochastic real-valued visible variables $\mathbf{v} \in \mathbb{R}^D$.
- Stochastic binary hidden variables $\mathbf{h} \in \{0,1\}^F$.
- Bipartite connections.

RBMs for Real-valued Data



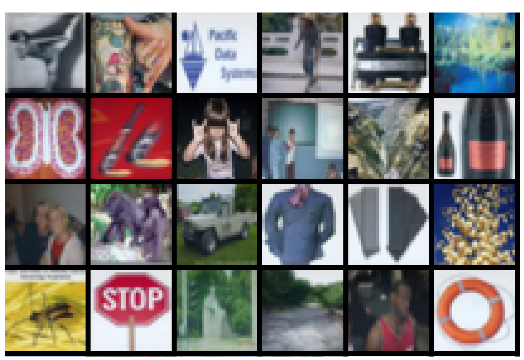
Pair-wise **Unary**

$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(\sum_{i=1}^{D}\sum_{j=1}^{F} W_{ij} h_j \frac{v_i}{\sigma_i} + \sum_{i=1}^{D} \frac{(v_i - b_i)^2}{2\sigma_i^2} + \sum_{j=1}^{F} a_j h_j \right)$$

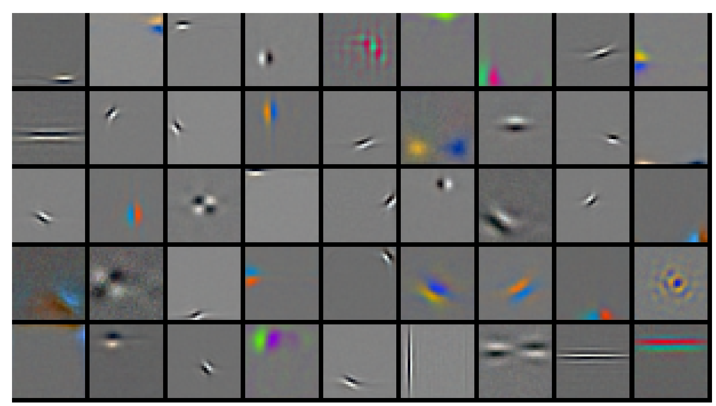
$$\theta = \{W, a, b\}$$

$$P_\theta(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^{D} P_\theta(v_i|\mathbf{h}) = \prod_{i=1}^{D} \mathcal{N}\left(b_i + \sum_{j=1}^{F} W_{ij} h_j, \sigma_i^2 \right)$$

4 million **unlabelled** images

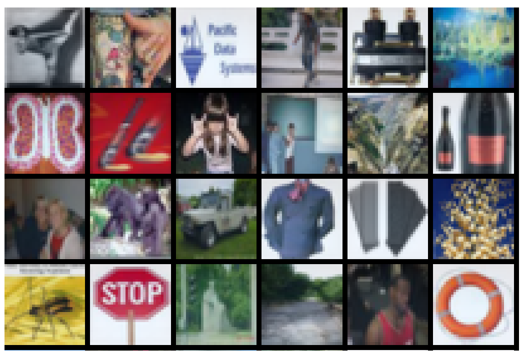


Learned features (out of 10,000)

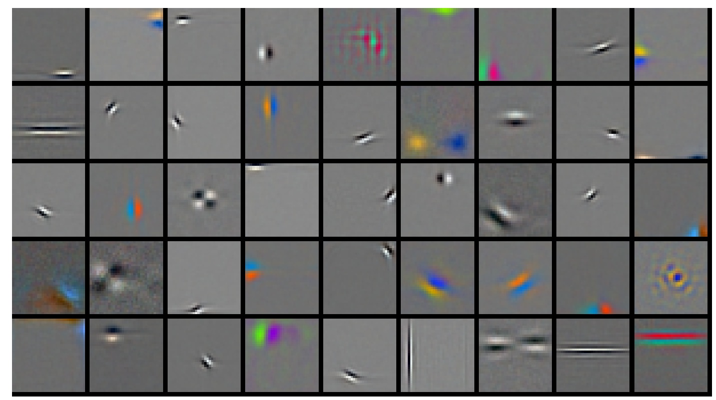


RBMs for Real-valued Data

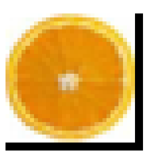
4 million **unlabelled** images



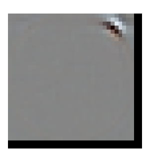
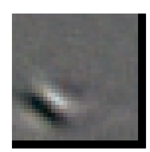
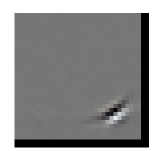
Learned features (out of 10,000)



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

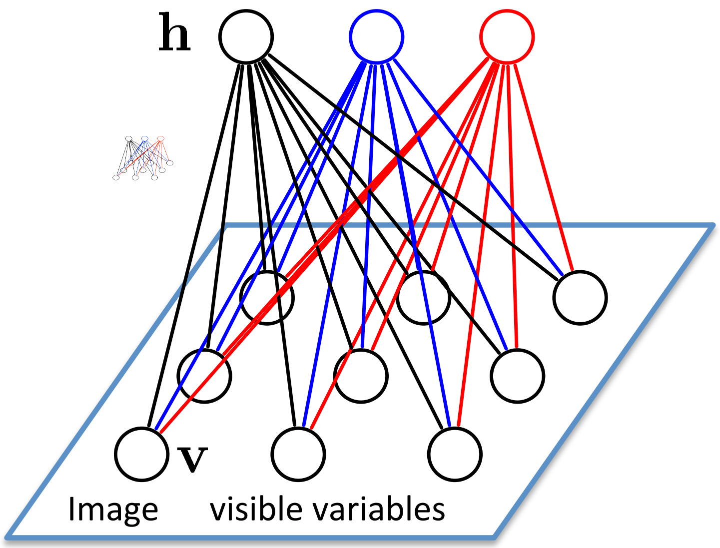


New Image

$= 0.9 *$  $+ \ 0.8 *$  $+ \ 0.6 *$  $\ldots$

RBMs for Images

Gaussian-Bernoulli RBM:



Interpretation: Mixture of exponential number of Gaussians

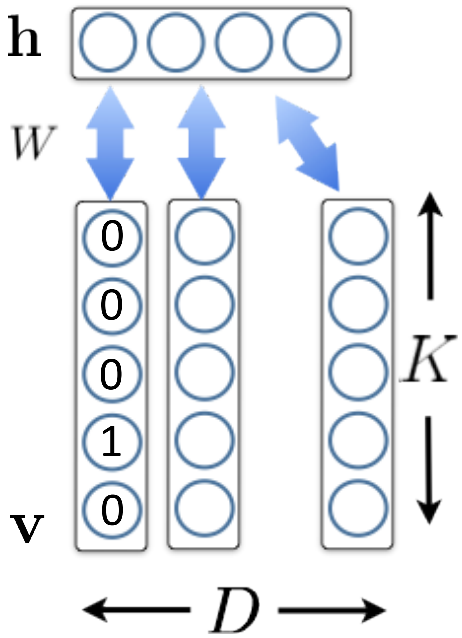
$$P_\theta(\mathbf{v}) = \sum_{\mathbf{h}} P_\theta(\mathbf{v}|\mathbf{h}) P_\theta(\mathbf{h}),$$

where

$$P_\theta(\mathbf{h}) = \int_{\mathbf{v}} P_\theta(\mathbf{v}, \mathbf{h}) d\mathbf{v}$$ is an implicit prior, and

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

RBMs for Word Counts



Pair-wise Unary

$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(\sum_{i=1}^{D}\sum_{k=1}^{K}\sum_{j=1}^{F} W_{ij}^{k} v_i^k h_j + \sum_{i=1}^{D}\sum_{k=1}^{K} v_i^k b_i^k + \sum_{j=1}^{F} h_j a_j \right)$$

$$\theta = \{W, a, b\}$$

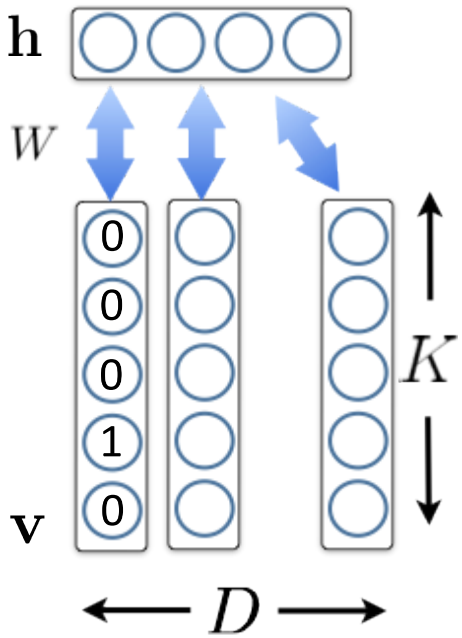
$$P_\theta(v_i^k = 1|\mathbf{h}) = \frac{\exp\left(b_i^k + \sum_{j=1}^{F} h_j W_{ij}^k\right)}{\sum_{q=1}^{K} \exp\left(b_i^q + \sum_{j=1}^{F} h_j W_{ij}^q\right)}$$

Replicated Softmax Model: undirected topic model:

- Stochastic 1-of-K visible variables.
- Stochastic binary hidden variables $\mathbf{h} \in \{0,1\}^F$.
- Bipartite connections.

(Salakhutdinov & Hinton, NIPS 2010, Srivastava & Salakhutdinov, NIPS 2012)

RBMs for Word Counts



Pair-wise **Unary**

$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(\sum_{i=1}^{D}\sum_{k=1}^{K}\sum_{j=1}^{F} W_{ij}^{k} v_i^k h_j + \sum_{i=1}^{D}\sum_{k=1}^{K} v_i^k b_i^k + \sum_{j=1}^{F} h_j a_j \right)$$

$$\theta = \{W, a, b\}$$

$$P_\theta(v_i^k = 1|\mathbf{h}) = \frac{\exp\left(b_i^k + \sum_{j=1}^{F} h_j W_{ij}^k\right)}{\sum_{q=1}^{K} \exp\left(b_i^q + \sum_{j=1}^{F} h_j W_{ij}^q\right)}$$

WIKIPEDIA
The Free Encyclopedia

REUTERS
AP Associated Press

Reuters dataset:
804,414 **unlabeled**
newswire stories
Bag-of-Words

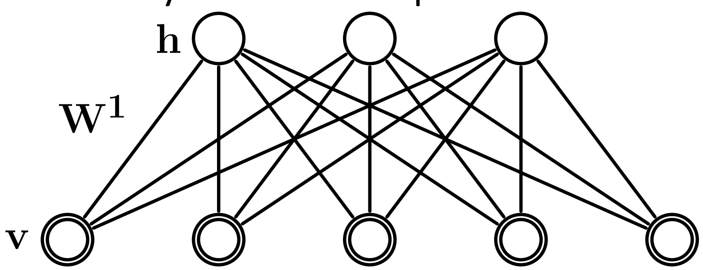
Learned features: ``topics''

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

$$P_\theta(\mathbf{v}, \mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(\sum_{ijk} W_{ij}^k v_i^k h_j + \sum_{ik} b_i^k v_i^k + \sum_j a_j h_j \right)$$

Binary hidden: user preferences



Multinomial visible: user ratings

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

Learned features: ``genre''

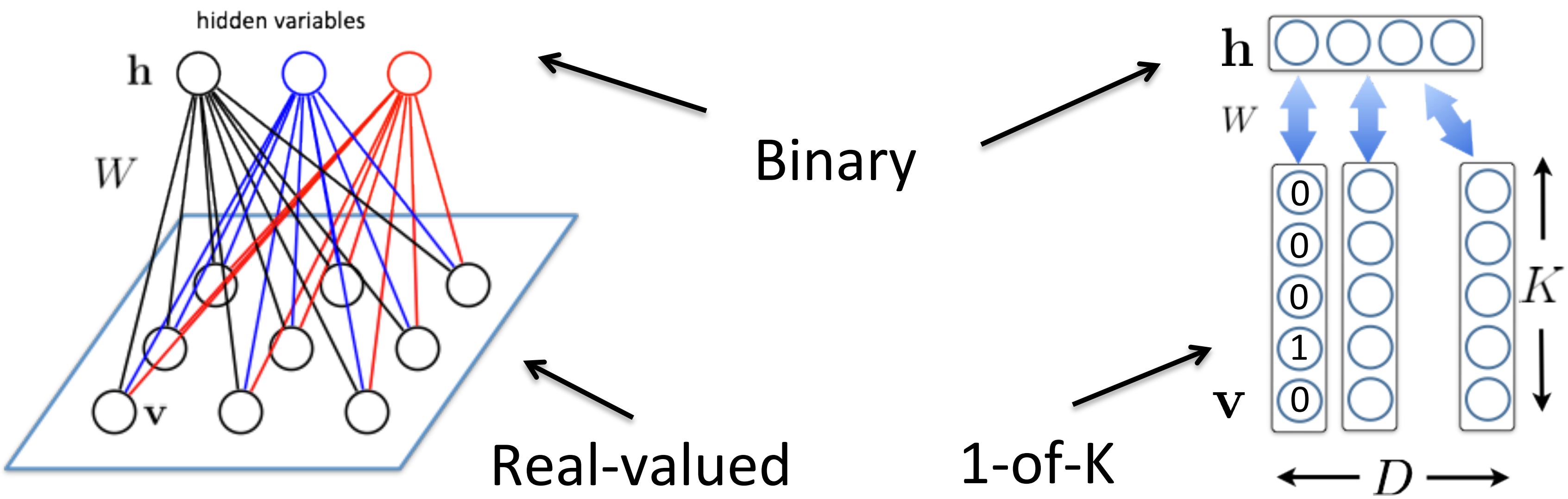
Fahrenheit 9/11	Independence Day
Bowling for Columbine	The Day After Tomorrow
The People vs. Larry Flynt	Con Air
Canadian Bacon	Men in Black II
La Dolce Vita	Men in Black
Friday the 13th	Scary Movie
The Texas Chainsaw Massacre	Naked Gun
Children of the Corn	Hot Shots!
Child's Play	American Pie
The Return of Michael Myers	Police Academy

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

(Salakhutdinov, Mnih, Hinton, ICML 2007)

Different Data Modalities

- Binary/Gaussian/Softmax RBMs: All have binary hidden variables but use them to model different kinds of data.



- It is easy to infer the states of the hidden variables:

$$P_\theta(\mathbf{h}|\mathbf{v}) = \prod_{j=1}^{F} P_\theta(h_j|\mathbf{v}) = \prod_{j=1}^{F} \frac{1}{1 + \exp(-a_j - \sum_{i=1}^{D} W_{ij} v_i)}$$

Product of Experts

The joint distribution is given by:

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

Marginalizing over hidden variables:

Product of Experts

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

government	clinton	bribery	oil	stock	...
auhority	house	corruption	barrel	wall	
power	president	dishonesty	exxon	street	
empire	bill	putin	putin	point	
putin	congress	fraud	drill	dow	

Putin

Topics "government", "corruption" and "oil" can combine to give very high probability to a word "Putin".

