Explicit Density $p(x)$

Unsupervised Learning

Non-probabilistic Models
- Sparse Coding
- Autoencoders
- Others (e.g. k-means)

Probabilistic (Generative) Models

Tractable Models
- Fully observed Belief Nets
- NADE
- PixelRNN

Non-Tractable Models
- Boltzmann Machines
- Variational Autoencoders
- Helmholtz Machines
- Many others...

Generative Adversarial Networks
- Moment Matching Networks

Implicit Density
Unsupervised Learning

• Basic Building Blocks:
  • Sparse Coding
  • Autoencoders

• Deep Generative Models
  • Restricted Boltzmann Machines
  • Deep Boltzmann Machines
  • Deep Belief Networks
  • Helmholtz Machines / Variational Autoencoders

• Generative Adversarial Networks
Deep Generative Model

25,000 characters from 50 alphabets around the world.

- 3,000 hidden variables
- 784 observed variables (28 x 28 images)
- About 2 million parameters

Bernoulli Markov Random Field
Deep Generative Model

Conditional Simulation

Why so difficult?

28

2^{28 \times 28} possible images!

P(image | partial image)

Bernoulli Markov Random Field
Fully Observed Models

• Explicitly model conditional probabilities:

\[
p_{\text{model}}(\mathbf{x}) = p_{\text{model}}(x_1) \prod_{i=2}^{n} p_{\text{model}}(x_i | x_1, \ldots, x_{i-1})
\]
Fully Observed Models

• Explicitly model conditional probabilities:

\[
p_{\text{model}}(\mathbf{x}) = p_{\text{model}}(x_1) \prod_{i=2}^{n} p_{\text{model}}(x_i \mid x_1, \ldots, x_{i-1})
\]

Each conditional can be a complicated neural network.
Fully Observed Models

- Explicitly model conditional probabilities:
  \[ p_{\text{model}}(\mathbf{x}) = p_{\text{model}}(x_1) \prod_{i=2}^{n} p_{\text{model}}(x_i | x_1, \ldots, x_{i-1}) \]

- A number of successful models, including
  - NADE, RNADE (Larochelle, et.al. 2011)
  - Pixel CNN (van den Ord et. al. 2016)
  - Pixel RNN (van den Ord et. al. 2016)
Restricted Boltzmann Machines

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

RBM is a Markov Random Field with:

• Stochastic binary visible variables $\mathbf{v} \in \{0, 1\}^D$.
• Stochastic binary hidden variables $\mathbf{h} \in \{0, 1\}^F$.
• Bipartite connections.

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

RBM is a Markov Random Field with:

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

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

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

Partition function (intractable)

Markov random fields, Boltzmann machines, log-linear models.
**Restricted Boltzmann Machines**

RBM is a Markov Random Field with:

- Stochastic binary visible variables \( \mathbf{v} \in \{0, 1\}^D \).
- Stochastic binary hidden variables \( \mathbf{h} \in \{0, 1\}^F \).
- Bipartite connections.

The probability distribution of the RBM is given by:

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

where \( \theta = \{W, a, b\} \).

The conditional probability distributions are:

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

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

Observed Data
Subset of 25,000 characters

New Image: \( p(h_7 = 1|v) \)

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

Logistic Function: Suitable for modeling binary images

\[ p(h_7 = 1|v) = \sigma(0.99 \times \text{image}) \]

\[ p(h_{29} = 1|v) = 0.97 \times \text{image} \]

\[ + 0.82 \times \text{image} \]

Sparse representations

Learned W: “edges”
Subset of 1000 features
Given a set of i.i.d. training examples $D = \{v^{(1)}, v^{(2)}, ..., v^{(N)}\}$ we want to learn $\theta = \{W, a, b\}$. 

$$P_\theta(v) = \frac{P^*(v)}{Z(\theta)} = \frac{1}{Z(\theta)} \sum_h \exp \left[ v^T W h + a^T h + b^T v \right]$$
Model Learning

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

Maximize log-likelihood objective:

\[
L(\theta) = \frac{1}{N} \sum_{n=1}^{N} \log P_{\theta}(\mathbf{v}^{(n)})
\]

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

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

Maximize log-likelihood objective:

$$L(\theta) = \frac{1}{N} \sum_{n=1}^{N} \log P_\theta(\mathbf{v}^{(n)})$$

Derivative of the log-likelihood:

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

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

\[ P_\theta(v) = \frac{P^*(v)}{Z(\theta)} = \frac{1}{Z(\theta)} \sum_h \exp \left[ v^T W h + a^T h + b^T v \right] \]

