Deep Learning II
Unsupervised Learning

Russ Salakhutdinov

Machine Learning Department
Carnegie Mellon University
Canadian Institute of Advanced Research
Talk Roadmap

Part 1: Supervised Learning: Deep Networks


Part 3: Open Research Questions
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...

Explicit Density $p(x)$

Implicit Density

Generative Adversarial Networks
- Moment Matching Networks
Talk Roadmap

• Basic Building Blocks:
  ➢ Sparse Coding
  ➢ Autoencoders

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

• Generative Adversarial Networks

• Model Evaluation
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 \( \{x_1, x_2, \ldots, x_N\} \), learn a dictionary of bases \( \{\phi_1, \phi_2, \ldots, \phi_K\} \), such that:

\[
x_n = \sum_{k=1}^{K} a_{nk} \phi_k,
\]

Sparse: mostly zeros

• Each data vector is represented as a sparse linear combination of bases.
Sparse Coding

Natural Images

Learned bases: “Edges”

New example

\[ x = 0.8 \cdot \phi_{36} + 0.3 \cdot \phi_{42} + 0.5 \cdot \phi_{65} \]

\[ [0, 0, ... 0.8, ..., 0.3, ..., 0.5, ...] = \text{coefficients (feature representation)} \]

Slide Credit: Honglak Lee
Sparse Coding: Training

• Input image patches: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \in \mathbb{R}^D$

• Learn dictionary of bases: $\phi_1, \phi_2, \ldots, \phi_K \in \mathbb{R}^D$

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

- Reconstruction error
- Sparsity penalty

• Alternating Optimization:

  1. Fix dictionary of bases $\phi_1, \phi_2, \ldots, \phi_K$ and solve for activations $\mathbf{a}$ (a standard Lasso problem).

  2. Fix activations $\mathbf{a}$, optimize the dictionary of bases (convex QP problem).
Sparse Coding: Testing Time

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

\[
\min_a \left\| x^* - \sum_{k=1}^{K} a_k \phi_k \right\|_2^2 + \lambda \sum_{k=1}^{K} |a_k|
\]

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

\([0, 0, \ldots, 0.8, \ldots, 0.3, \ldots, 0.5, \ldots] = \text{coefficients (feature representation)}\)
Image Classification

Evaluated on Caltech101 object category dataset.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (Fei-Fei et al., 2004)</td>
<td>16%</td>
</tr>
<tr>
<td>PCA</td>
<td>37%</td>
</tr>
<tr>
<td>Sparse Coding</td>
<td>47%</td>
</tr>
</tbody>
</table>

Slide Credit: Honglak Lee  
(Lee, Battle, Raina, Ng, NIPS 2007)
Interpreting Sparse Coding

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

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

Feature Representation

Decoder

Encoder

Input Image

Feed-back, generative, top-down

Feed-forward, bottom-up

• Details of what goes inside the encoder and decoder matter!
• Need constraints to avoid learning an identity.
Autoencoder

Binary Features $z$

Decoder
filters $D$

Linear function

$Dz$

Encoder
filters $W$. 

Sigmoid function

$z = \sigma(Wx)$

Input Image $x$

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

\[ z = \sigma(Wx) \]

\[ \sigma(W^Tz) \]

Binary Features z

Binary Input x

Encoder filters W.

Sigmoid function

Decoder filters D

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

- Feed-forward neural network trained to reproduce its input at the output layer

\[ h(x) = g(a(x)) \]
\[ b_x = o(b_a(x)) = \text{sigm}(c + W^* h(x)) \]

\[ f(x) \triangleq b_x \quad l(f(x)) = \sum_k (b_x^k \log(b_x^k) + (1 - b_x^k) \log(1 - b_x^k)) \]

Encoder

\[ h(x) = g(a(x)) = \text{sigm}(b + Wx) \]

Decoder

\[ \hat{x} = o(\hat{a}(x)) \]
Loss Function

• **Loss function** for binary inputs

\[
l(f(x)) = - \sum_k (x_k \log(\hat{x}_k) + (1 - x_k) \log(1 - \hat{x}_k))
\]