(Salakhutdinov & Hinton, NIPS 2010)

Product of Experts

The joint distribution is given by:

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

Marginalizing o

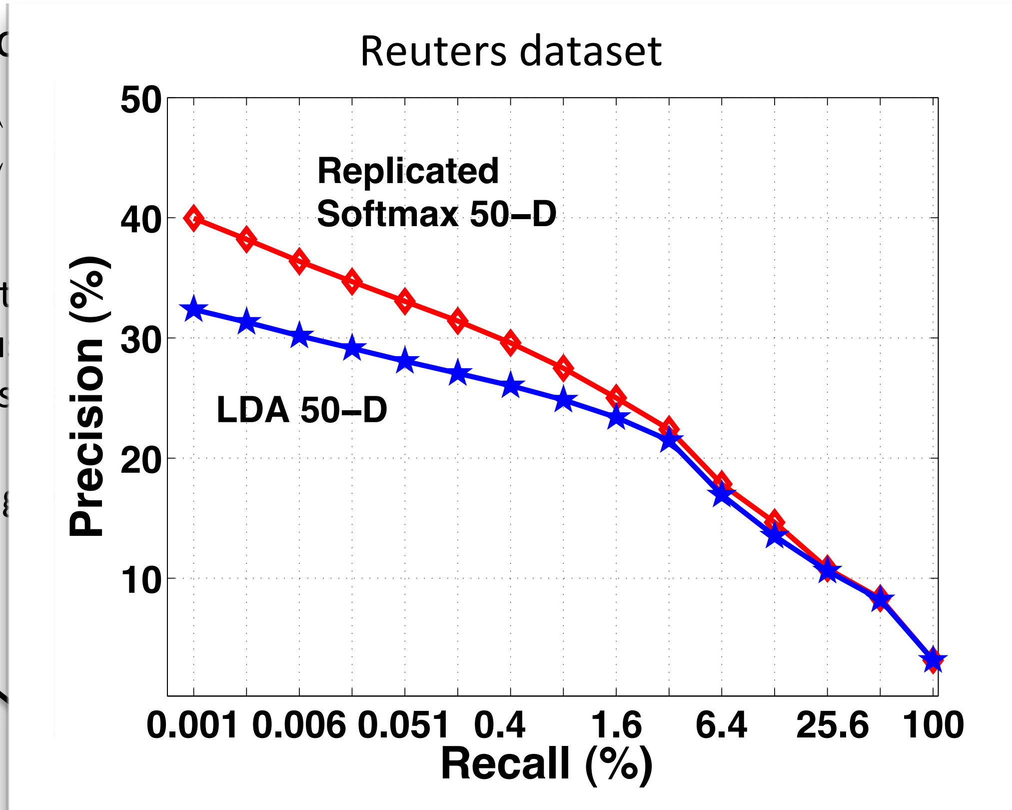
$$P_\theta(\mathbf{v}) = \sum_{\mathbf{h}}$$

duct of Experts

$W_{ij} v_i)\Big)$

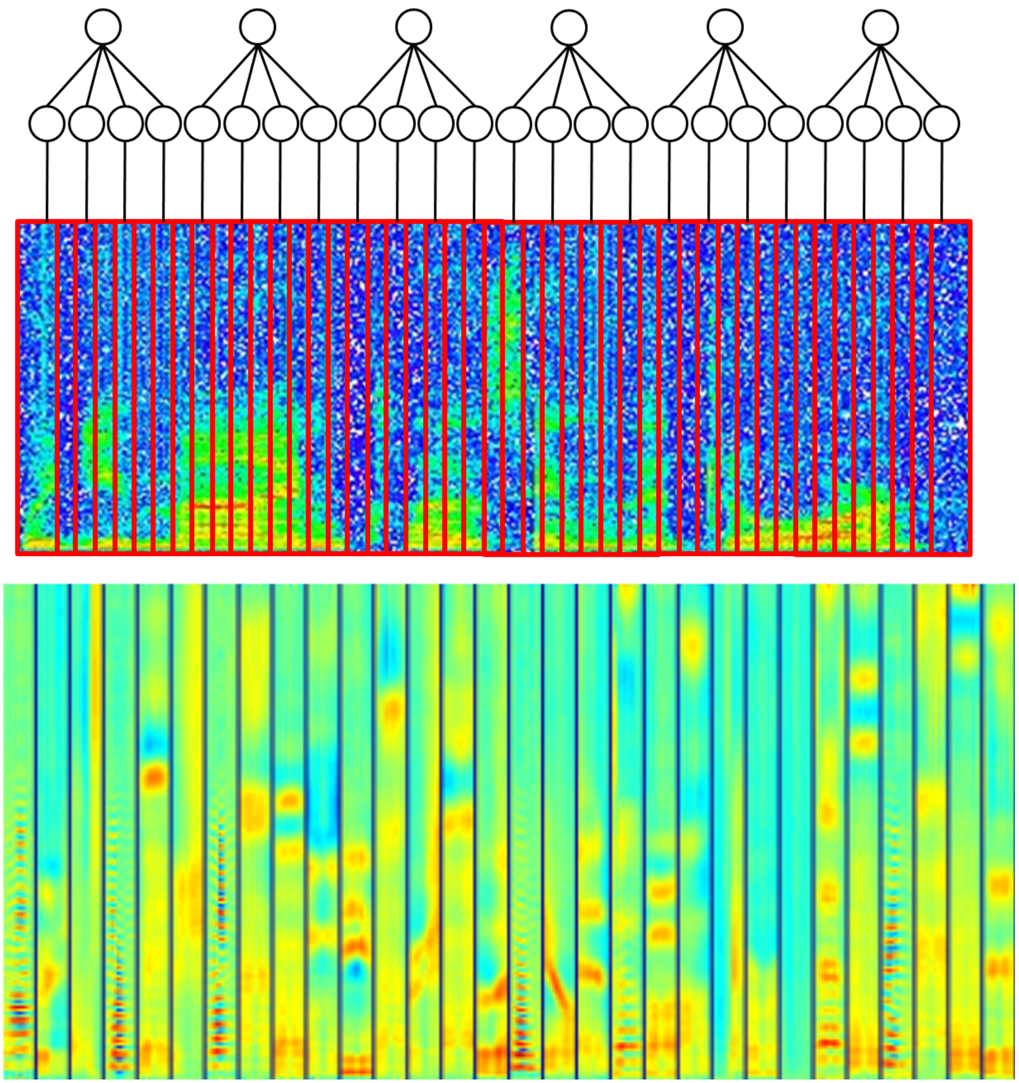
government
auhority
power
empire
putin

clint
hou
pres
bill
con



tations allow the
, "corruption" and
ive very high
probability to a word "Putin".

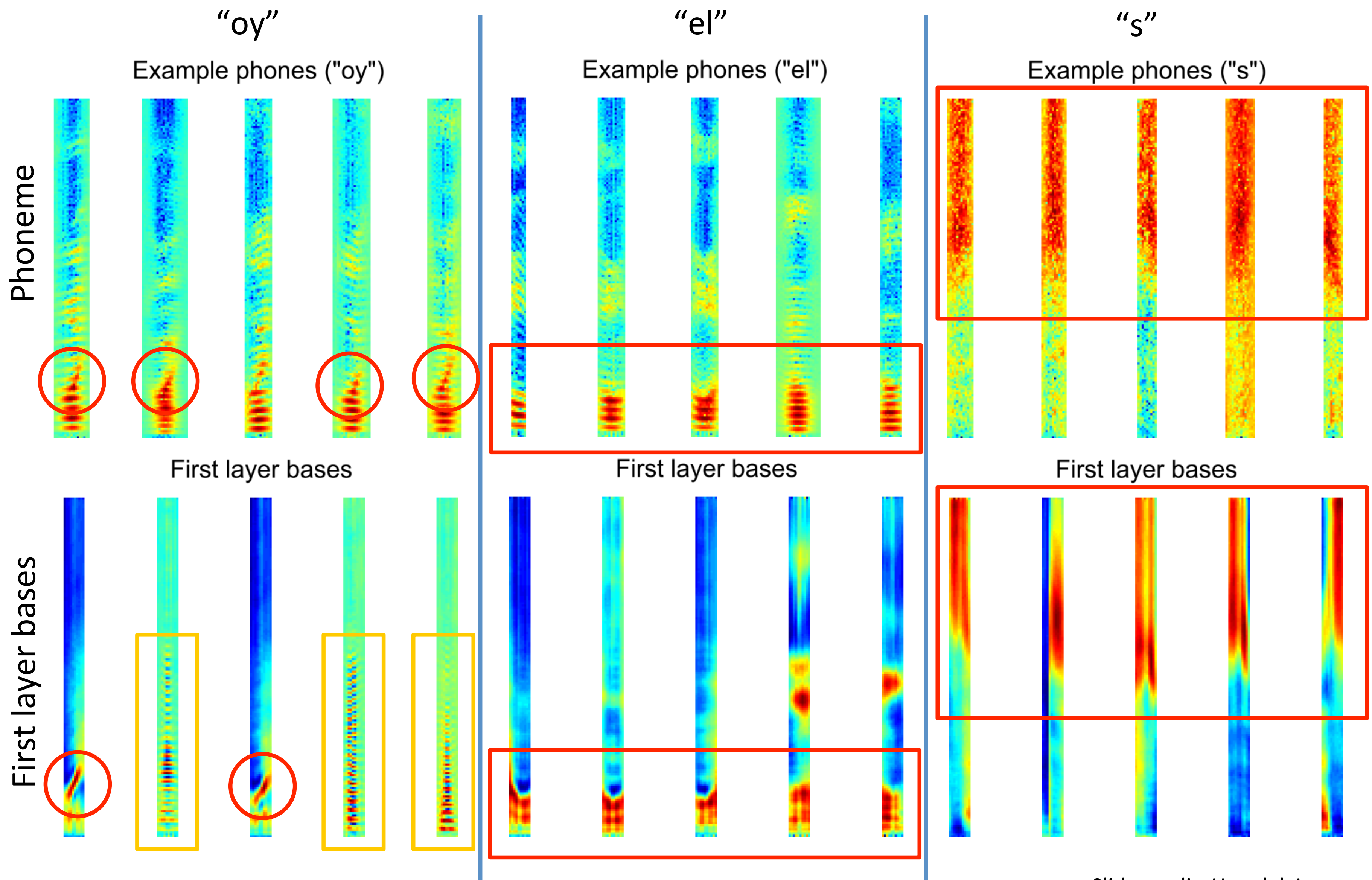
Speech



Learned first-layer bases

Lee et.al., NIPS 2009

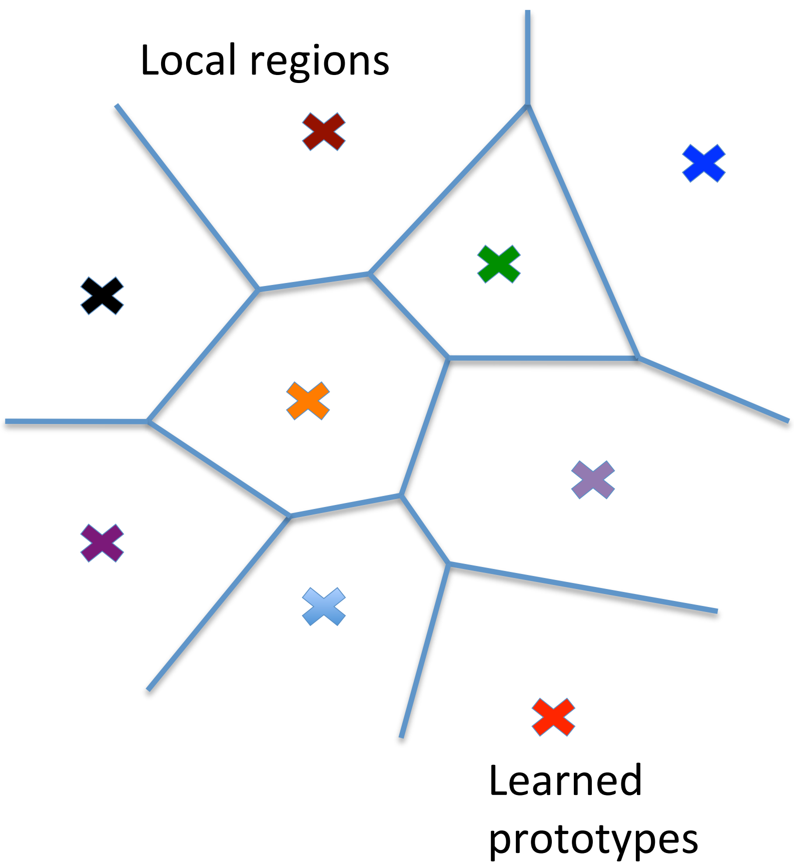
Comparison of bases to phonemes



Slide credit: Honglak Lee

Local vs. Distributed Representations

- Clustering, Nearest Neighbors, RBF SVM, local density estimators

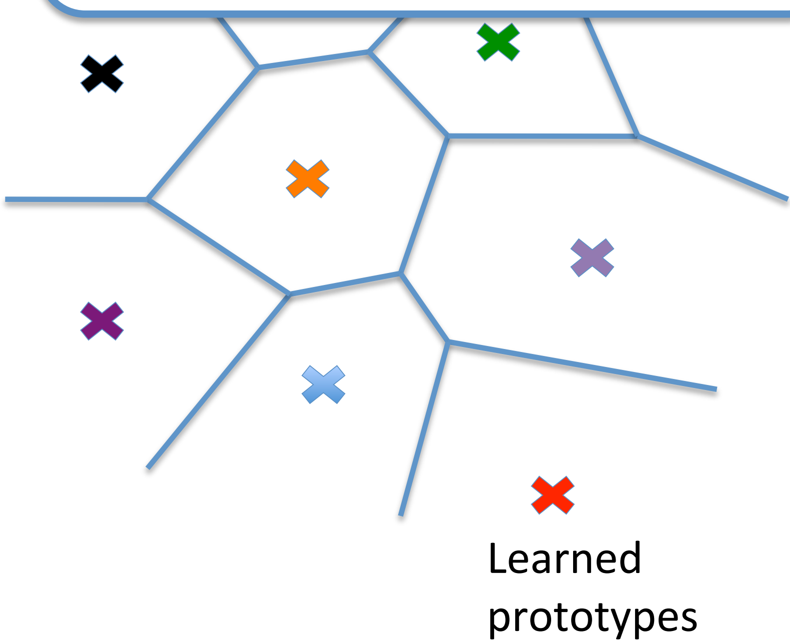


Bengio, 2009, Foundations and Trends in Machine Learning

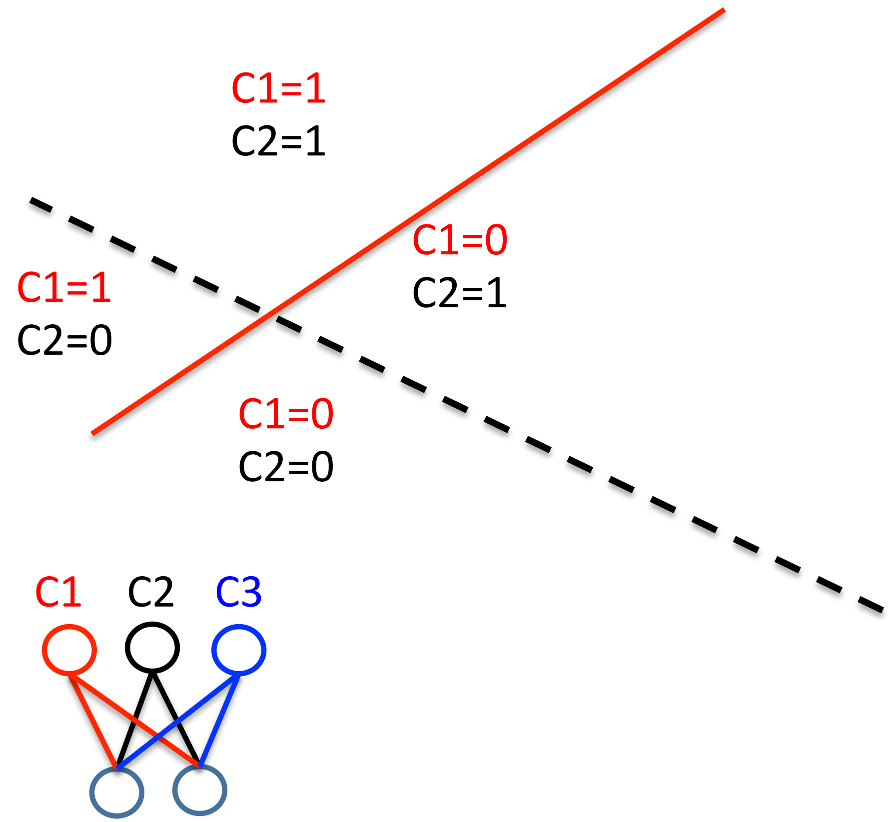
Local vs. Distributed Representations

- Clustering, Nearest Neighbors, RBF SVM, local density estimators

- Parameters for each region.
- # of regions is linear with # of parameters.



- RBMs, Factor models, PCA, Sparse Coding, Deep models

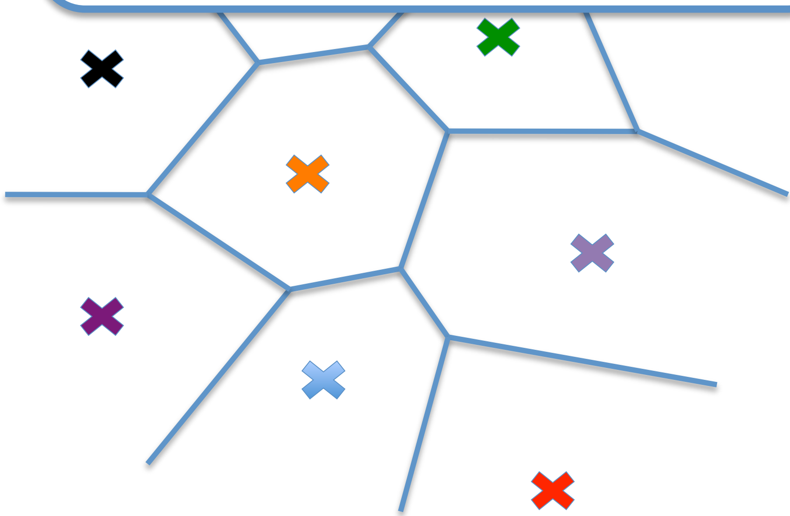


Bengio, 2009, Foundations and Trends in Machine Learning

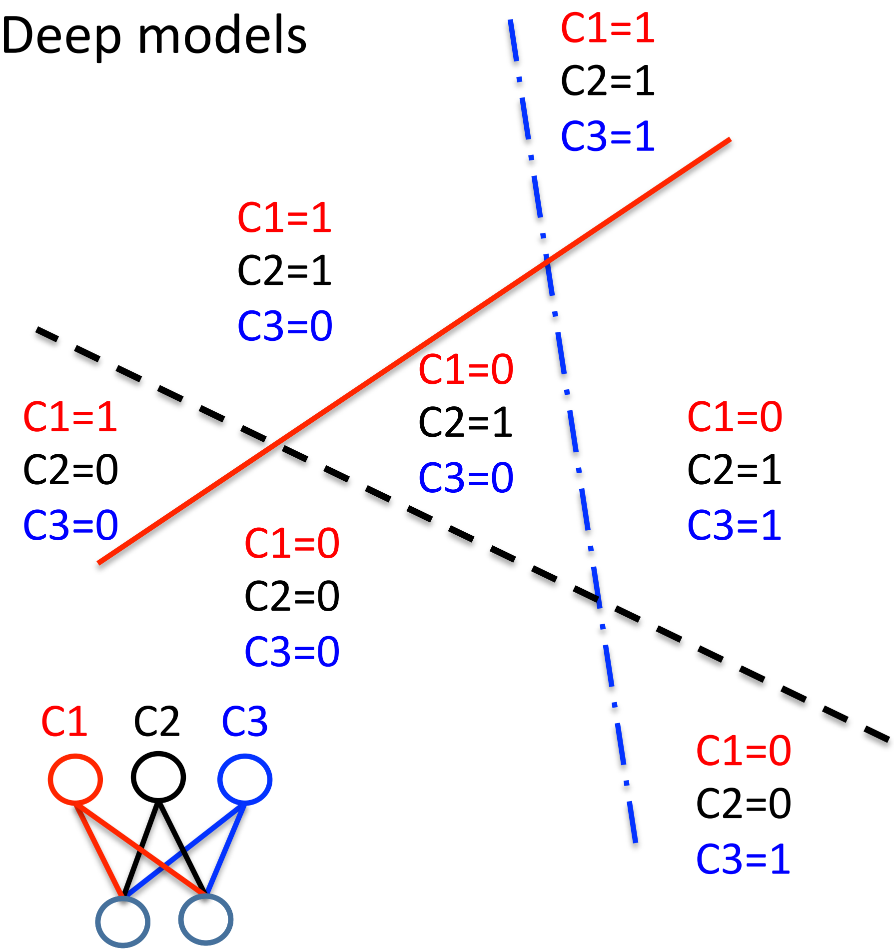
Local vs. Distributed Representations

- Clustering, Nearest Neighbors, RBF SVM, local density estimators

- Parameters for each region.
- # of regions is linear with # of parameters.



