Deep Unsupervised Learning

Russ Salakhutdinov

Machine Learning Department
Carnegie Mellon University
Canadian Institute of Advanced Research
Mining for Structure

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

Images & Video
- flickr
- Google
- YouTube

Text & Language
- Wikipedia
- Reuters
- Associated Press

Speech & Audio
- Gene Expression
- fMRI
- Tumor region

Product Recommendation
- Amazon
- eBay
- Netflix
- Facebook
- Twitter

Relational Data/Social Network

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

Deep Learning 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 Autoencoder Model

Learned latent code

- Bag of words

- Reuters dataset: 804,414 newswire stories: **unsupervised**
  - Interbank Markets
  - European Community Monetary/Economic
  - Disasters and Accidents
  - Legal/Judicial
  - Government Borrowings
  - Energy Markets
  - Leading Economic Indicators
  - Accounts/Earnings

(Hinton & Salakhutdinov, Science 2006)
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 Boltzmann Machines
  - Helmholtz Machines / Variational Autoencoders

• Generative Adversarial Networks
Talk Roadmap

• Basic Building Blocks:
  - Sparse Coding
  - Autoencoders

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

• Generative Adversarial Networks
Learning Feature Representations

Input Space

Learning Algorithm

Segway
Non-Segway
Learning Feature Representations

- **Input Space**
  - Segway
  - Non-Segway

- **Feature Representation**
  - Handle
  - Wheel

- **Feature Space**
  - Pixel 1
  - Pixel 2
  - Wheel
  - Handle
Traditional Approaches

Data → Feature extraction → Learning algorithm

Object detection
- Image
- Vision features
- Recognition

Audio classification
- Audio
- Audio features
- Speaker identification
Computer Vision Features

- **SIFT**
  - Image gradients
  - Keypoint descriptor
  - Scale-space extrema
  - Difference of Gaussian (DoG)

- **HoG**
  - Orientation Voting
  - Overlapping Blocks
  - Local Normalization
  - Input Image
  - Gradient Image

- **Textons**
  - Normalized patch

- **GIST**

- **RIFT**
  - Normalized patch
  - Orientation bins: $d=0.3, \theta=\pi$, $d=0.6, \theta=\pi/2$, $d=0.9, \theta=\pi/4$
Audio Features

Spectrogram

MFCC

Flux

ZCR

Rolloff
Audio Features

Representation Learning: Can we automatically learn these representations?

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 \( \{x_1, x_2, \ldots, x_N\} \), 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, ...] = coefficients (feature representation)
Sparse Coding: Training

• Input image patches: $x_1, x_2, \ldots, x_N \in \mathbb{R}^D$
• Learn dictionary of bases: $\phi_1, \phi_2, \ldots, \phi_K \in \mathbb{R}^D$

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

- **Reconstruction error**
- **Sparsity penalty**

• Alternating Optimization:
  1. Fix dictionary of bases $\phi_1, \phi_2, \ldots, \phi_K$ and solve for activations $a$ (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 $x^*$.

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

$\begin{align*}
x^* &= 0.8 \phi_{36} + 0.3 \phi_{42} + 0.5 \phi_{65} \\
[0, 0, ..., 0.8, ..., 0.3, ..., 0.5, ...] &= \text{coefficients (feature representation)}
\end{align*}$
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, 2006
Interpreting Sparse Coding

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

- 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

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

$z = \sigma(Wx)$

Decoder filters $D$

Linear function

$Dz$

Encoder filters $W$. Sigmoid function

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

Binary Features $z$

Input Image $x$
Autoencoder

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

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

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

Decoder

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

Encoder

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

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

\[ z = \sigma(W^T z) \]

\[ z = \sigma(Wx) \]

- Need additional constraints to avoid learning an identity.
- Relates to Restricted Boltzmann Machines (later).
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
Kavukcuoglu, Ranzato, Fergus, LeCun, 2009
Stacked Autoencoders
Stacked Autoencoders

Class Labels

Decoder

Features

Encoder

Sparsity

Decoder

Features

Sparsity

Decoder

Features

Input x

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

Unrolling

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-dimentinoal PCA.
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)
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 2011
- Norouzi and Fleet, ICML 2011,
Talk Roadmap

• Basic Building Blocks:
  ➢ Sparse Coding
  ➢ Autoencoders

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

• Generative Adversarial Networks
Deep Generative Model

Sanskrit

Model $P(\text{image})$

25,000 characters from 50 alphabets around the world.

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

Bernoulli Markov Random Field
Deep Generative Model

Conditional Simulation

P(image | partial image)

Bernoulli Markov Random Field
Deep Generative Model

Conditional Simulation

Why so difficult?