  ➢ Cross-entropy error function (reconstruction loss)  \( f(x) \equiv \hat{x} \)

• **Loss function** for real-valued inputs

\[
l(f(x)) = \frac{1}{2} \sum_k (\hat{x}_k - x_k)^2
\]

  ➢ sum of squared differences (reconstruction loss)
  ➢ we use a linear activation function at the output
• 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.
Denoising Autoencoder

• **Idea:** representation should be robust to introduction of noise:

  - random assignment of subset of inputs to 0, with probability $\nu$
  - Similar to dropouts on the input layer
  - Gaussian additive noise

• **Reconstruction** $\hat{X}$ computed from the corrupted input $\tilde{X}$

• **Loss function** compares $\hat{X}$ reconstruction with the noiseless input $X$

(Vincent et al., ICML 2008)
\[ \hat{x} = \text{sigmoid}(c + W^* h(\tilde{x})) \]
Learned Filters

Non-corrupted

25% corrupted input

(Vincent et al., ICML 2008)
Learned Filters

Non-corrupted

50% corrupted input

(Vincent et al., ICML 2008)
Predictive Sparse Decomposition

At training time:

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

Decoder

Encoder

(Kavukcuoglu, Ranzato, Fergus, LeCun, 2009)
Stacked Autoencoders

Class Labels

Decoder

Encoder

Features

Features

Sparsity

Sparsity

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.
Deep Autoencoders

Pretraining

RBM
2000
W₁

RBM
1000
W₂

RBM
500
W₃

RBM
30
W₄

Encoder

RBM
2000
W₁^T

RBM
1000
W₂^T

RBM
500
W₃^T

RBM
30
W₄^T

Code layer

Unrolling

Decoder

RBM
2000
W₁^T + ε₁

RBM
1000
W₂^T + ε₂

RBM
500
W₃^T + ε₃

RBM
30
W₄^T + ε₄

RBM
2000
W₁^T + ε₅

RBM
1000
W₂^T + ε₆

RBM
500
W₃^T + ε₇

RBM
30
W₄^T + ε₈

Fine-tuning
Deep Autoencoders

- 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-dimentional PCA.

(Hinton and Salakhutdinov, Science 2006)
• 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)
• 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.

<table>
<thead>
<tr>
<th>Input image</th>
<th>30–RBM</th>
<th>64–RBM</th>
<th>128–RBM</th>
<th>256–RBM</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Input image" /></td>
<td><img src="image2.png" alt="30–RBM" /></td>
<td><img src="image3.png" alt="64–RBM" /></td>
<td><img src="image4.png" alt="128–RBM" /></td>
<td><img src="image5.png" alt="256–RBM" /></td>
</tr>
</tbody>
</table>

- 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 2011
- Norouzi and Fleet, ICML 2011,
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...

Explicit Density $p(x)$

Implicit Density

- Generative Adversarial Networks
- Moment Matching Networks
Talk Roadmap

• Basic Building Blocks:
  ➢ Sparse Coding
  ➢ Autoencoders

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

• Generative Adversarial Networks

• Model Evaluation
Deep Generative Model

25,000 characters from 50 alphabets around the world.

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

Model $P(\text{image})$

Bernoulli Markov Random Field
Deep Generative Model

Conditional Simulation

P(image | partial image)

Bernoulli Markov Random Field
Deep Generative Model

Why so difficult?

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

Bernoulli Markov Random Field

Conditional Simulation

P(image | partial image)
Fully Observed Models

• Explicitly model conditional probabilities:

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

• A number of successful models, including
  - NADE, RNDAE (Larochelle, et.al. 20011)
  - Pixel CNN (van den Ord et. al. 2016)
  - Pixel RNN (van den Ord et. al. 2016)
Restricted Boltzmann Machines

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

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

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

RBM is a Markov Random Field with:

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

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

Observed Data
Subset of 25,000 characters

Learned W: "edges"
Subset of 1000 features

New Image:

\[ p(h_7 = 1 | v) \]
\[ = \sigma \left( 0.99 \times \text{image} \right) \]
\[ p(h_{29} = 1 | v) \]
\[ = 0.97 \times \text{image} \]
\[ + 0.82 \times \text{image} \]