- RBMs, Factor models, PCA, Sparse Coding, Deep models

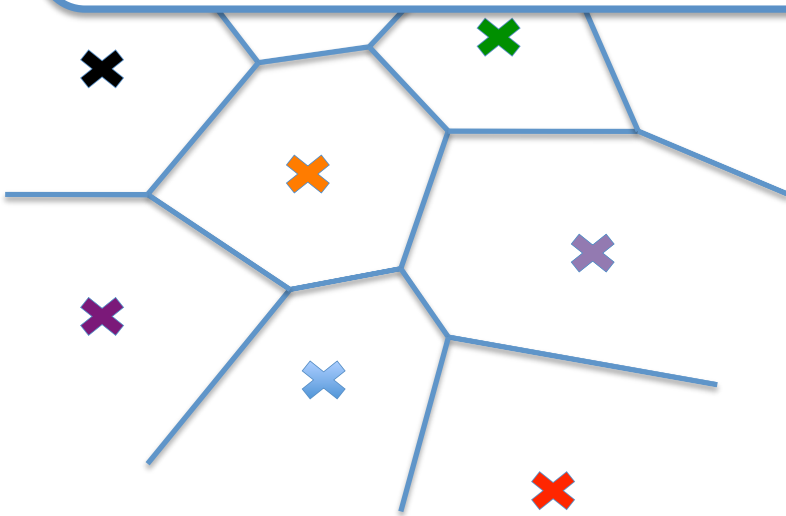


Bengio, 2009, Foundations and Trends in Machine Learning

Local vs. Distributed Representations

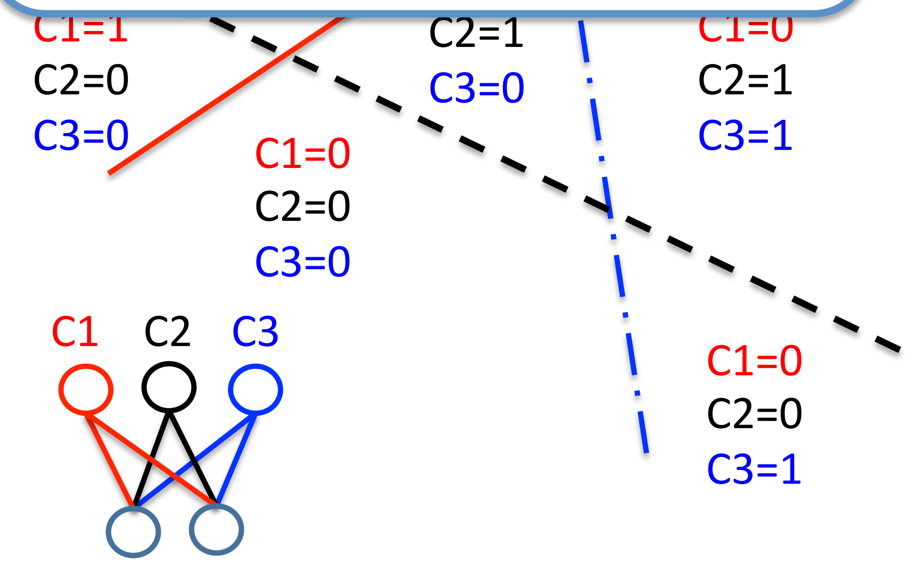
- Clustering, Nearest Neighbors, RBF SVM, local density estimators

- Parameters for each region.
- # of regions is linear with # of parameters.



- RBMs, Factor models, PCA, Sparse Coding, Deep models

- Each parameter affects many regions, not just local.
- # of regions grows (roughly) exponentially in # of parameters.



Bengio, 2009, Foundations and Trends in Machine Learning

Multiple Application Domains

- Natural Images
- Text/Documents
- Collaborative Filtering / Matrix Factorization
- 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

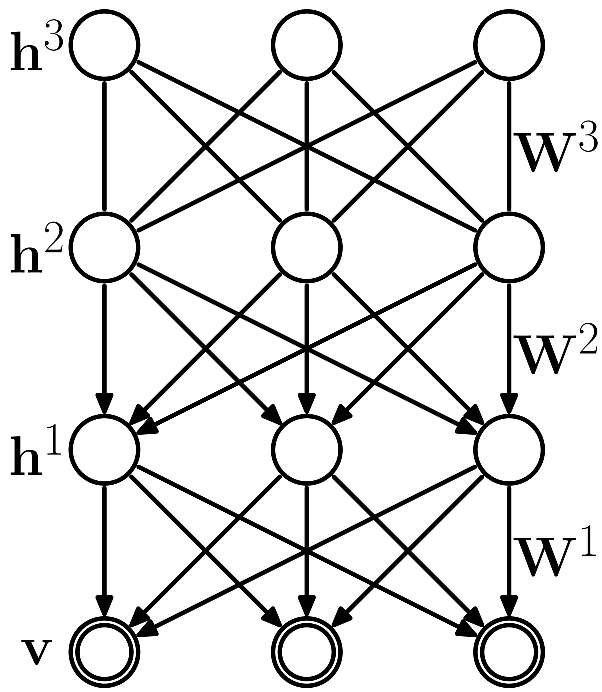
Part 1: Deep Networks

- Introduction, Sparse Coding, Autoencoders.
- Introduction to Graphical models.
- Restricted Boltzmann Machines: Learning low-level features.
- Deep Belief Networks: Learning Part-based Hierarchies.

Part 2: Deep Boltzmann Machines.

- Inference and Learning
- Advanced Deep Models

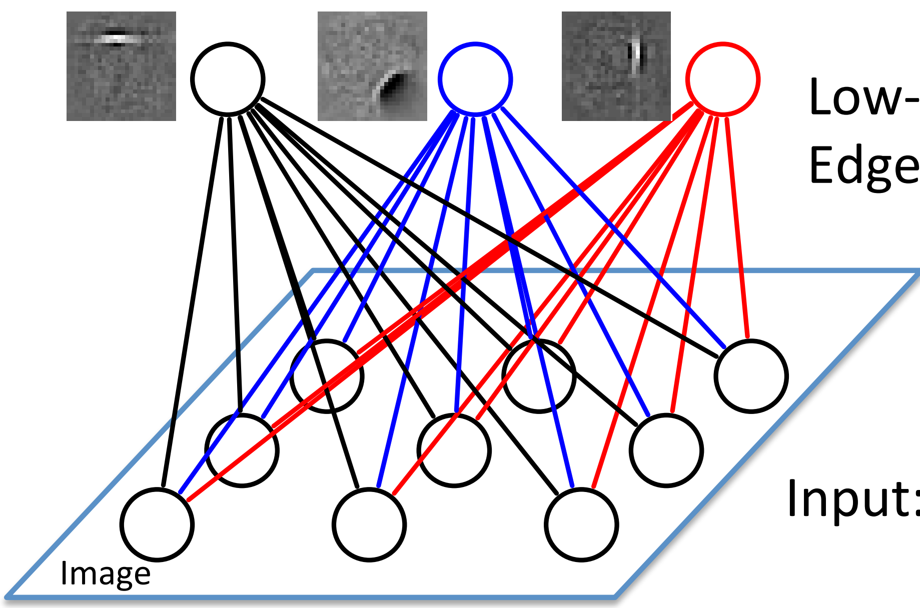
Deep Belief Network



- Probabilistic Generative model.
- Contains multiple layers of nonlinear representation.
- Fast, greedy layer-wise pretraining algorithm.
- Inferring the states of the latent variables in highest layers is easy.

- Inferring the states of the latent variables in highest layers is easy.

Deep Belief Network



Low-level features:
Edges

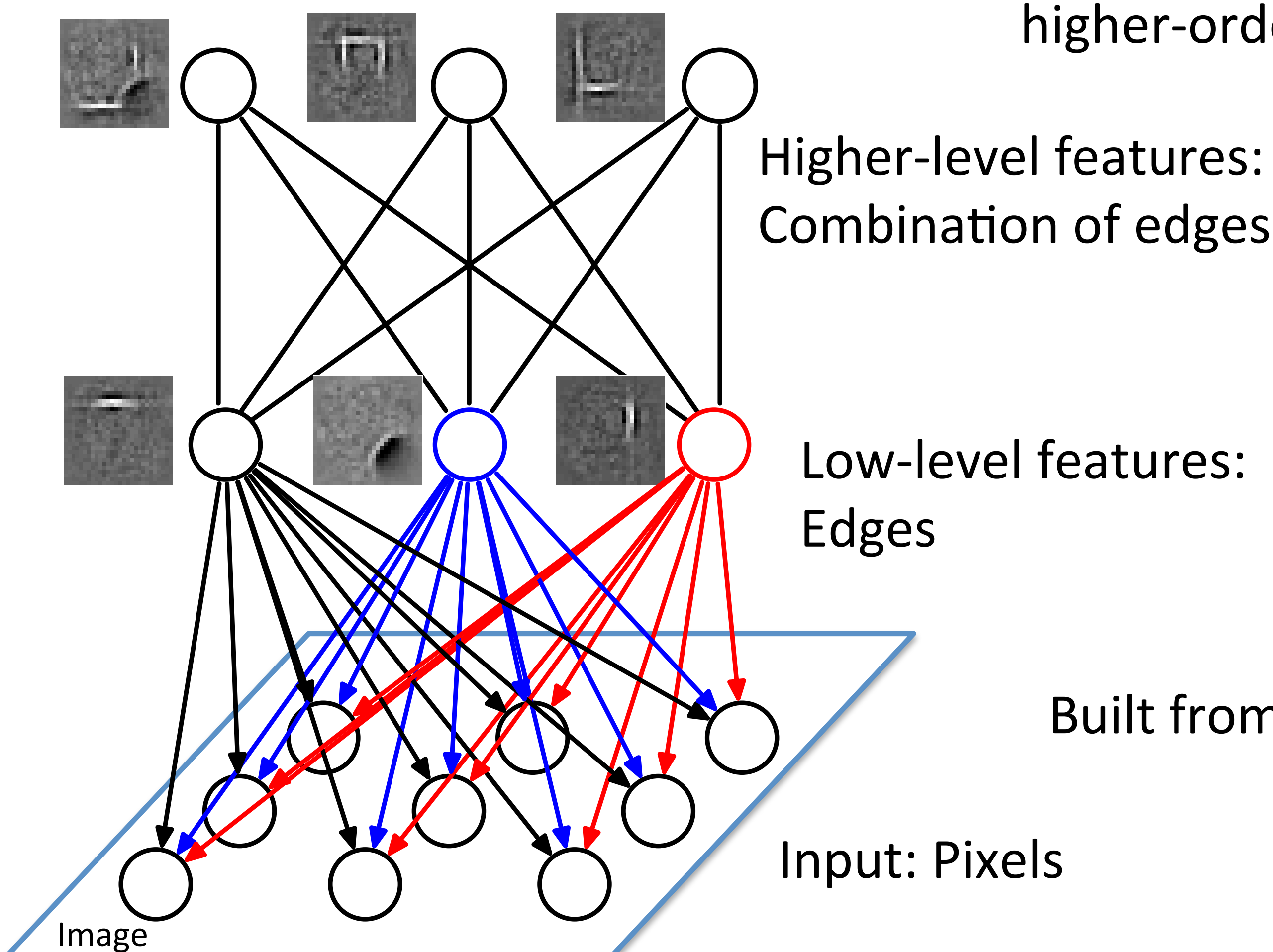
Built from **unlabeled** inputs.

Input: Pixels

(Hinton et.al. Neural Computation 2006)

Deep Belief Network

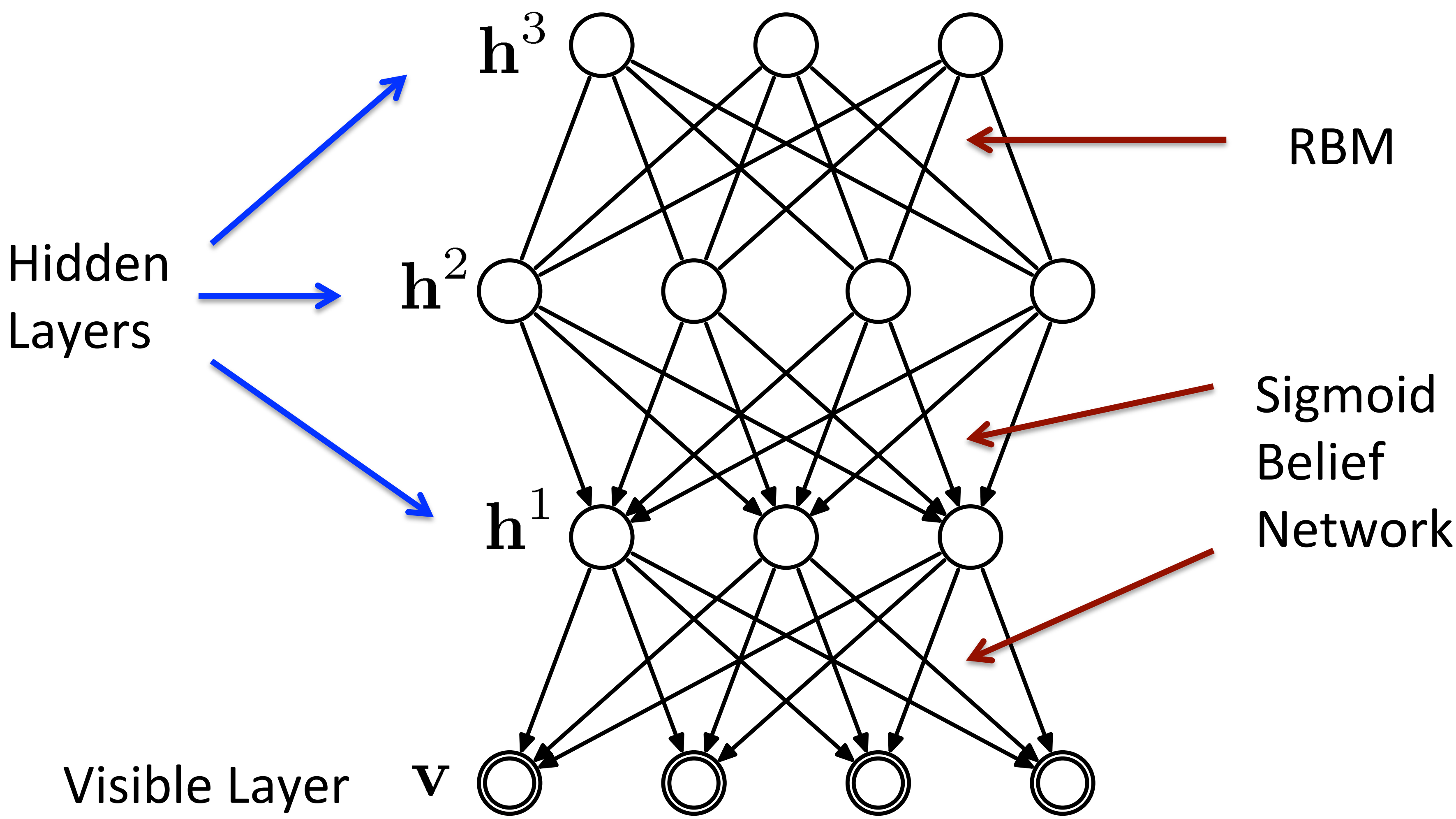
Internal representations capture higher-order statistical structure



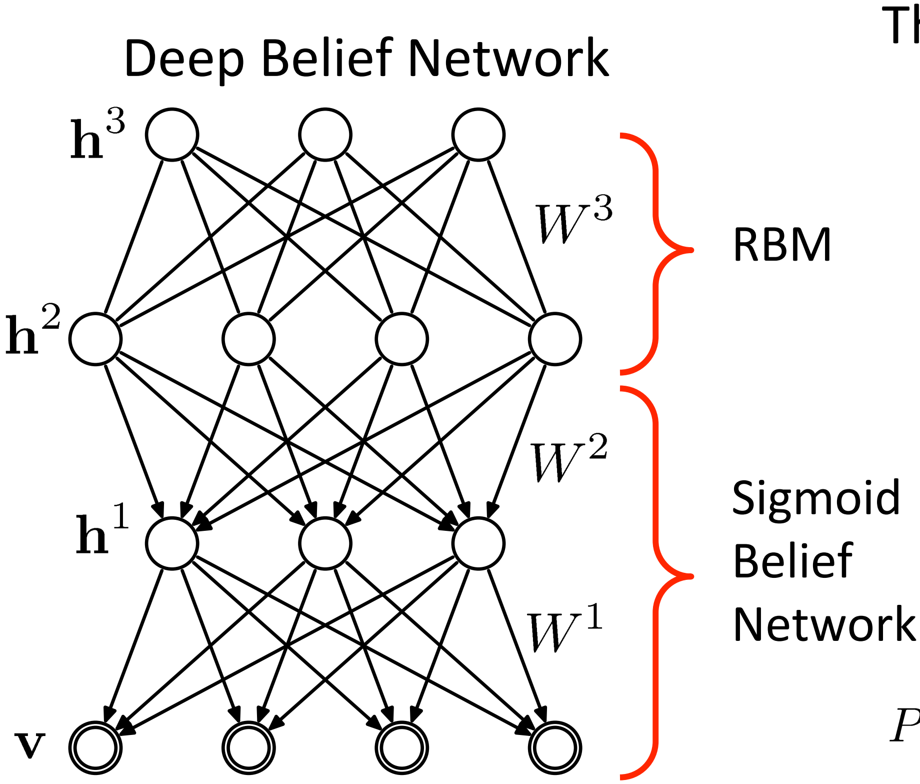
Built from **unlabeled** inputs.