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

Bernoulli Markov Random Field

$P(\text{image} | \text{partial image})$
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, RNADE (Larochelle, et al. 2001)
- Pixel CNN (van den Ord et. al. 2016)
- Pixel RNN (van den Ord et. al. 2016)
Restricted Boltzmann Machines

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

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

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

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

New Image:
\[ p(h_7 = 1|v) = \sigma(0.99 \times \text{image}) \]
\[ p(h_{29} = 1|v) = 0.97 \times \text{image} \]
\[ p(h_{32} = 1|v) = 0.82 \times \text{image} \]

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

Logistic Function: Suitable for modeling binary images
Model Learning

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

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

Maximize log-likelihood objective:

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

Derivative of the log-likelihood:

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

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

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

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

Difficult to compute: exponentially many configurations
Model Learning

Derivative of the log-likelihood:

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

- 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):

\[ P(h|v) \]

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

\[ P(v|h) = \prod_i P(v_i|h) \quad P(v_i = 1|h) = \frac{1}{1 + \exp(-\sum_j W_{ij}h_j - b_i)} \]
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: \(\sim10\) 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.

The probability distribution is given by:

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

And the conditional distribution is:

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

New Image

\[ p(h_7 = 1|v) = 0.9 \ast \quad \quad p(h_{29} = 1|v) = 0.8 \ast \quad \quad + 0.6 \ast \quad \quad \ldots \]

Learned features (out of 10,000)
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

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

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

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

Learned features: ``topics''

Reuters 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

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

**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.

- 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:

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

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”.

- government
- authority
- power
- empire
- federation
- clinton
- house
- president
- bill
- congress
- bribery
- corruption
- dishonesty
- gang
- mob
- insider
- stock
- wall
- street
- point
- dow
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 $h$ gives:

$$P_{\theta}(v) = \sum_h P_{\theta}(v, h)$$

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

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

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.

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 Boltzmann Machines

Input: Pixels

Low-level features: Edges

Built from unlabeled inputs.

(Salakhutdinov 2008, Salakhutdinov & Hinton 2012)
Deep Boltzmann Machines

*Learn simpler representations, then compose more complex ones*

Higher-level features:
Combination of edges

Low-level features:
Edges

Built from **unlabeled** inputs.

Input: Pixels

(Salakhutdinov 2008, Salakhutdinov & Hinton 2012)
Model Formulation

\[ P_{\theta}(v, h^{(1)}, h^{(2)}, h^{(3)}) = \frac{1}{\mathcal{Z}(\theta)} \exp \left[ v^T W^{(1)} h^{(1)} + h^{(1)T} W^{(2)} h^{(2)} + h^{(2)T} W^{(3)} h^{(3)} \right] \]

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

\[ P(h_j^2 = 1|h^1, h^3) = \sigma \left( \sum_k W_{k,j}^3 h_k^3 + \sum_m W_{m,j}^2 h_m^1 \right) \]

Same as RBMs

\[ \theta = \{W^1, W^2, W^3\} \text{ model parameters} \]

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

- Hidden variables are dependent even when \textit{conditioned on the input}.
Approximate Learning

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

(Approximate) Maximum Likelihood:

\[ \frac{\partial \log P_\theta(v)}{\partial W^1} = \mathbb{E}_{P_{data}}[vh^{1\top}] - \mathbb{E}_{P_\theta}[vh^{1\top}] \]

• Both expectations are intractable!

\[ P_{data}(v, h^1) = P_\theta(h^1|v)P_{data}(v) \]

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

Not factorial any more!
Approximate Learning

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

(Approximate) Maximum Likelihood:

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

$$P_{data}(v, h^1) = P_\theta(h^1|v)P_{data}(v)$$

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

Not factorial any more!
Approximate Learning

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

(Approximate) Maximum Likelihood:

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

Variational Inference

Stochastic Approximation (MCMC-based)

Not factorial any more!
Good Generative Model?

Handwritten Characters
Good Generative Model?

Handwritten Characters
Good Generative Model?

Handwritten Characters

Simulated Real Data
Good Generative Model?

Handwritten Characters

Real Data Simulated
Good Generative Model?