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

Logistic Function: Suitable for modeling binary images
Model Learning

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

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

Maximize log-likelihood objective:

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

Derivative of the log-likelihood:

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

\[
= \mathbb{E}_{P_{\text{data}}}[v_i h_j] - \mathbb{E}_{P_{\theta}}[v_i h_j]
\]

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

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

Difficult to compute: exponentially many configurations.
Model Learning

Derivative of the log-likelihood:

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

Easy to compute exactly

Difficult to compute: exponentially many configurations.
Use MCMC

Approximate maximum likelihood learning

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

\[
P_{data}(v) = \frac{1}{N} \sum_n \delta(v - v^{(n)})
\]
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 \(P(h \mid v^0)\)
- 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):

\[ P(h|v) \]

\[ v \quad \text{Data} \]
\[ h \]
\[ T=1 \]
\[ T=\infty \]

Equilibrium Distribution

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

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

A quick way to learn RBM:

- 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)
RBMs for Real-valued Data

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.

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

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

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

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

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

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

4 million unlabelled images

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

4 million unlabeled images

New Image

Learning features (out of 10,000)

\[ p(h_7 = 1|v) = 0.9 * \quad + \quad p(h_{29} = 1|v) = 0.8 * \quad + \quad 0.6 * \quad \ldots \]
RBMs for Word Counts

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

\[
P_\theta(v, h) = \frac{1}{Z(\theta)} \exp \left( \sum_{i=1}^{D} \sum_{k=1}^{K} \sum_{j=1}^{F} W_{ij} v_{k}^i h_{j} + \sum_{i=1}^{D} \sum_{k=1}^{K} v_{i}^k b_{i}^k + \sum_{j=1}^{F} h_{j} a_{j} \right)
\]

\[
P_\theta(v_{i}^k = 1|h) = \frac{\exp \left( b_{i}^k + \sum_{j=1}^{F} h_{j} W_{ij}^k \right)}{\sum_{q=1}^{K} \exp \left( b_{i}^q + \sum_{j=1}^{F} h_{j} W_{ij}^q \right)}
\]

Learned features: "topics"

Reuter dataset: 804,414 unlabeled newswire stories

Bag-of-Words

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

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

Binary hidden: user preferences

Multinomial visible: user ratings

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

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

(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.

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

• It is easy to infer the states of the hidden variables:
Product of Experts

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
Product of Experts

The joint distribution is given by:

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

Marginalizing over hidden variables:

\[ P_\theta(v) = \sum_h P_\theta(v, h) \]

Topics "government", "corruption" and "mafia" can combine to give very high probability to a word "Silvio Berlusconi".
Talk Roadmap

• Basic Building Blocks (non-probabilistic models):
  - Sparse Coding
  - Autoencoders

• Deep Generative Models
  - Restricted Boltzmann Machines
  - Deep Boltzmann Machines
  - Helmholtz Machines / Variational Autoencoders

• Generative Adversarial Networks
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.

(Hinton et al. Neural Computation 2006)
Low-level features: Edges

Built from \textit{unlabeled} inputs.

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

Internal representations capture higher-order statistical structure

Higher-level features: Combination of edges

Low-level features: Edges

Input: Pixels

Built from unlabeled inputs.

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

Hidden Layers

Visible Layer

RBM

Sigmoid Belief Network
Deep Belief Network

The joint probability distribution factorizes:

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

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

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

\[ P(v_i = 1|h^1) = \frac{1}{1 + \exp \left( - \sum_j W^1_{ij} h_j^1 \right)} \]
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 $h$. 

![Diagram of DBN with layers $v$ and $h$ and weight $W^1$]
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\textsuperscript{nd}-layer RBM.

• Learn and freeze 2\textsuperscript{nd} layer RBM.
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.

Unsupervised Feature Learning.
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 $Q(h^1|v)$

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

Train 2\textsuperscript{nd}-layer RBM
Learning Part-based Representation

Convolutional DBN

FACES

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)