(Hinton et.al. Neural Computation 2006)

Deep Belief Network



Deep Belief Network



The joint probability distribution factorizes:

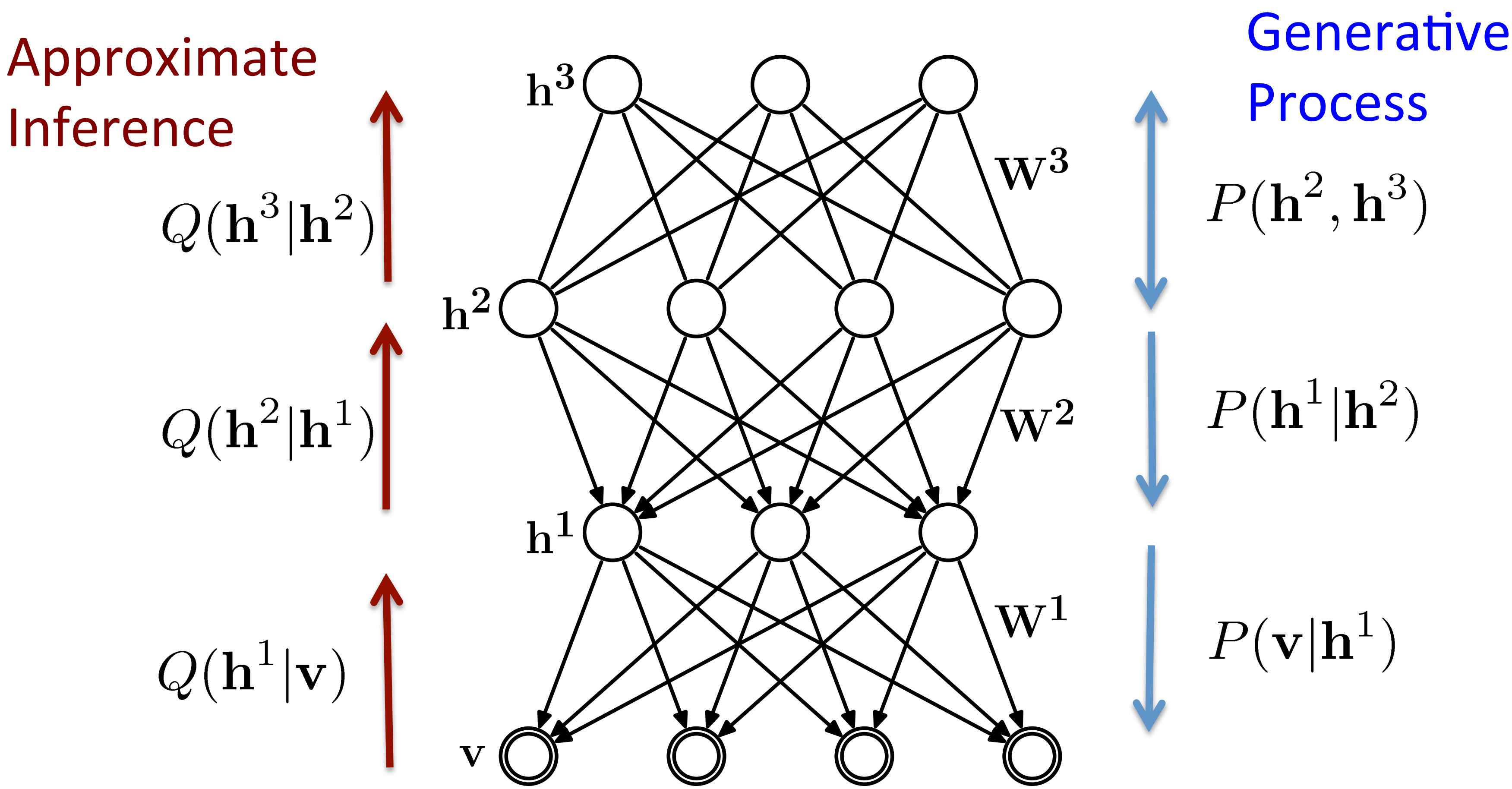
$$P(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2, \mathbf{h}^3) = \underbrace{P(\mathbf{v}|\mathbf{h}^1) P(\mathbf{h}^1|\mathbf{h}^2)}_{\text{Sigmoid Belief Network}} \underbrace{P(\mathbf{h}^2, \mathbf{h}^3)}_{\text{RBM}}$$

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

$$P(\mathbf{h}^1|\mathbf{h}^2) = \prod_j P(h^1_j|\mathbf{h}^2) \qquad P(h^1_j = 1|\mathbf{h}^2) = \frac{1}{1 + \exp\left(-\sum_k W^2_{jk} h^2_k\right)}$$

$$P(\mathbf{v}|\mathbf{h}^1) = \prod_i P(v_i|\mathbf{h}^1) \qquad P(v_i = 1|\mathbf{h}^1) = \frac{1}{1 + \exp\left(-\sum_j W^1_{ij} h^1_j\right)}$$

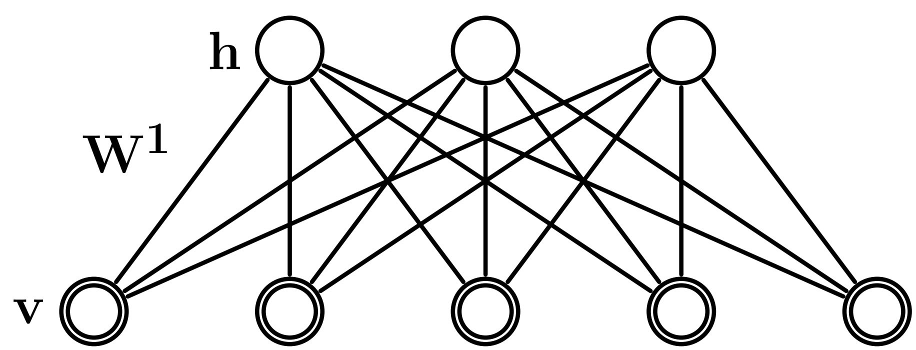
Deep Belief Network



$$Q(\mathbf{h}^t|\mathbf{h}^{t-1}) = \prod_j \sigma\left(\sum_i W^t h_i^{t-1}\right) \qquad P(\mathbf{h}^{t-1}|\mathbf{h}^t) = \prod_j \sigma\left(\sum_i W^t h_i^t\right)$$

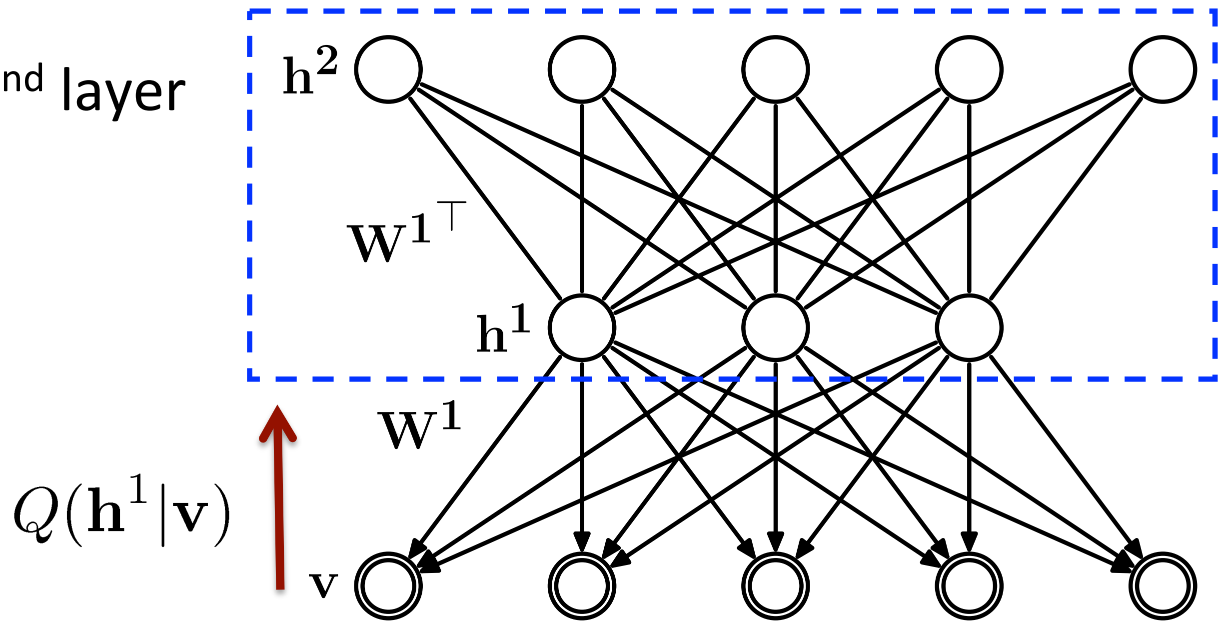
DBN Layer-wise Training

- Learn an RBM with an input layer v and a hidden layer h.



DBN Layer-wise Training

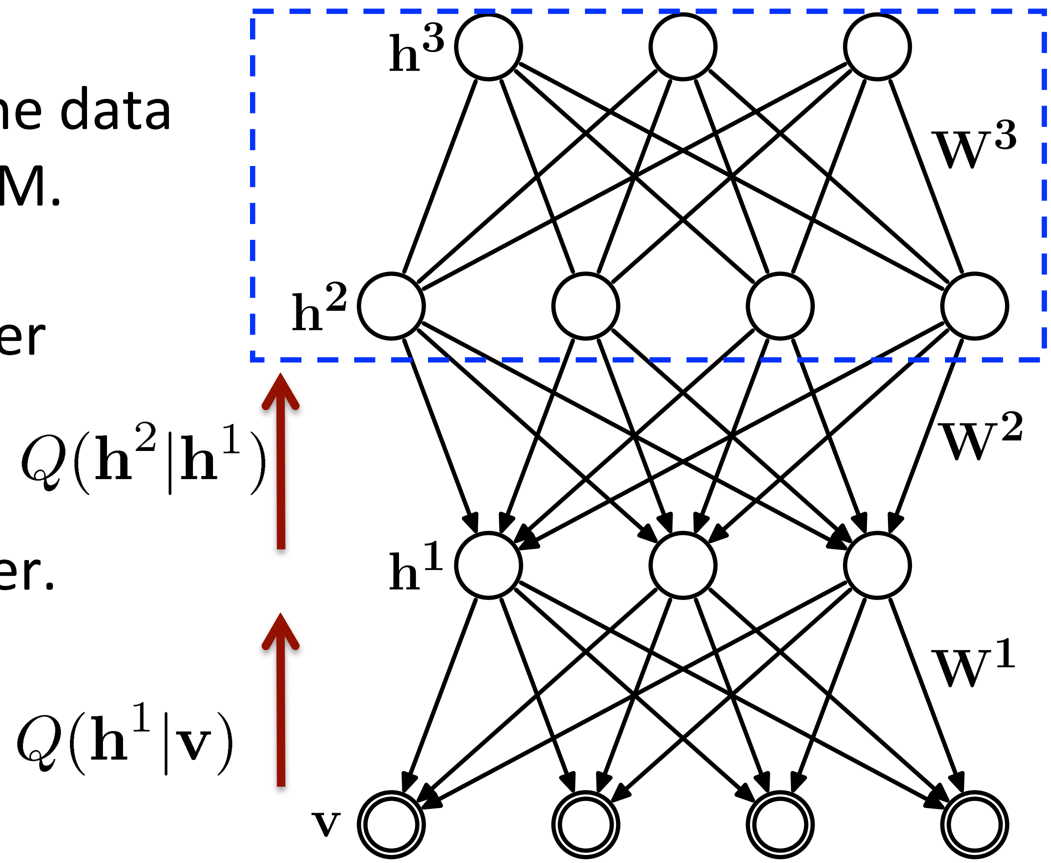
- Learn an RBM with an input layer v and a hidden layer h.
- Treat inferred values $Q(\mathbf{h}^1|\mathbf{v}) = P(\mathbf{h}^1|\mathbf{v})$ as the data for training 2nd-layer RBM.
- Learn and freeze 2nd layer RBM.



DBN Layer-wise Training

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

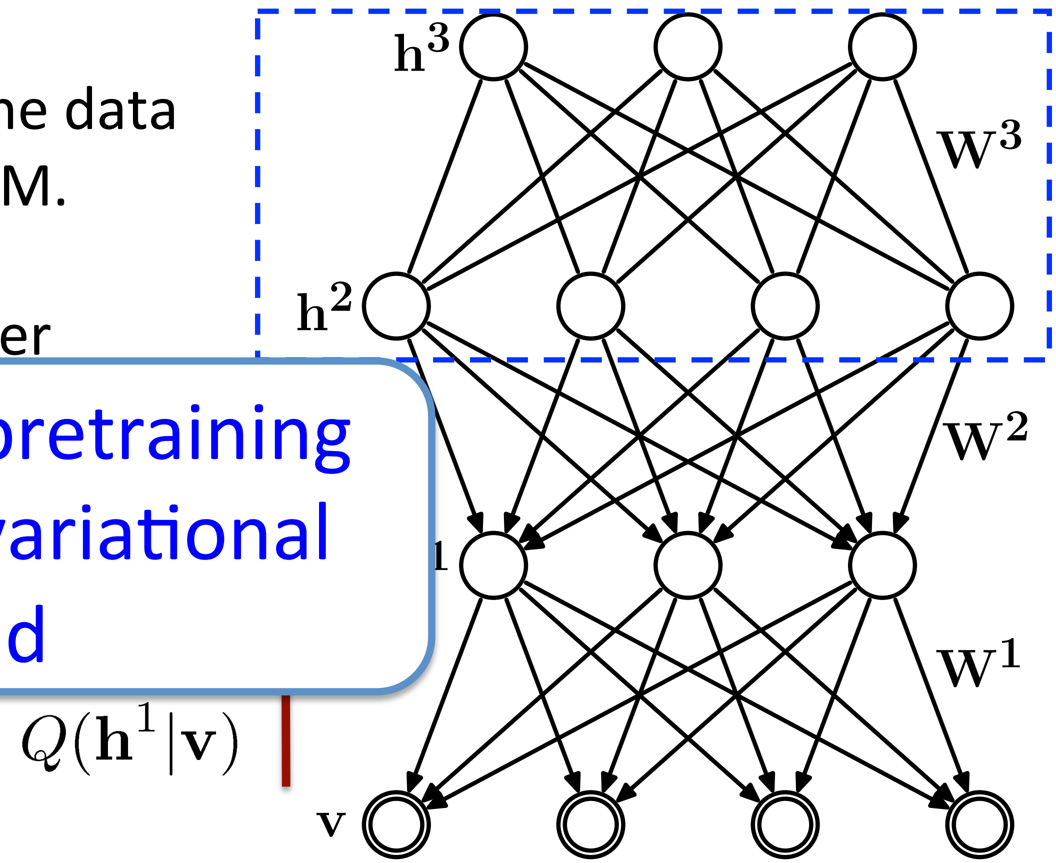
Unsupervised Feature Learning.