Handwritten Characters
## Handwriting Recognition

### MNIST Dataset
- **60,000 examples of 10 digits**

<table>
<thead>
<tr>
<th>Learning Algorithm</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic regression</td>
<td>12.0%</td>
</tr>
<tr>
<td>K-NN</td>
<td>3.09%</td>
</tr>
<tr>
<td>Neural Net (Platt 2005)</td>
<td>1.53%</td>
</tr>
<tr>
<td>SVM (Decoste et.al. 2002)</td>
<td>1.40%</td>
</tr>
<tr>
<td>Deep Autoencoder (Bengio et. al. 2007)</td>
<td>1.40%</td>
</tr>
<tr>
<td>Deep Belief Net (Hinton et. al. 2006)</td>
<td>1.20%</td>
</tr>
<tr>
<td><strong>DBM</strong></td>
<td><strong>0.95%</strong></td>
</tr>
</tbody>
</table>

### Optical Character Recognition
- **42,152 examples of 26 English letters**

<table>
<thead>
<tr>
<th>Learning Algorithm</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic regression</td>
<td>22.14%</td>
</tr>
<tr>
<td>K-NN</td>
<td>18.92%</td>
</tr>
<tr>
<td>Neural Net</td>
<td>14.62%</td>
</tr>
<tr>
<td>SVM (Larochelle et.al. 2009)</td>
<td>9.70%</td>
</tr>
<tr>
<td>Deep Autoencoder (Bengio et. al. 2007)</td>
<td>10.05%</td>
</tr>
<tr>
<td>Deep Belief Net (Larochelle et. al. 2009)</td>
<td>9.68%</td>
</tr>
<tr>
<td><strong>DBM</strong></td>
<td><strong>8.40%</strong></td>
</tr>
</tbody>
</table>

Permutation-invariant version.
Generative Model of 3-D Objects

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

<table>
<thead>
<tr>
<th>Learning Algorithm</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic regression</td>
<td>22.5%</td>
</tr>
<tr>
<td>K-NN (LeCun 2004)</td>
<td>18.92%</td>
</tr>
<tr>
<td>SVM (Bengio &amp; LeCun 2007)</td>
<td>11.6%</td>
</tr>
<tr>
<td>Deep Belief Net (Nair &amp; Hinton 2009)</td>
<td>9.0%</td>
</tr>
<tr>
<td>DBM</td>
<td>7.2%</td>
</tr>
</tbody>
</table>

Pattern Completion

Permutation-invariant version.

Where else can we use generative models?
Data – Collection of Modalities

- Multimedia content on the web - image + text + audio.
- Product recommendation systems.
- Robotics applications.

Touch sensors
Motor control
Vision
Audio
Shared Concept

“Modality-free” representation

“Concept”

“Modality-full” representation

- sunset, pacific ocean
- baker beach, seashore, ocean
Challenges - I

Very different input representations

- Images – real-valued, dense
- Text – discrete, sparse

Difficult to learn cross-modal features from low-level representations.
<table>
<thead>
<tr>
<th>Image</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.jpg" alt="Image" /></td>
<td>pentax, k10d, pentaxda50200, kangarooisland, sa, australiansealion</td>
</tr>
<tr>
<td><img src="image2.jpg" alt="Image" /></td>
<td>mickikrimmel, mickipedia, headshot</td>
</tr>
<tr>
<td><img src="image3.jpg" alt="Image" /></td>
<td>&lt; no text &gt;</td>
</tr>
<tr>
<td><img src="image4.jpg" alt="Image" /></td>
<td>unveulpixel, naturey</td>
</tr>
</tbody>
</table>

**Challenges - II**

Noisy and missing data
## Challenges - II

<table>
<thead>
<tr>
<th>Image</th>
<th>Text</th>
<th>Text generated by the model</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Beach Image" /></td>
<td>pentax, k10d, pentaxda50200, kangarooisland, sa, australiansealion</td>
<td>beach, sea, surf, strand, shore, wave, seascape, sand, ocean, waves</td>
</tr>
<tr>
<td><img src="image" alt="Portrait Image" /></td>
<td>mickikrimmel, mickipedia, headshot</td>
<td>portrait, girl, woman, lady, blonde, pretty, gorgeous, expression, model</td>
</tr>
<tr>
<td><img src="image" alt="Night Image" /></td>
<td>&lt; no text &gt;</td>
<td>night, notte, traffic, light, lights, parking, darkness, lowlight, nacht, glow</td>
</tr>
<tr>
<td><img src="image" alt="Nature Image" /></td>
<td>unseulpixel, naturey</td>
<td>fall, autumn, trees, leaves, foliage, forest, woods, branches, path</td>
</tr>
</tbody>
</table>
A Simple Multimodal Model

- Use a joint binary hidden layer.
- **Problem**: Inputs have very different statistical properties.
- Difficult to learn cross-modal features.
Multimodal DBM

Gaussian model

Dense, real-valued image features

\( h \)