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

Maximize log-likelihood objective:

\[ L(\theta) = \frac{1}{N} \sum_{n=1}^{N} \log P_\theta(v^{(n)}) \]

\[ \frac{\partial L(\theta)}{\partial W_{ij}} = \frac{1}{N} \sum_{n=1}^{N} \frac{\partial}{\partial W_{ij}} \log \left( \sum_h \exp \left[ v^{(n)^T} W h + a^T h + b^T v^{(n)} \right] \right) - \frac{\partial}{\partial W_{ij}} \log Z(\theta) \]

\[ = E_{P_{data}}[v_i h_j] - E_{P_\theta}[v_i h_j]. \]

\[ P_{data}(v, h; \theta) = P(h|v; \theta) P_{data}(v) \]

\[ P_{data}(v) = \frac{1}{N} \sum_n \delta(v - v^{(n)}) \]
Model Learning

Difficult to compute: Exponentially many configurations

\[ P_\theta(v) = \frac{P^*(v)}{Z(\theta)} = \frac{1}{Z(\theta)} \sum_h \exp \left[ v^T W h + a^T h + b^T v \right] \]

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

Maximize log-likelihood objective:

\[ L(\theta) = \frac{1}{N} \sum_{n=1}^{N} \log P_\theta(v^{(n)}) \]

Derivative of the log-likelihood:

\[ \frac{\partial L(\theta)}{\partial W_{ij}} = \frac{1}{N} \sum_{n=1}^{N} \frac{\partial}{\partial W_{ij}} \log \left( \sum_h \exp \left[ v^{(n)^T} W h + a^T h + b^T v^{(n)} \right] \right) - \frac{\partial}{\partial W_{ij}} \log Z(\theta) \]

\[ = E_{P_{\text{data}}} [v_i h_j] - E_{P_\theta} [v_i h_j] \]

\[ P_{\text{data}}(v, h; \theta) = P(h|v; \theta) P_{\text{data}}(v) \]

\[ P_{\text{data}}(v) = \frac{1}{N} \sum_n \delta(v - v^{(n)}) \]

Difficult to compute: Exponentially many configurations
Model Learning

Derivative of the log-likelihood:

\[
\frac{\partial L(\theta)}{\partial W_{ij}} = \mathbb{E}_{P_{\text{data}}} [v_i h_j] - \mathbb{E}_{P_{\theta}} [v_i h_j] + \sum_{v, h} v_i h_j P_{\theta}(v, h)
\]

Easy to compute exactly

Difficult to compute: Exponentially many configurations
Use MCMC

Approximate maximum likelihood learning

\[
P_{\text{data}}(v, h; \theta) = P(h|v; \theta)P_{\text{data}}(v)
\]

\[
P_{\text{data}}(v) = \frac{1}{N} \sum_n \delta(v - v^{(n)})
\]
Approximate Learning

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

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

• 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 \( P(h | v^0) \)
  • For \( t=1:T \)
    – Sample \( v^t \) from \( P(v | h^{t-1}) \)
    – Sample \( h^t \) from \( P(h | v^t) \)
Approximate ML Learning for RBMs

• Run Markov chain (alternating Gibbs Sampling):
Approximate ML Learning for RBMs

• Run Markov chain (alternating Gibbs Sampling):
Approximate ML Learning for RBMs

• Run Markov chain (alternating Gibbs Sampling):

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

Data
Approximate ML Learning for RBMs

- Run Markov chain (alternating Gibbs Sampling):

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

\[
P(v|h) = \prod_i P(v_i|h) \quad P(v_i = 1|h) = \frac{1}{1 + \exp(-\sum_j W_{ij}h_j - b_i)}
\]
Approximate ML Learning for RBMs

- Run Markov chain (alternating Gibbs Sampling):

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

\[
P(v|h) = \prod_i P(v_i|h) \quad P(v_i = 1|h) = \frac{1}{1 + \exp(-\sum_j W_{ij}h_j - b_i)}
\]
Contrastive Divergence

• A quick way to learn RBM:

\[ P(h|v) \]

\[ v \quad ||| \quad h \quad ||| \quad \text{Data} \quad ||| \quad \text{Reconstructed Data} \]

\[ P(v|h) \]

• 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:

\[ \Delta W_{ij} = E_{P_{data}}[v_i h_j] - E_{P_1}[v_i h_j] \]

• Implementation: ~10 lines of Matlab code.

(Hinton, Neural Computation 2002)
Contrastive Divergence

• A quick way to learn RBM:

\[ P(h|v) \]

\[ \begin{array}{c}
\text{Data} \\
\text{h} \\
\text{v}
\end{array} \quad \begin{array}{c}
\text{Reconstructed Data} \\
P(v|h)
\end{array} \]

• 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.

The distributions of data and reconstructed data should be the same.

• Implementation: ~10 lines of Matlab code.

(Hinton, Neural Computation 2002)
RBMs for Real-valued Data

Gaussian-Bernoulli RBM:
- Stochastic real-valued visible variables
- Stochastic binary hidden variables
- Bipartite connections.