DBN Layer-wise Training

- Learn an RBM with an input layer v and a hidden layer h.
- Treat inferred values $Q(\mathbf{h}^1|\mathbf{v}) = P(\mathbf{h}^1|\mathbf{v})$ as the data for training 2nd-layer RBM.
- Learn and freeze 2nd layer RBM.
- Procee

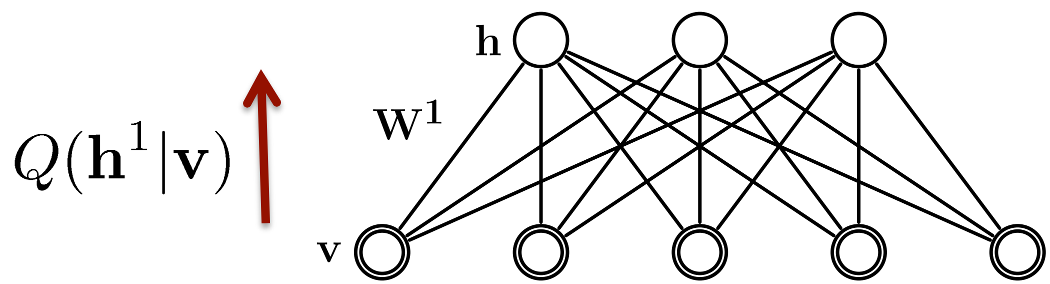
Unsupervised Feature Learning.



Layerwise pretraining improves variational lower bound

Why this Pre-training Works?

- Greedy pre-training improves variational lower bound!



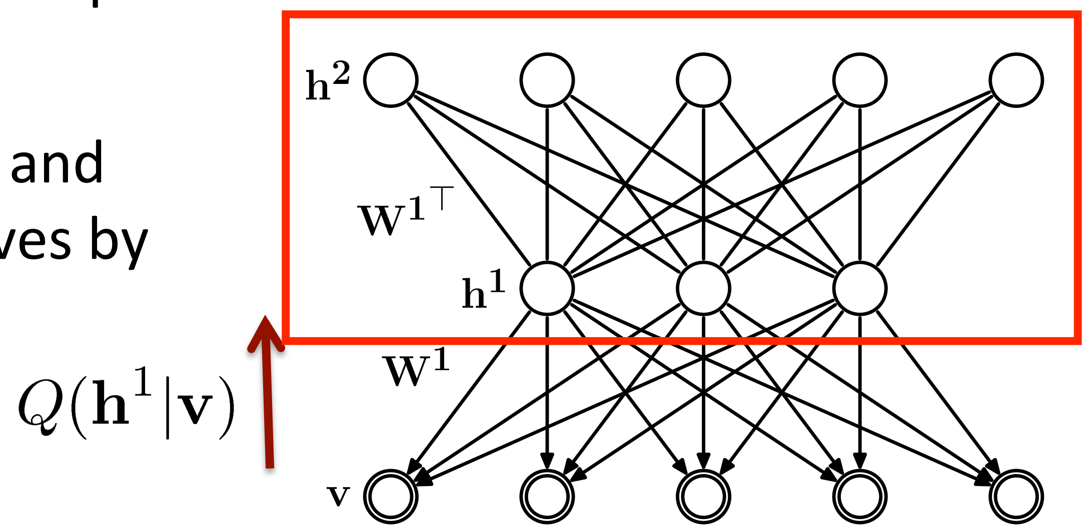
- For any approximating distribution $Q(\mathbf{h}^1|\mathbf{v})$

$$\log P_\theta(\mathbf{v}) = \sum_{\mathbf{h}^1} P_\theta(\mathbf{v}, \mathbf{h}^1)$$

$$\geq \sum_{\mathbf{h}^1} Q(\mathbf{h}^1|\mathbf{v}) \Big[\log P(\mathbf{h}^1) + \log P(\mathbf{v}|\mathbf{h}^1) \Big] + \mathcal{H}(Q(\mathbf{h}^1|\mathbf{v}))$$

Why this Pre-training Works?

- Greedy training improves variational lower bound.
- RBM and 2-layer DBN are equivalent when $W^2 = {W^1}^\top$.
- The lower bound is tight and the log-likelihood improves by greedy training.



- For any approximating distribution $Q(\mathbf{h}^1|\mathbf{v})$

$$\log P_\theta(\mathbf{v}) = \sum_{\mathbf{h}^1} P_\theta(\mathbf{v}, \mathbf{h}^1)$$

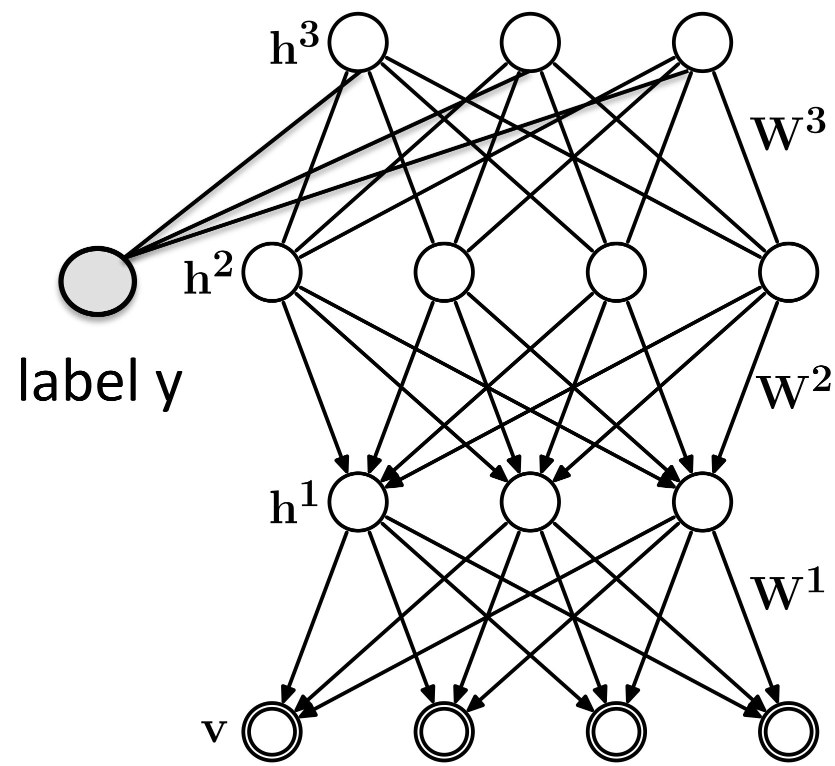
Train 2nd-layer RBM

$$\geq \sum_{\mathbf{h}^1} Q(\mathbf{h}^1|\mathbf{v}) \Big[\log P(\mathbf{h}^1) + \log P(\mathbf{v}|\mathbf{h}^1) \Big] + \mathcal{H}(Q(\mathbf{h}^1|\mathbf{v}))$$

Supervised Learning with DBNs

- If we have access to label information, we can train the joint generative model by maximizing the joint log-likelihood of data and labels

$$\log P(\mathbf{y}, \mathbf{v})$$



- Discriminative fine-tuning:
  - Use DBN to initialize a multilayer neural network.
  - Maximize the conditional distribution:

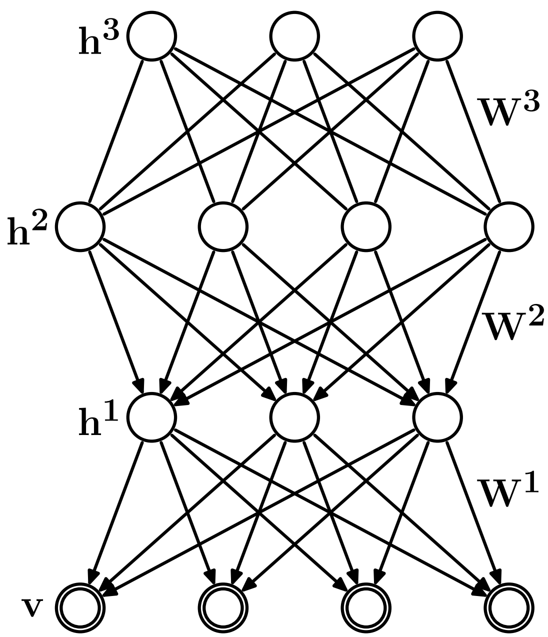
$$\log P(\mathbf{y}|\mathbf{v})$$

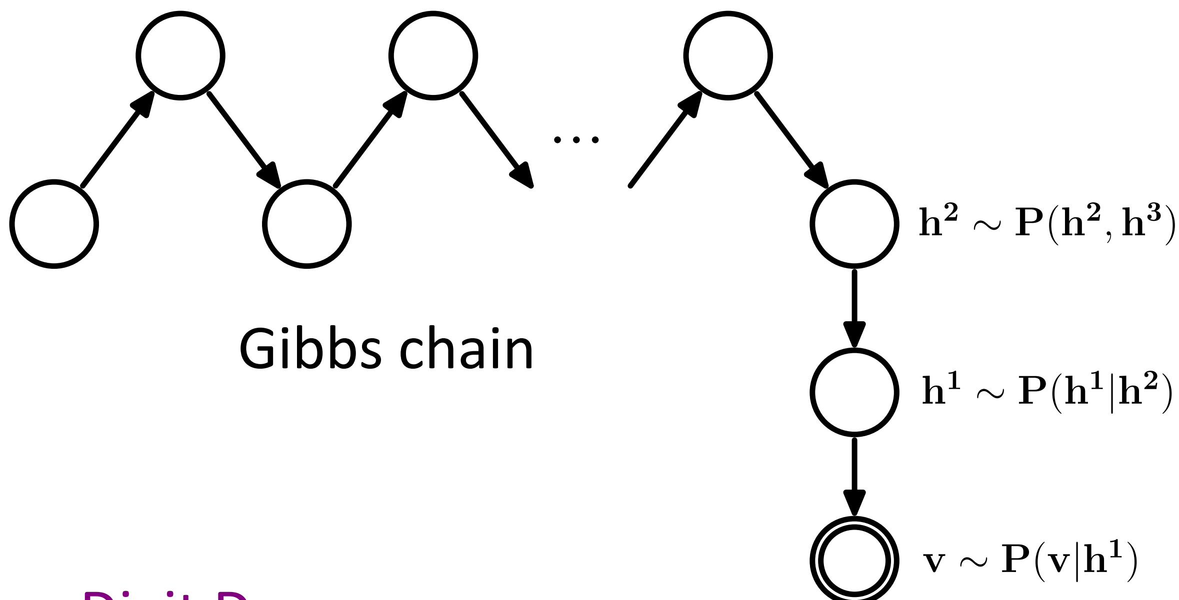
Sampling from DBNs

- To sample from the DBN model:

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

- Sample h^2 using alternating Gibbs sampling from RBM.
- Sample lower layers using sigmoid belief network.