\( \mathbf{V}_{\text{image}} \)

Replicated Softmax

\( \mathbf{V}_{\text{text}} \)

Word counts

(Srivastava & Salakhutdinov, NIPS 2012, JMLR 2014)
Multimodal DBM

Dense, real-valued image features

Gaussian model

Replicated Softmax

Word counts

(Srivastava & Salakhutdinov, NIPS 2012, JMLR 2014)
Multimodal DBM

Gaussian model

Dense, real-valued image features

V_{image}

Replicated Softmax

Word counts

V_{text}

(Srivastava & Salakhutdinov, NIPS 2012, JMLR 2014)
Multimodal DBM

Dense, real-valued image features

Bottom-up + Top-down

Gaussian model

Replicated Softmax

Word counts

(Srivastava & Salakhutdinov, NIPS 2012, JMLR 2014)
Text Generated from Images

Given

Generated

dog, cat, pet, kitten, puppy, ginger, tongue, kitty, dogs, furry

sea, france, boat, mer, beach, river, bretagne, plage, brittany

portrait, child, kid, ritratto, kids, children, boy, cute, boys, italy

insect, butterfly, insects, bug, butterflies, lepidoptera

graffiti, streetart, stencil, sticker, urbanart, graff, sanfrancisco

canada, nature, sunrise, ontario, fog, mist, bc, morning
Given

- portrait
- women
- army
- soldier
- mother
- postcard
- soldiers

Generated

- obama
- barackobama
- election
- politics
- president
- hope
- change
- sanfrancisco
- convention
- rally

- water
- glass
- beer
- bottle
- drink
- wine
- bubbles
- splash
- drops
- drop
Images from Text

Given
- water, red, sunset
- nature, flower, red, green
- blue, green, yellow, colors
- chocolate, cake

Retrieved
MIR-Flickr Dataset

• 1 million images along with user-assigned tags.

- sculpture, beauty, stone
- d80
- nikon, abigfave, goldstaraward, d80, nikond80
- food, cupcake, vegan
- anawesomeshot, theperfectphotographer, flash, damniwishidtakenthat, spiritofphotography
- nikon, green, light, photoshop, apple, d70
- white, yellow, abstract, lines, bus, graphic
- sky, geotagged, reflection, cielo, bilbao, reflejo

Huiskes et. al.
Results

• Logistic regression on top-level representation.

• Multimodal Inputs

<table>
<thead>
<tr>
<th>Learning Algorithm</th>
<th>MAP</th>
<th>Precision@50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>0.124</td>
<td>0.124</td>
</tr>
<tr>
<td>LDA [Huiskes et. al.]</td>
<td>0.492</td>
<td>0.754</td>
</tr>
<tr>
<td>SVM [Huiskes et. al.]</td>
<td>0.475</td>
<td>0.758</td>
</tr>
<tr>
<td>DBM-Labelled</td>
<td>0.526</td>
<td>0.791</td>
</tr>
</tbody>
</table>

Mean Average Precision

Labeled 25K examples
Results

- Logistic regression on top-level representation.
- Multimodal Inputs

<table>
<thead>
<tr>
<th>Learning Algorithm</th>
<th>MAP</th>
<th>Precision@50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>0.124</td>
<td>0.124</td>
</tr>
<tr>
<td>LDA [Huiskes et. al.]</td>
<td>0.492</td>
<td>0.754</td>
</tr>
<tr>
<td>SVM [Huiskes et. al.]</td>
<td>0.475</td>
<td>0.758</td>
</tr>
<tr>
<td>DBM-Labelled</td>
<td>0.526</td>
<td>0.791</td>
</tr>
<tr>
<td>Deep Belief Net</td>
<td>0.638</td>
<td>0.867</td>
</tr>
<tr>
<td>Autoencoder</td>
<td>0.638</td>
<td>0.875</td>
</tr>
<tr>
<td>DBM</td>
<td>0.641</td>
<td>0.873</td>
</tr>
</tbody>
</table>

Mean Average Precision

Labeled 25K examples + 1 Million unlabelled
Multimodal Linguistic Regularities

- day + night =
- flying + sailing =
- bowl + box =
- box + bowl =

(Kiros, Salakhutdinov, Zemel, TACL 2015)
Talk Roadmap

• Basic Building Blocks:
  - Sparse Coding
  - Autoencoders

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

• Generative Adversarial Networks
Helmholtz Machines


Approximate Inference

- Kingma & Welling, 2014
- Rezende, Mohamed, Daan, 2014
- Mnih & Gregor, 2014
- Bornschein & Bengio, 2015
- Tang & Salakhutdinov, 2013
Helmholtz Machines vs. DBMs