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

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

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

Given a set of i.i.d. training examples, we want to learn model parameters.

Maximize log-likelihood objective:

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

\[ \theta = \{W, a, b\} \]

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

4 million unlabeled images

Learned features (out of 10,000)
RBMs for Real-valued Data

4 million unlabeled images

New Image

\[ p(h_7 = 1|v) = 0.9 * \]

\[ p(h_{29} = 1|v) = 0.8 * \]

\[ + 0.6 * \] ...
RBMs for Word Counts

Replicated Softmax Model: undirected topic model:
• Stochastic 1-of-K visible variables.
• Stochastic binary hidden variables
• Bipartite connections.

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

\[ \theta = \{ W, a, b \} \]

(Salakhutdinov & Hinton, NIPS 2010, Srivastava & Salakhutdinov, NIPS 2012)
RBMs for Word Counts

Replicated Softmax Model: undirected topic model:

- Stochastic 1-of-K visible variables.
- Stochastic binary hidden variables
- Bipartite connections.

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

\[ \theta = \{W, a, b\} \]

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

(Salakhutdinov & Hinton, NIPS 2010, Srivastava & Salakhutdinov, NIPS 2012)
RBMs for Word Counts

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

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

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

Reuters dataset: 804,414 unlabeled newswire stories

Bag-of-Words

(Salakhutdinov & Hinton, NIPS 2010, Srivastava & Salakhutdinov, NIPS 2012)
RBMs for Word Counts

- One-step reconstruction from the Replicated Softmax model.

<table>
<thead>
<tr>
<th>Input</th>
<th>Reconstruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>chocolate, cake</td>
<td>cake, chocolate, sweets, dessert, cupcake, food, sugar, cream, birthday</td>
</tr>
<tr>
<td>nyc</td>
<td>nyc, newyork, brooklyn, queens, gothamist, manhattan, subway, streetart</td>
</tr>
<tr>
<td>dog</td>
<td>dog, puppy, perro, dogs, pet, filmshots, tongue, pets, nose, animal</td>
</tr>
<tr>
<td>flower, high, 花</td>
<td>flower, 花, high, japan, sakura, 日本, blossom, tokyo, lily, cherry</td>
</tr>
<tr>
<td>girl, rain, station, norway</td>
<td>norway, station, rain, girl, oslo, train, umbrella, wet, railway, weather</td>
</tr>
<tr>
<td>fun, life, children</td>
<td>children, fun, life, kids, child, playing, boys, kid, play, love</td>
</tr>
<tr>
<td>forest, blur</td>
<td>forest, blur, woods, motion, trees, movement, path, trail, green, focus</td>
</tr>
<tr>
<td>españa, agua, granada</td>
<td>españa, agua, spain, granada, water, andalucía, naturaleza, galicia, nieve</td>
</tr>
</tbody>
</table>
Collaborative Filtering

\[
P_\theta(v, h) = \frac{1}{Z(\theta)} \exp \left( \sum_{i,j,k} W^k_{ij} v_i^k h_j + \sum_{i,k} b^k_i v_i^k + \sum_j a_j h_j \right)
\]

Binary hidden: user preferences

Multinomial visible: user ratings

(\text{Salakhutdinov, Mnih, Hinton, ICML 2007})
Collaborative Filtering

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

Binary hidden: user preferences
Multinomial visible: user ratings

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

(Salakhutdinov, Mnih, Hinton, ICML 2007)
Collaborative Filtering

\[ P_\theta(v, h) = \frac{1}{Z(\theta)} \exp \left( \sum_{i,j,k} W_{ij} v_i^k h_j + \sum_{i,k} b_i^k v_i^k + \sum_j a_j h_j \right) \]

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
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

Scary Movie
Naked Gun
Hot Shots!
American Pie
Police Academy

(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:

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

The joint distribution is given by:

\[ P_\theta(v, h) = \frac{1}{\mathcal{Z}(\theta)} \exp \left( \sum_{i,j} W_{ij} v_i h_j + \sum_i b_i v_i + \sum_j a_j h_j \right) \]

Marginalizing over hidden variables:

\[ P_\theta(v) = \sum_h P_\theta(v, h) = \frac{1}{\mathcal{Z}(\theta)} \prod_i \exp(b_i v_i) \prod_j \left( 1 + \exp(a_j + \sum_i W_{ij} v_i) \right) \]
The joint distribution is given by:

\[ P_\theta(v, h) = \frac{1}{\mathcal{Z}(\theta)} \exp \left( \sum_{ij} W_{ij} v_i h_j + \sum_i b_i v_i + \sum_j a_j h_j \right) \]

Marginalizing over hidden variables:

\[ P_\theta(v) = \sum_h P_\theta(v, h) = \frac{1}{\mathcal{Z}(\theta)} \prod_i \exp(b_i v_i) \prod_j \left( 1 + \exp(a_j + \sum_i W_{ij} v_i) \right) \]

Topics “government”, “corruption” and “mafia” can combine to give very high probability to a word “Silvio Berlusconi”.