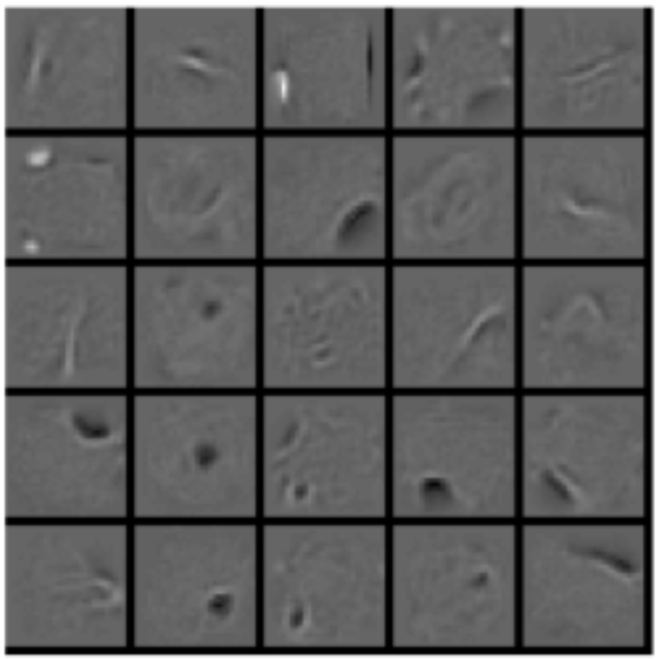




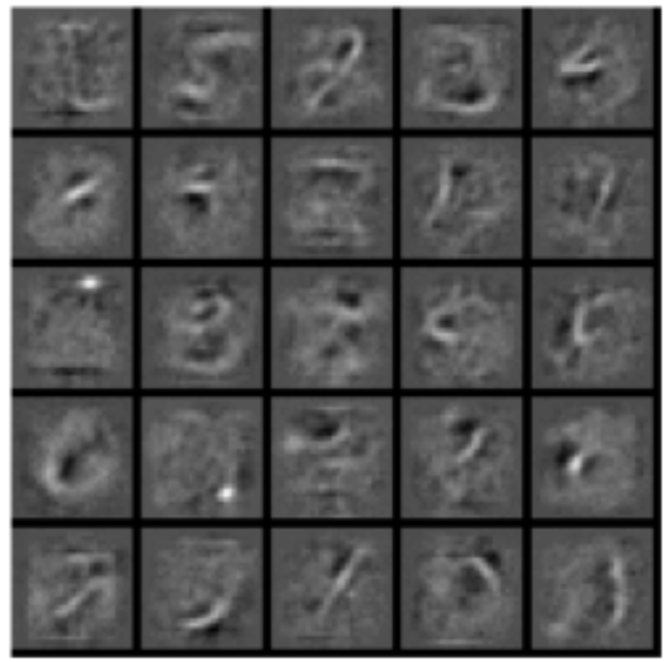
Digit Demo

Learned Features

1^{st}-layer features

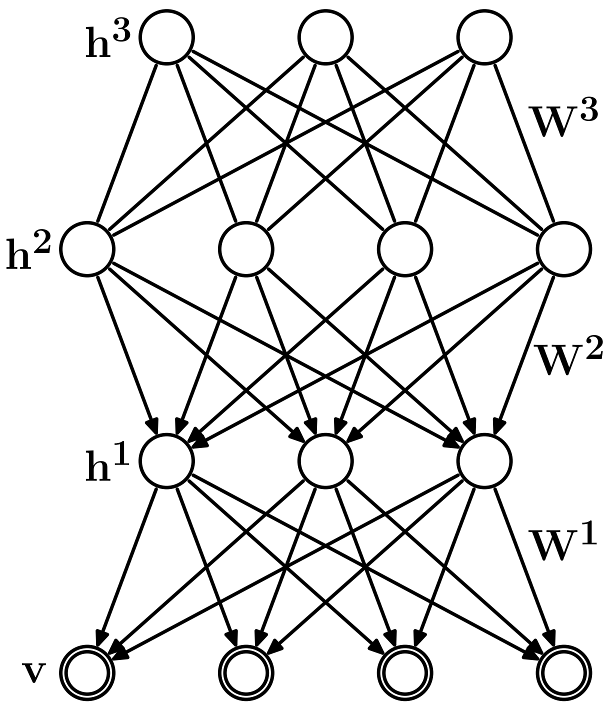


2^{nd}-layer features

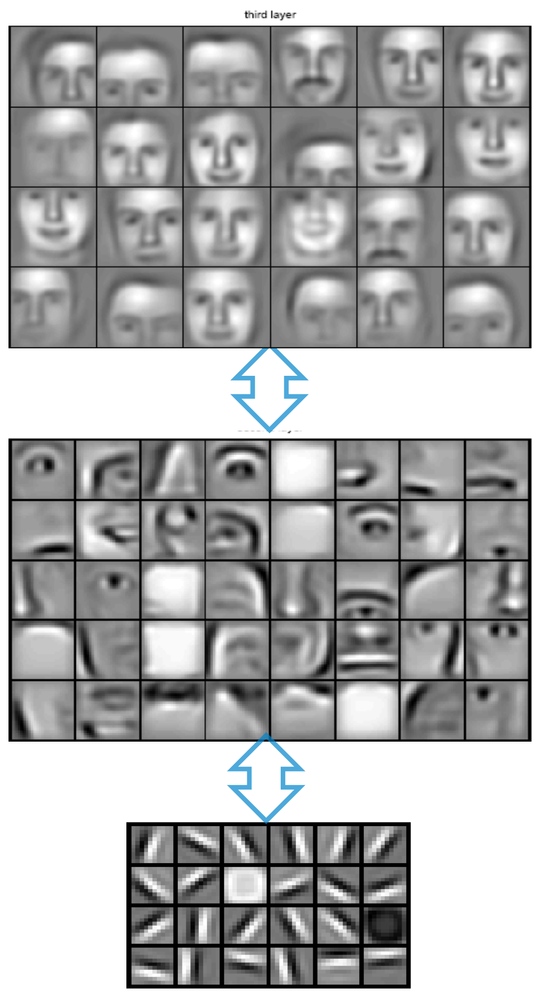


Learning Part-based Representation

Convolutional DBN



Faces



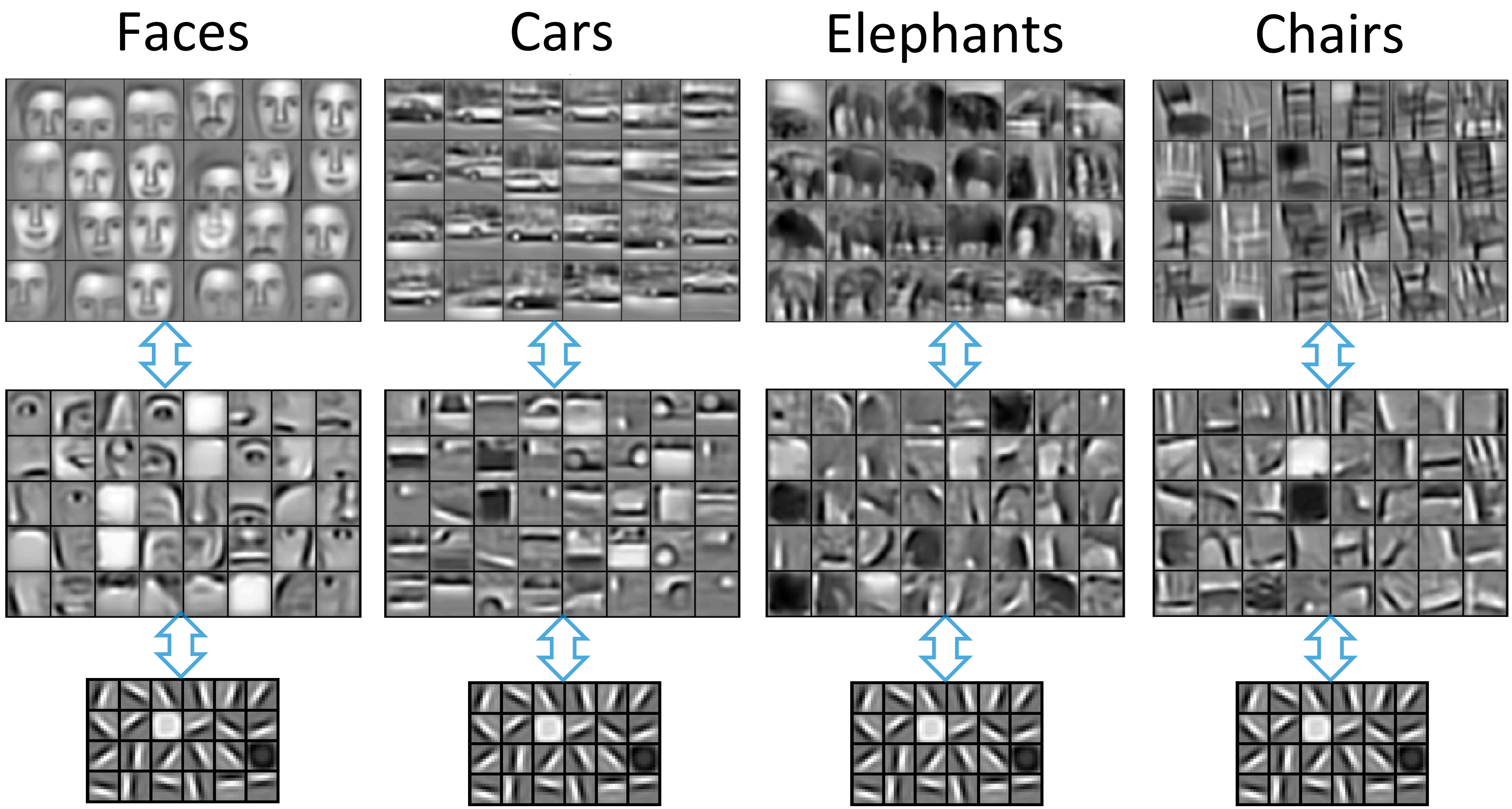
Groups of parts.

Object Parts

Trained on face images.

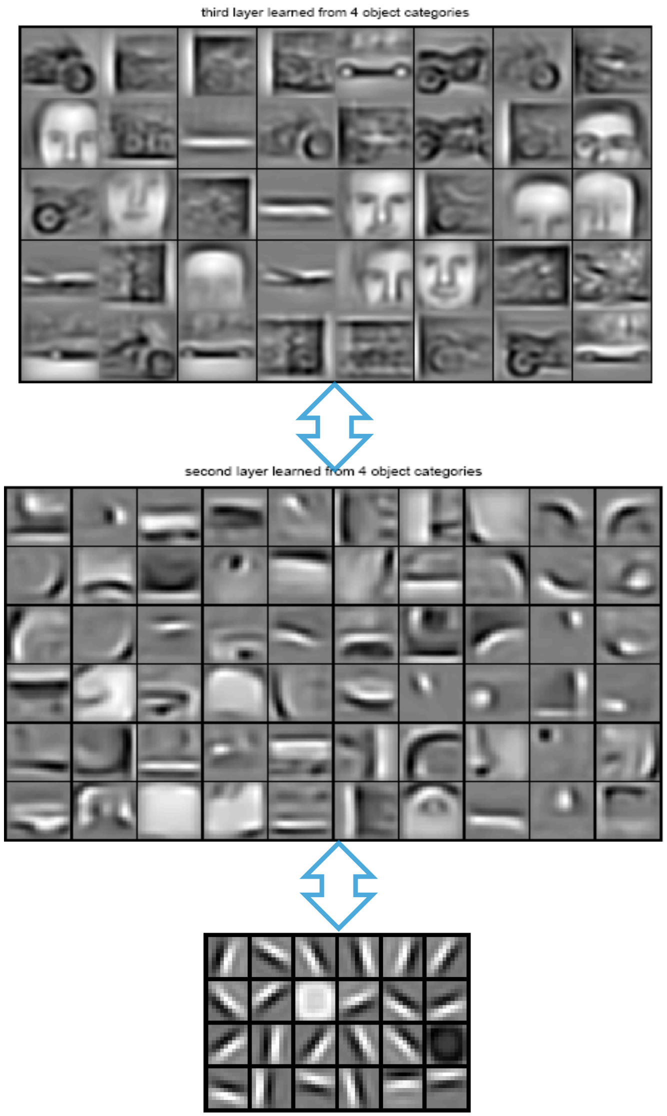
Lee et.al., ICML 2009

Learning Part-based Representation



Lee et.al., ICML 2009

Learning Part-based Representation



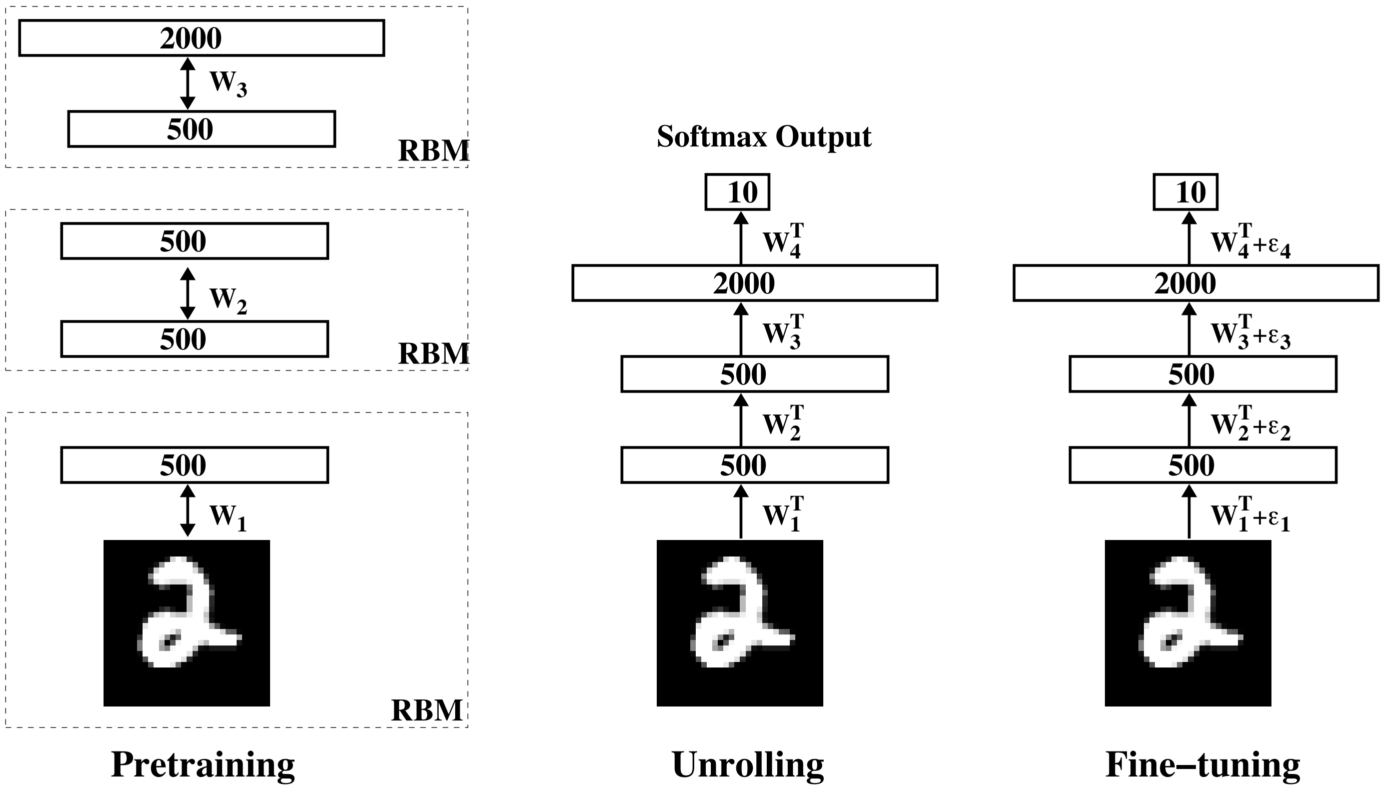
Groups of parts.

Class-specific object parts

Trained from multiple classes (cars, faces, motorbikes, airplanes).

Lee et.al., ICML 2009

DBNs for Classification



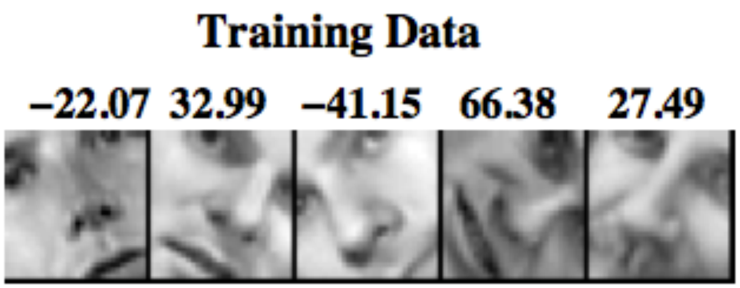
- After layer-by-layer **unsupervised pretraining**, discriminative fine-tuning by backpropagation achieves an error rate of 1.2% on MNIST. SVM's get 1.4% and randomly initialized backprop gets 1.6%.

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

(Hinton and Salakhutdinov, Science 2006)

DBNs for Regression

Predicting the orientation of a face patch



Training Data: 1000 face patches of 30 training people.

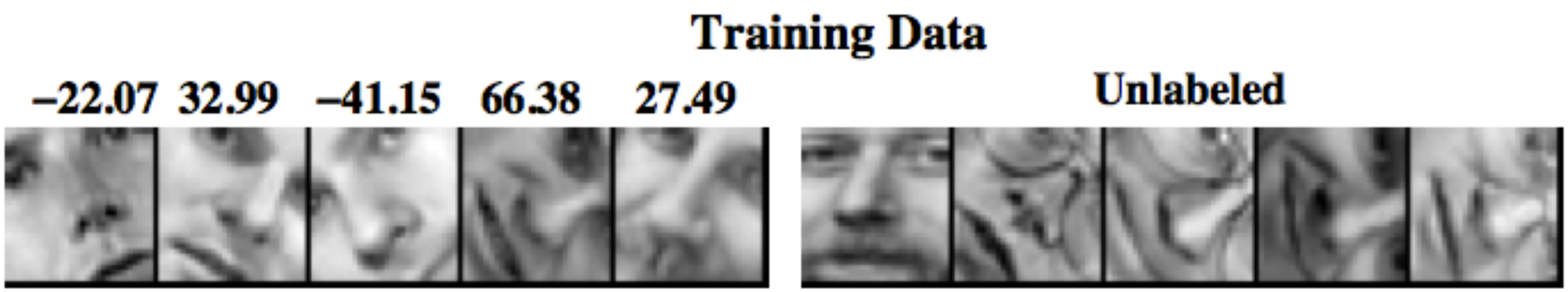
Test Data: 1000 face patches of **10 new people**.

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.

(Salakhutdinov and Hinton, NIPS 2007)

DBNs for Regression

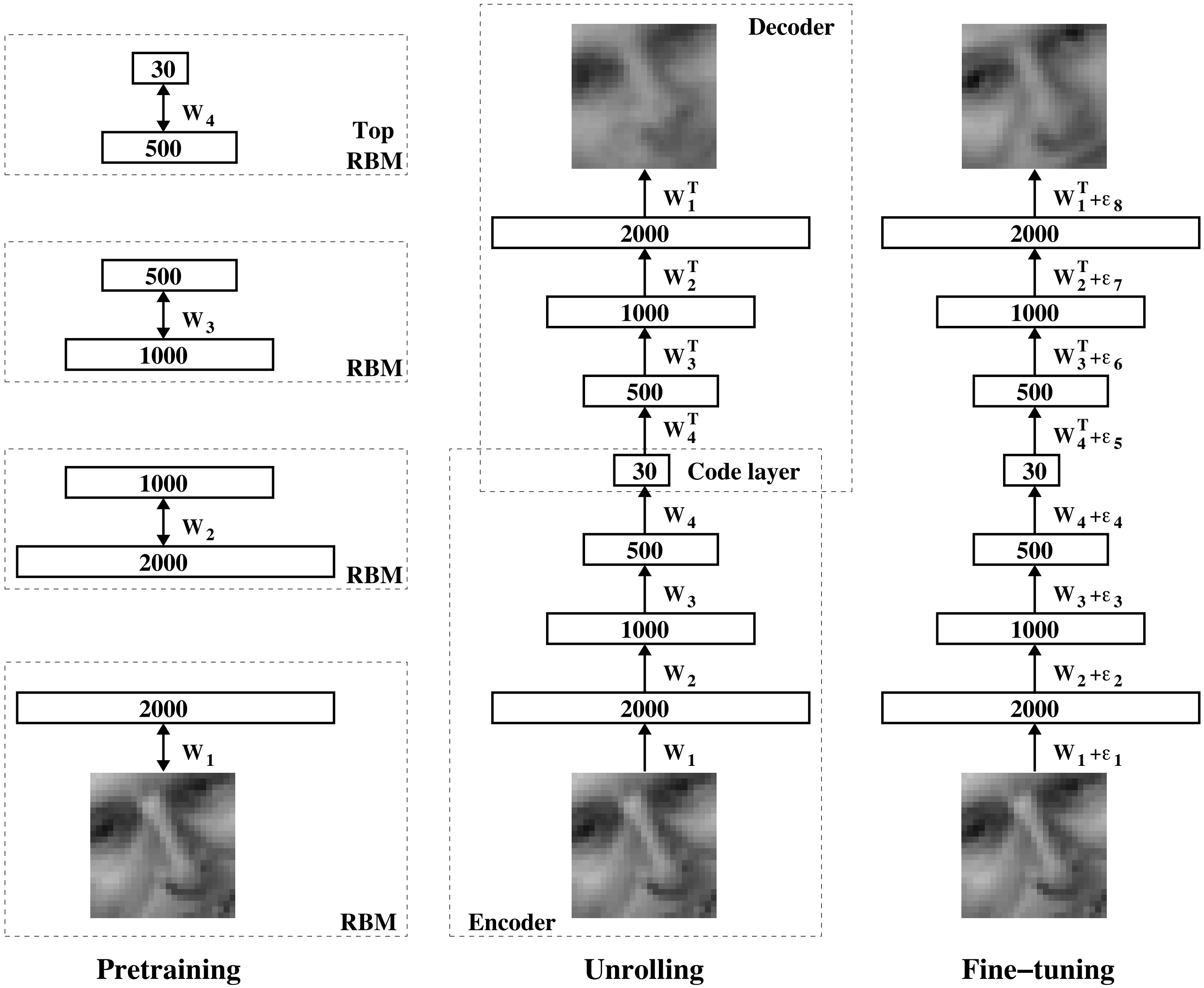


Additional Unlabeled Training Data: 12000 face patches from 30 training people.

- 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

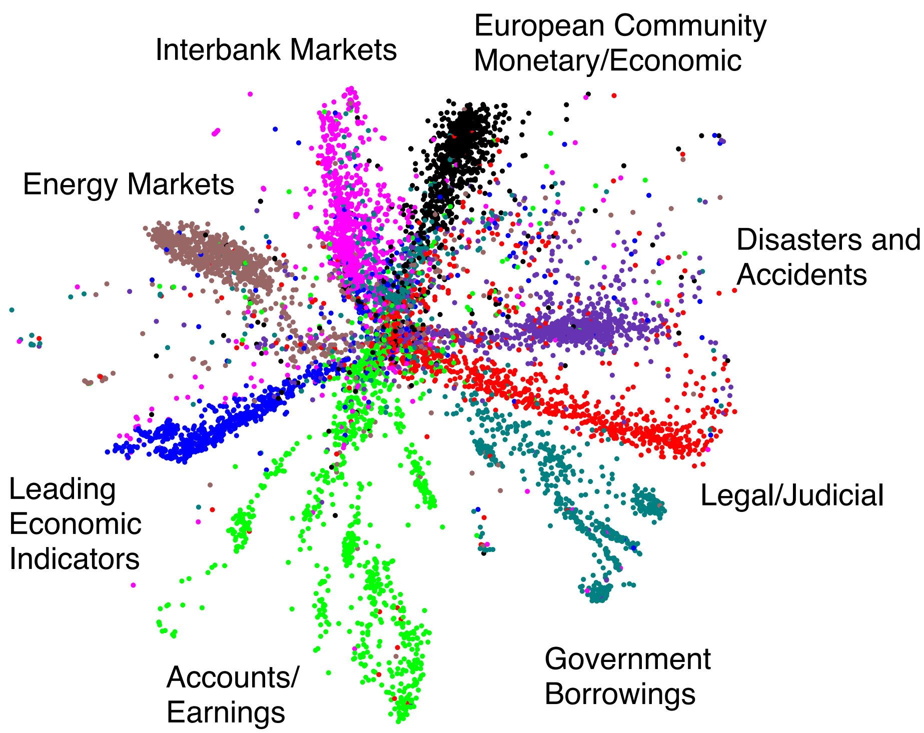


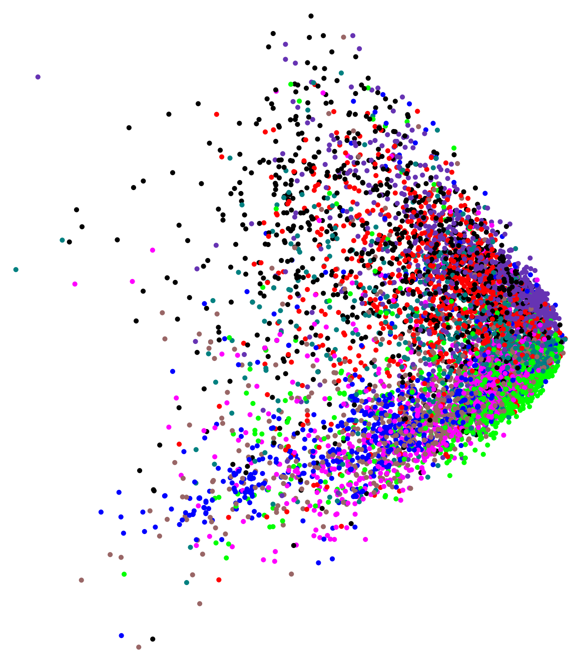
Deep Autoencoders

- We used 25x25 – 2000 – 1000 – 500 – 30 autoencoder to extract 30-D real-valued codes for Olivetti face patches.

- **Top**: Random samples from the test dataset.
- **Middle**: Reconstructions by the 30-dimensional deep autoencoder.
- **Bottom**: Reconstructions by the 30-dimentinoal PCA.

Information Retrieval





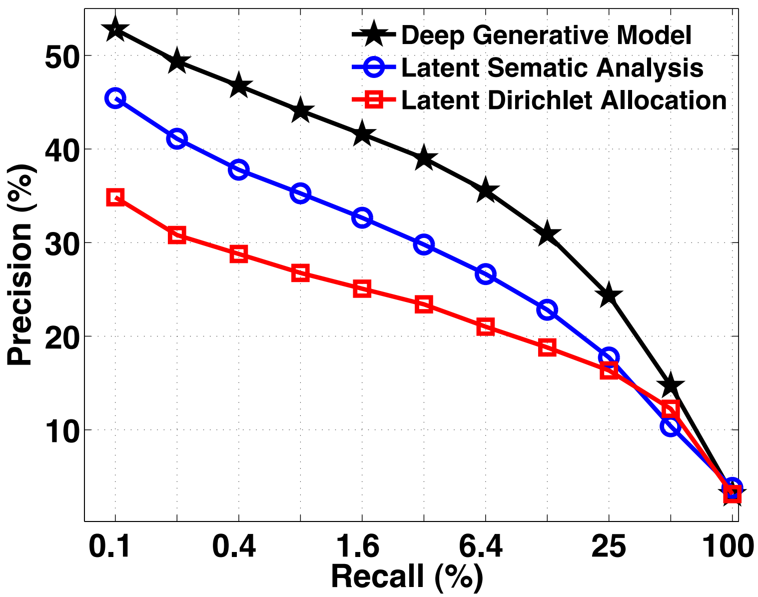
- The Reuters Corpus Volume II contains 804,414 newswire stories (randomly split into **402,207 training** and **402,207 test).**

- “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.

(Hinton and Salakhutdinov, Science 2006)

Information Retrieval

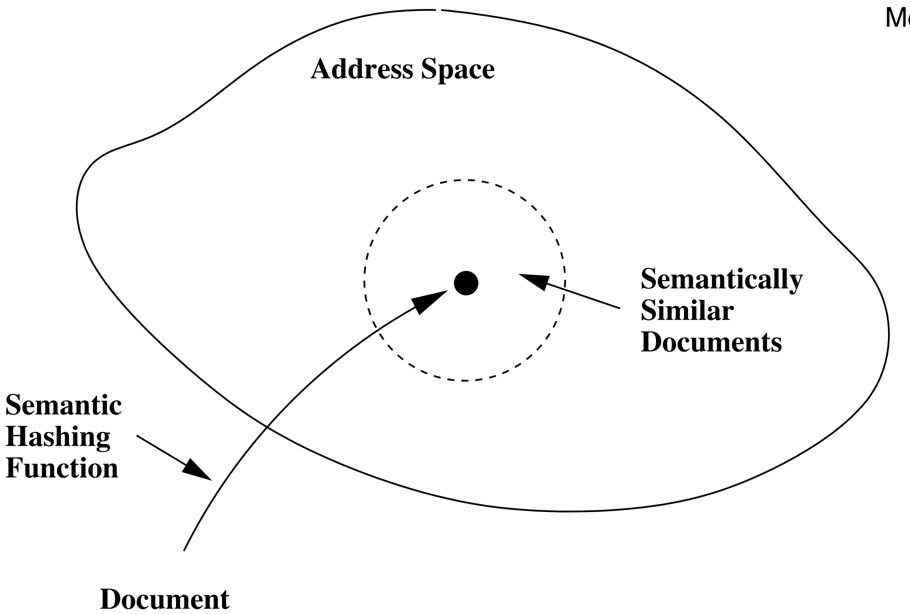
Reuters Dataset

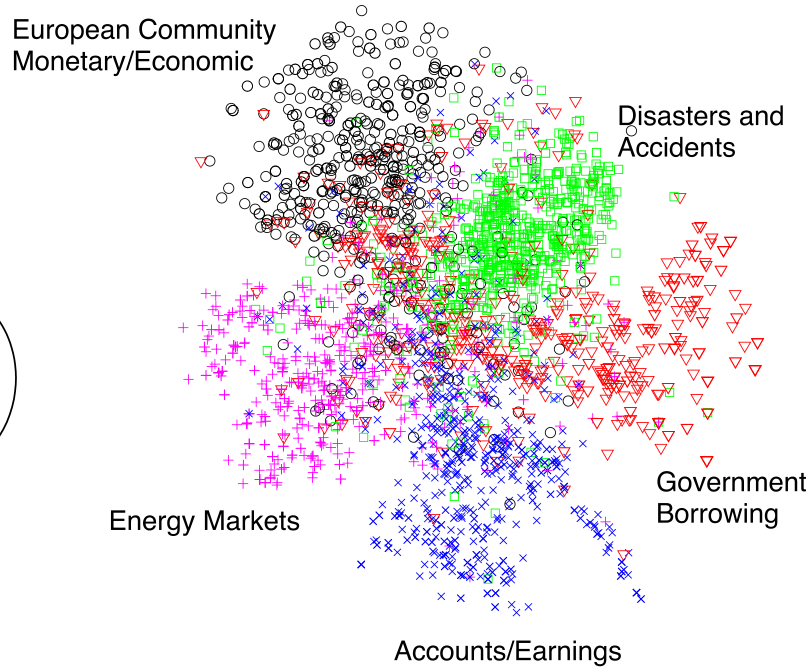


Reuters dataset: 804,414 newswire stories.

Deep generative model significantly outperforms LSA and LDA topic models

Semantic Hashing



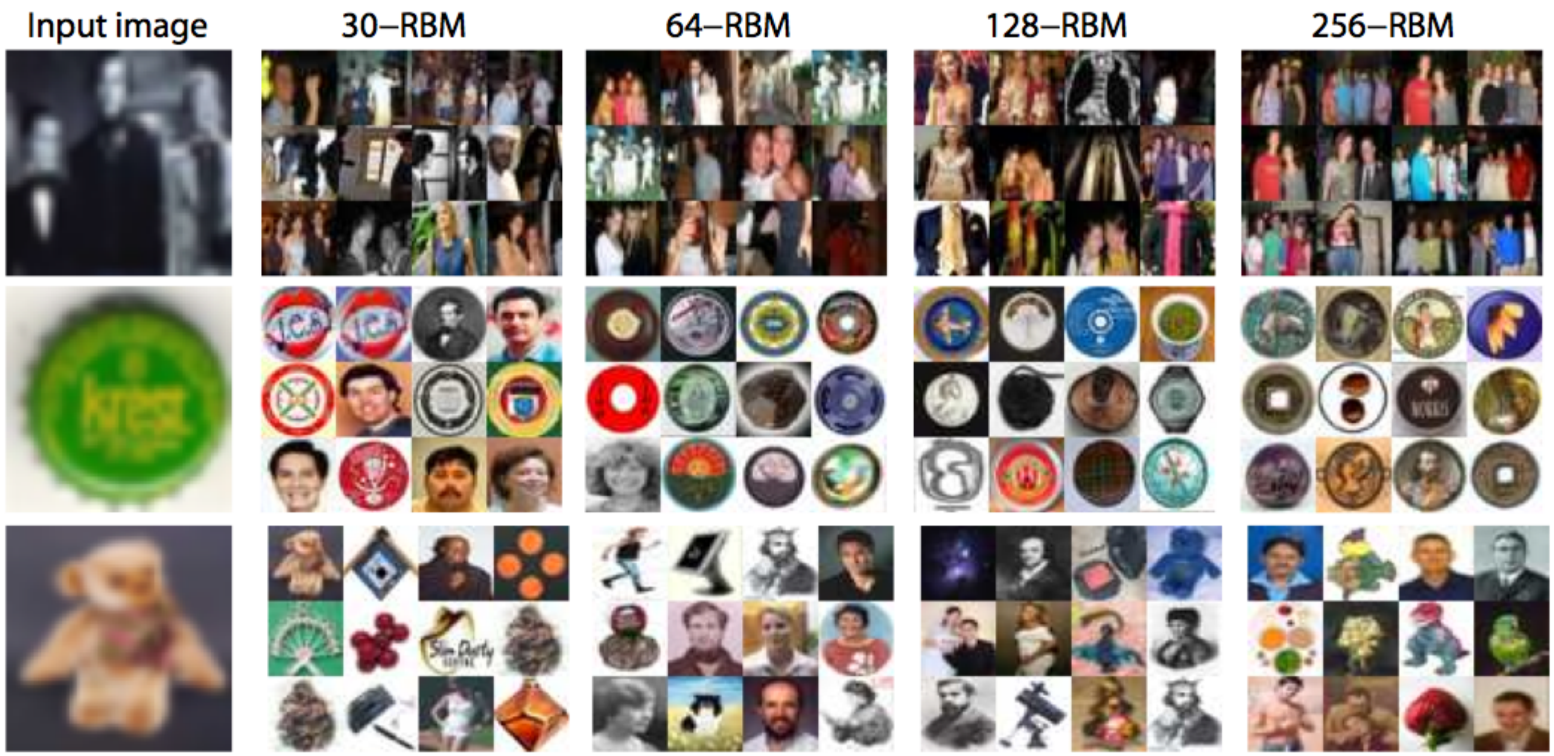


- Learn to map documents into **semantic 20-D binary codes.**
- Retrieve similar documents stored at the nearby addresses **with no search at all.**

(Salakhutdinov and Hinton, SIGIR 2007)

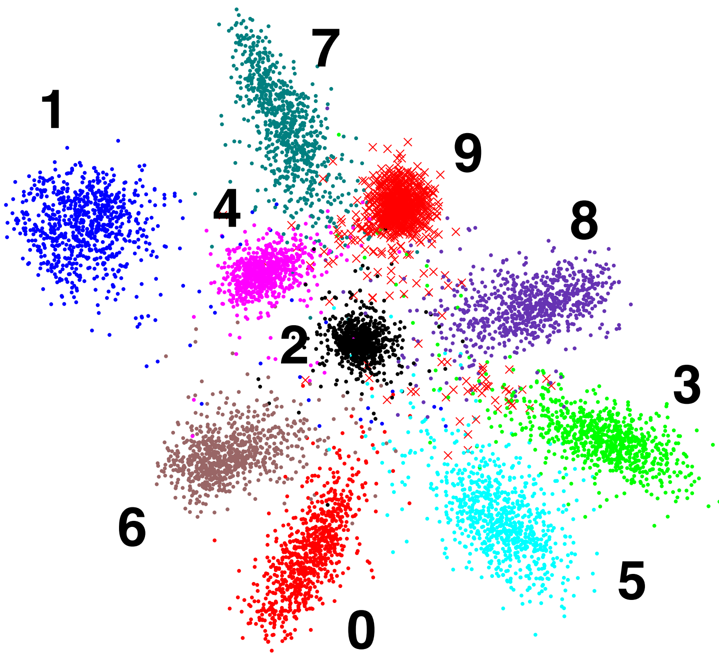
Searching Large Image Database using Binary Codes

- Map images into binary codes for fast retrieval.



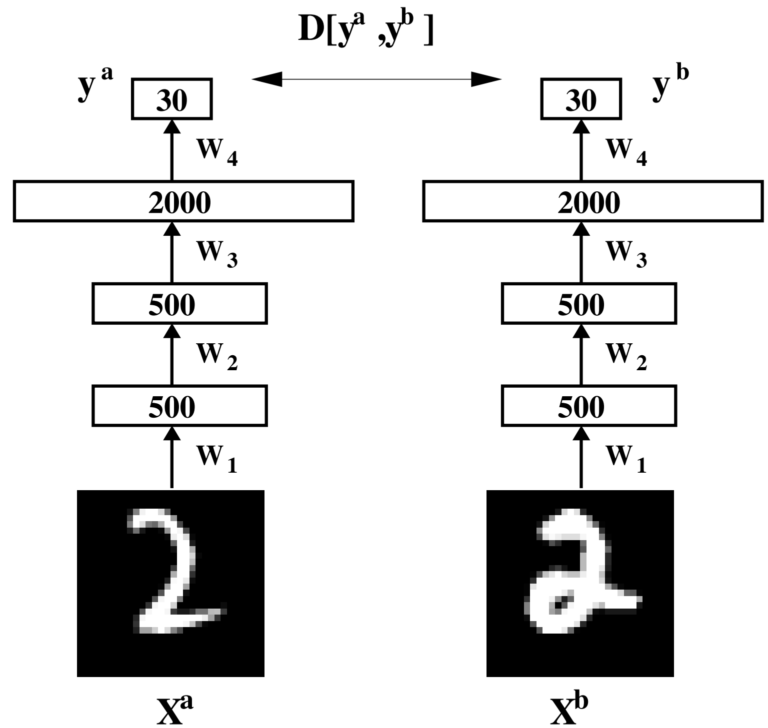
- Small Codes, Torralba, Fergus, Weiss, CVPR 2008
- Spectral Hashing, Y. Weiss, A. Torralba, R. Fergus, NIPS 2008
- Kulis and Darrell, NIPS 2009, Gong and Lazebnik, CVPR 20111
- Norouzi and Fleet, ICML 2011,

Learning Similarity Measures



Related to Siamese Networks of LeCun.