Silvio Berlusconi
The joint distribution is given by:

\[
P_\theta(v, h) = \frac{1}{Z_\theta} \exp \left( \sum W_{ij} v_i h_j + \sum h_i + \sum a_j h_j \right)
\]

Marginalizing over hidden variables:

\[
P_\theta(v) = \sum h \exp \left( \sum W_{ij} v_i h_j + \sum h_i + \sum a_j h_j \right)
\]

Topics “government”, “corruption” and “mafia” can combine to give very high probability to a word “Silvio Berlusconi”.

Silvio Berlusconi
Unsupervised Learning

• Basic Building Blocks:
  • Sparse Coding
  • Autoencoders

• Deep Generative Models
  • Restricted Boltzmann Machines
  • Deep Boltzmann Machines
  • Deep Belief Networks
  • Helmholtz Machines / Variational Autoencoders

• Generative Adversarial Networks
Deep Boltzmann Network

Input: Pixels

Low-level features:
Edges

Built from unlabeled inputs.

(Hinton et al. Neural Computation 2006)
Deep Boltzmann Network

Internal representations capture higher-order statistical structure

Higher-level features: Combination of edges

Low-level features: Edges

Built from unlabeled inputs.

Input: Pixels

(Hinton et al. Neural Computation 2006)
Unsupervised Learning

• Basic Building Blocks:
  • Sparse Coding
  • Autoencoders

• Deep Generative Models
  • Restricted Boltzmann Machines
  • Deep Boltzmann Machines
  • Deep Belief Networks
  • Helmholtz Machines / Variational Autoencoders

• Generative Adversarial Networks
Deep Belief Network
Deep Belief Network

The joint probability distribution factorizes:

\[ P(v, h^1, h^2, h^3) = P(v|h^1)P(h^1|h^2)P(h^2, h^3) \]
Deep Belief Network

The joint probability distribution factorizes:

\[ P(v, h^1, h^2, h^3) = P(v|h^1)P(h^1|h^2)P(h^2, h^3) \]

\[ P(h^2, h^3) = \frac{1}{Z(W^3)} \exp \left[ h^2 \top W^3 h^3 \right] \]
Deep Belief Network

The joint probability distribution factorizes:

\[ P(v, h^1, h^2, h^3) = P(v|h^1)P(h^1|h^2)P(h^2|h^3) \]

\[ P(h^2, h^3) = \frac{1}{Z(W^3)} \exp [h^2^T W^3 h^3] \]

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

\[ P(v_i = 1|h^1) = \frac{1}{1 + \exp \left( - \sum_j W^1_{ij} h^1_j \right)} \]
Deep Belief Network

Approximate Inference

\[ Q(h^3|h^2) \]
\[ Q(h^2|h^1) \]
\[ Q(h^1|v) \]

Generative Process

\[ P(h^2, h^3) \]
\[ P(h^1|h^2) \]
\[ P(v|h^1) \]

\[ Q(h^t|h^{t-1}) = \prod_j \sigma \left( \sum_i W^t h_{i}^{t-1} \right) \]
\[ P(h^{t-1}|h^t) = \prod_j \sigma \left( \sum_i W^t h_{i}^t \right) \]
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 $Q(h^1|v) = P(h^1|v)$ as the data for training $2^{nd}$-layer RBM.

• Learn and freeze $2^{nd}$ layer RBM

\[
Q(h^1|v)
\]
DBN Layer-wise Training

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

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

- Treat inferred values $Q(h^1 | v) = P(h^1 | v)$ as the data for training 2$^{nd}$-layer RBM.

- Learn and freeze 2$^{nd}$ layer RBM.

Layerwise pretraining improves variational lower bound.
Why this Pre-training Works?

• Greedy training improves variational lower bound!

• For any approximating distribution $Q(h^1|v)$

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

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

$$
\log P_\theta(v) = \sum_{h^1} P_\theta(v, h^1)
\geq \sum_{h^1} Q(h^1|v) \left[ \log P(h^1) + \log P(v|h^1) \right] + \mathcal{H}(Q(h^1|v))
$$
Learning Part-based Representation

Convolutional DBN

Groups of parts

Object Parts

Trained on face images

(Lee, Grosse, Ranganath, Ng, ICML 2009)
Learning Part-based Representation

Faces                       Cars                   Elephants                  Chairs

(Lee, Grosse, Ranganath, Ng, ICML 2009)
Learning Part-based Representation

Groups of parts

Class-specific object parts

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

(Lee, Grosse, Ranganath, Ng, ICML 2009)
Next lecture:
Deep Generative Models Part I