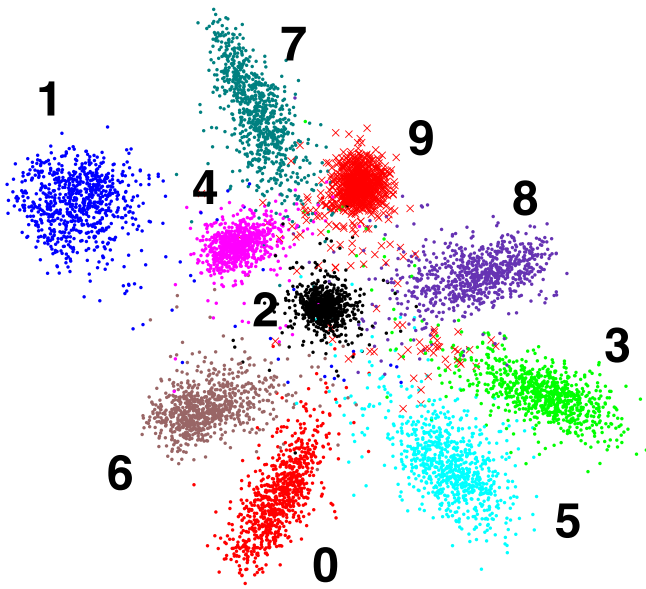
Maximize the Agreement

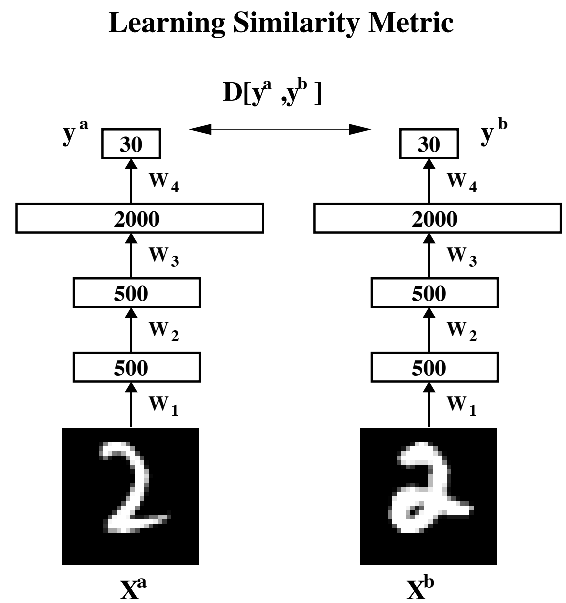


- Learn a nonlinear transformation of the input space.
- Optimize to make KNN perform well in the low-dimensional feature space

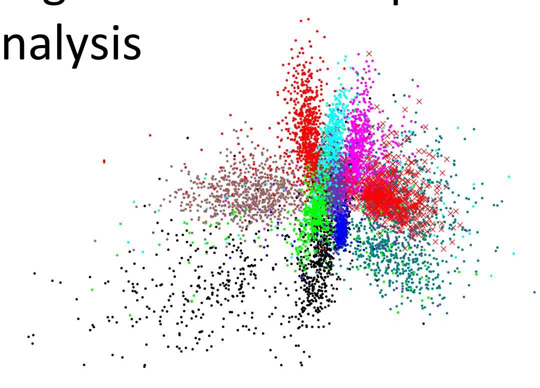
(Salakhutdinov and Hinton, AI and Statistics 2007)

Learning Similarity Measures

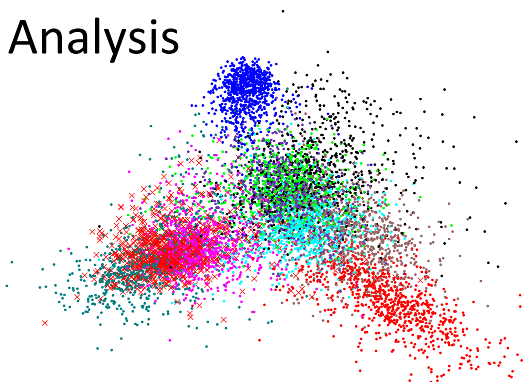




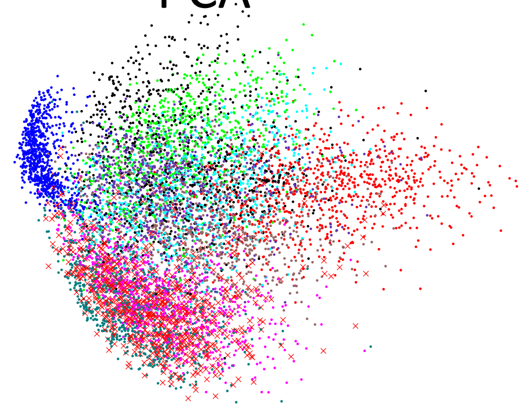
Neighborhood Component Analysis



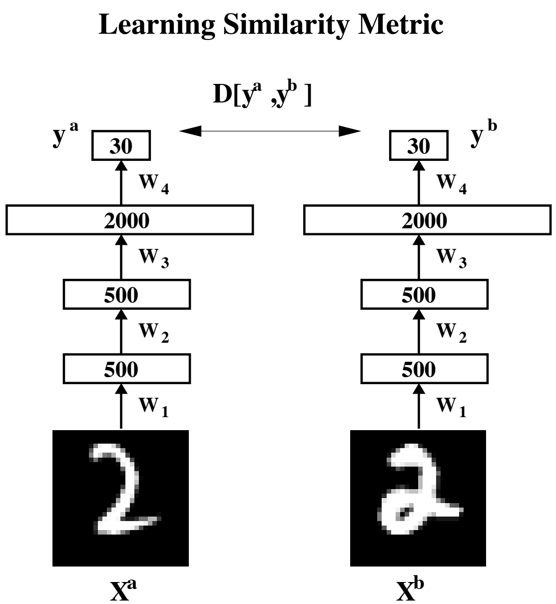
Linear discriminant Analysis

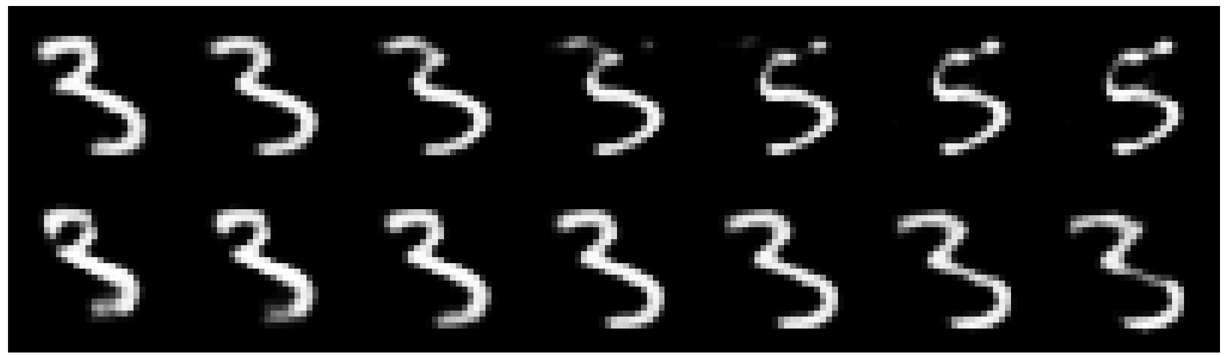


PCA



Learning Similarity Measures



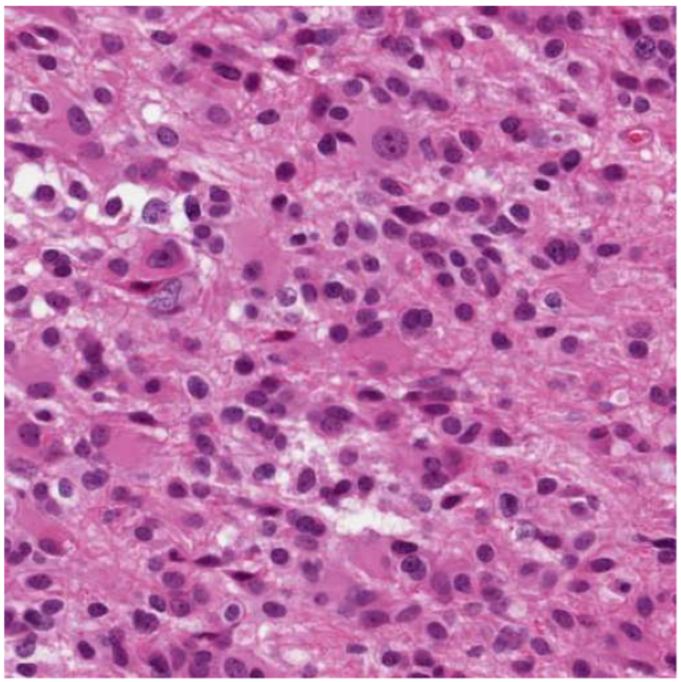


• As we change unit 25 in the code layer, ``3'' image turns into ``5'' image

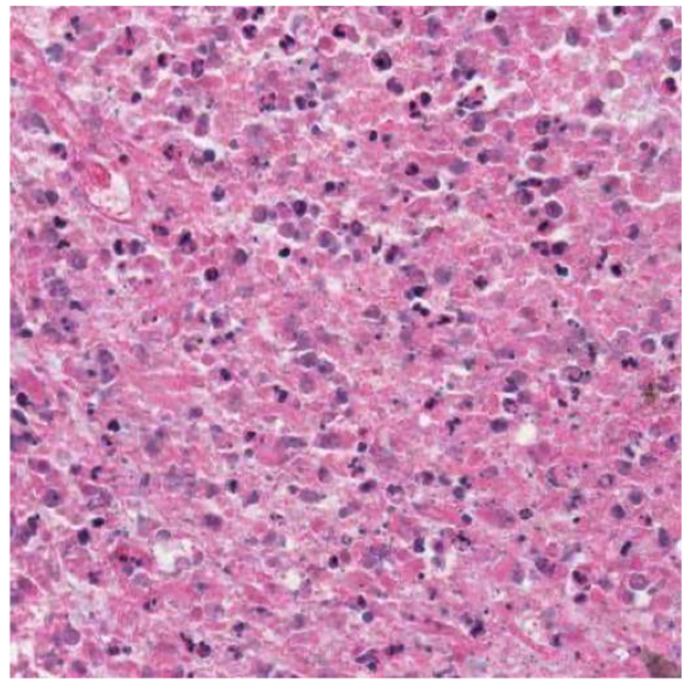
• As we change unit 42 in the code layer, thick ``3'' image turns into skinny ``3''.

Learning Invariant Features of Tumor Signature

A viable tumor region



Mixed necrotic and apoptotic regions

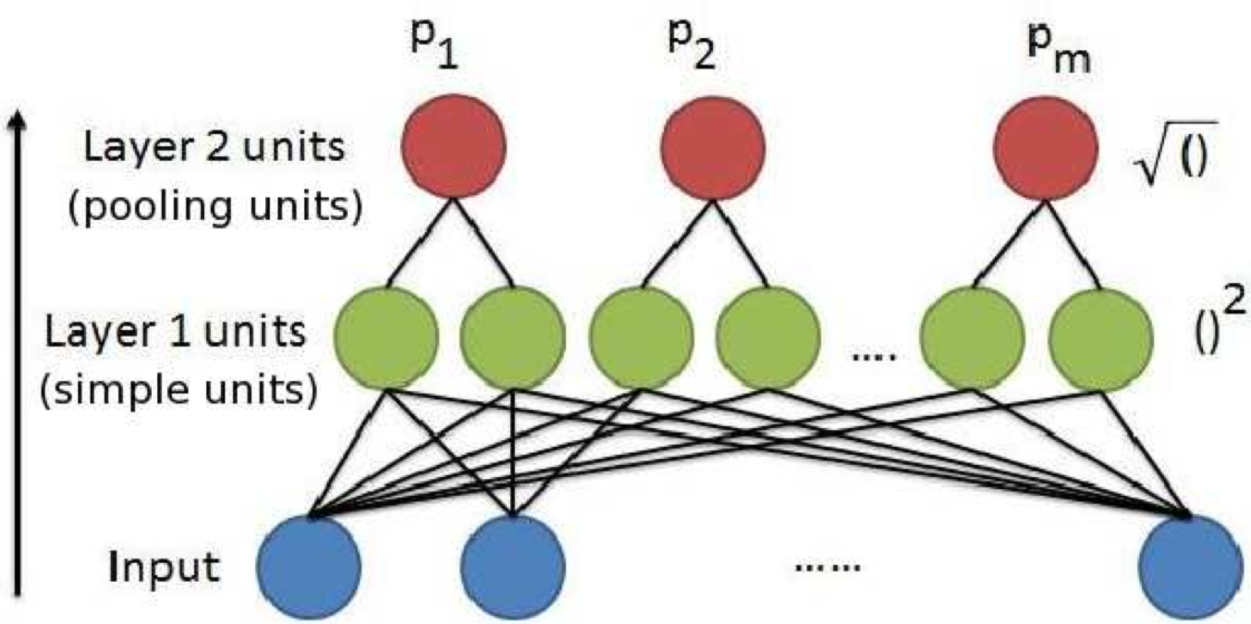


(Le, Han, Spellman, Borowsky, Parvin, ISBI 2012.

Reconstruction Independent Subspace Analysis (RISA)

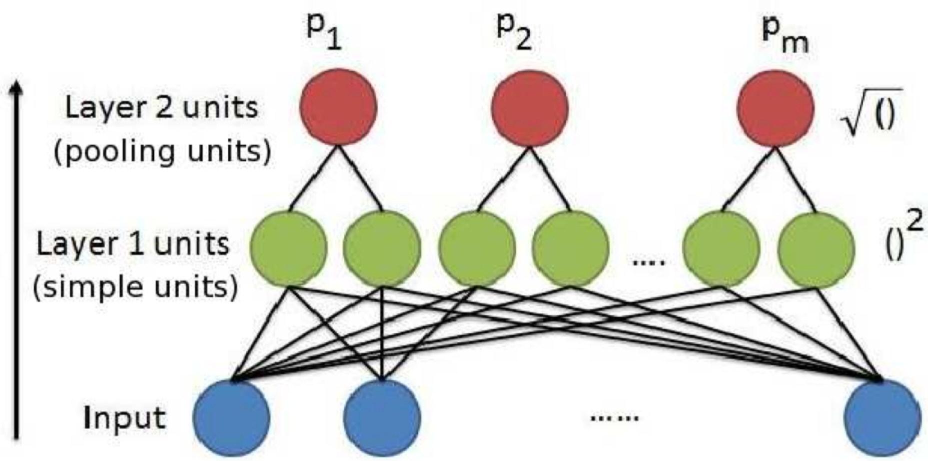
$$p_i(\mathbf{x}; W, V) = \sqrt{\sum_{k=1}^{m} V_{ik} \left(\sum_{j=1}^{d} W_{kj} x_i \right)^2}$$

Total input into the 1st layer.



(Le, Han, Spellman, Borowsky, Parvin, ISBI 2012.

Reconstruction Independent Subspace Analysis (RISA)

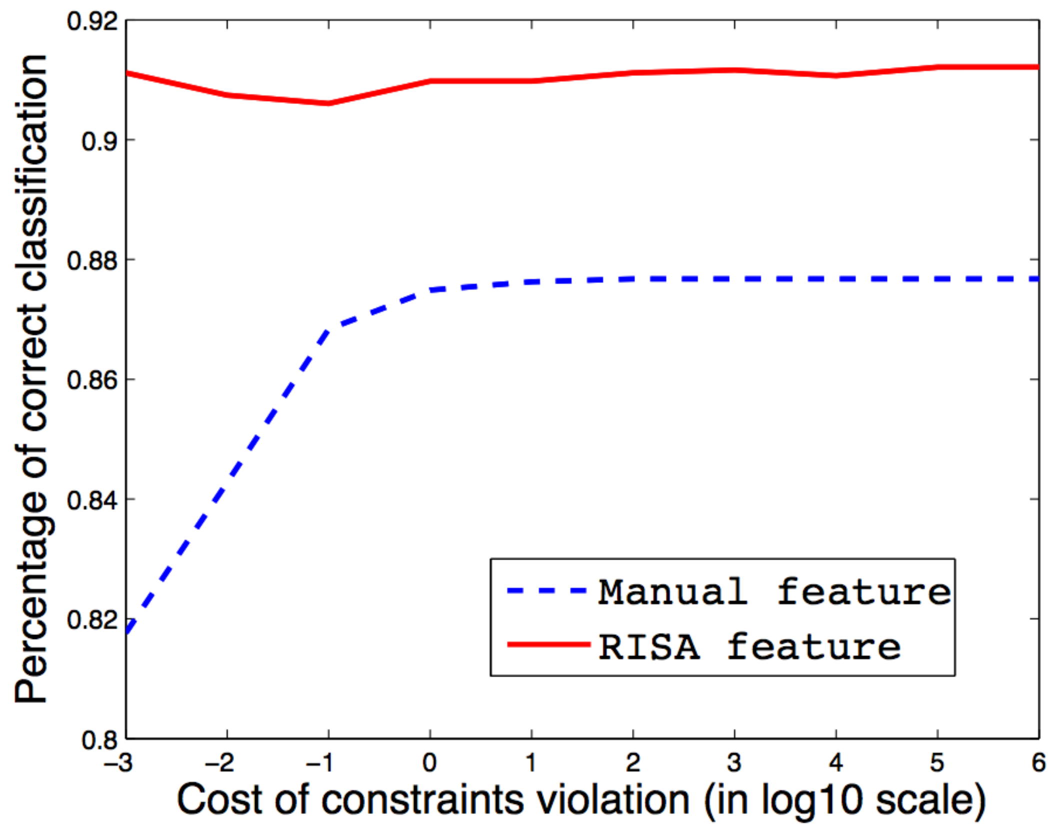


- Given a set of training patches: $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(N)}\}$, we minimize:

$$\min_{W} \sum_{n=1}^{N} \left(\sum_{i=1}^{m} p_i(\mathbf{x}^{(n)}; W, V) + \underbrace{\lambda || W W^\top \mathbf{x}^{(n)} - \mathbf{x}^{(n)} ||^2}_{\text{Reconstruction term}} \right)$$

(Le, Han, Spellman, Borowsky, Parvin, ISBI 2012.

Reconstruction Independent Subspace Analysis (RISA)



• RISA features work much better for classification compared to hand-crafted features.

(Le, Han, Spellman, Borowsky, Parvin, ISBI 2012.