Helmholtz Machine

Approximate Inference

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

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

\[ Q(h^1|x) \]

Input data

\[ h^3 \]

\[ h^2 \]

\[ h^1 \]

\[ x \]

Generative Process

\[ P(h^3) \]

\[ P(h^2|h^3) \]

\[ P(h^1|h^2) \]

\[ P(x|h^1) \]

Deep Boltzmann Machine

\[ h^3 \]

\[ h^2 \]

\[ h^1 \]

\[ v \]

\[ W^3 \]

\[ W^2 \]

\[ W^1 \]
Variational Autoencoders (VAEs)

• The VAE defines a generative process in terms of ancestral sampling through a cascade of hidden stochastic layers:

\[ p(x|\theta) = \sum_{h^1, \ldots, h^L} p(h^L|\theta)p(h^{L-1}|h^L, \theta) \cdots p(x|h^1, \theta) \]

Each term may denote a complicated nonlinear relationship

- \( \theta \) denotes parameters of VAE.
- \( L \) is the number of stochastic layers.
- Sampling and probability evaluation is tractable for each \( p(h^l|h^{l+1}) \).
VAE: Example

• The VAE defines a generative process in terms of ancestral sampling through a cascade of hidden stochastic layers:

\[ p(x|\theta) = \sum_{h^1, h^2} p(h^2|\theta)p(h^1|h^2, \theta)p(x|h^1, \theta) \]

This term denotes a one-layer neural net.

• \( \theta \) denotes parameters of VAE.

• \( L \) is the number of stochastic layers.

• Sampling and probability evaluation is tractable for each \( p(h^l|h^{l+1}) \).
Variational Bound

• The VAE is trained to maximize the variational lower bound:

\[
\log p(x) = \log \mathbb{E}_{q(h|x)} \left[ \frac{p(x, h)}{q(h|x)} \right] \geq \mathbb{E}_{q(h|x)} \left[ \log \frac{p(x, h)}{q(h|x)} \right] = \mathcal{L}(x)
\]

\[
\mathcal{L}(x) = \log p(x) - D_{KL} (q(h|x)) \| p(h|x))
\]

• Trading off the data log-likelihood and the KL divergence from the true posterior.

• Hard to optimize the variational bound with respect to the recognition network (high-variance).

• Key idea of Kingma and Welling is to use reparameterization trick.
Reparameterization Trick

• Assume that the recognition distribution is Gaussian:

\[ q(\mathbf{h}^\ell | \mathbf{h}^{\ell-1}, \theta) = \mathcal{N}(\mu(\mathbf{h}^{\ell-1}, \theta), \Sigma(\mathbf{h}^{\ell-1}, \theta)) \]

with mean and covariance computed from the state of the hidden units at the previous layer.

• Alternatively, we can express this in term of auxiliary variable:

\[ \epsilon^\ell \sim \mathcal{N}(0, \mathbf{I}) \]
\[ \mathbf{h}^\ell (\epsilon^\ell, \mathbf{h}^{\ell-1}, \theta) = \Sigma(\mathbf{h}^{\ell-1}, \theta)^{1/2} \epsilon^\ell + \mu(\mathbf{h}^{\ell-1}, \theta) \]
Reparameterization Trick

• Assume that the recognition distribution is Gaussian:

\[ q(h^\ell | h^{\ell-1}, \theta) = \mathcal{N}(\mu(h^{\ell-1}, \theta), \Sigma(h^{\ell-1}, \theta)) \]

• Or

\[ \epsilon^\ell \sim \mathcal{N}(0, I) \]

\[ h^\ell (\epsilon^\ell, h^{\ell-1}, \theta) = \Sigma(h^{\ell-1}, \theta)^{1/2} \epsilon^\ell + \mu(h^{\ell-1}, \theta) \]

• The recognition distribution \( q(h^\ell | h^{\ell-1}, \theta) \) can be expressed in terms of a deterministic mapping:

\[ h(\epsilon, x, \theta), \quad \text{with} \quad \epsilon = (\epsilon^1, \ldots, \epsilon^L) \]

- Deterministic Encoder
- Distribution of \( \epsilon \) does not depend on \( \theta \)
Computing the Gradients

• The gradient w.r.t the parameters: both recognition and generative:

\[ \nabla_{\theta} \mathbb{E}_{h \sim q(h|x, \theta)} \left[ \log \frac{p(x, h|\theta)}{q(h|x, \theta)} \right] \]

\[ = \nabla_{\theta} \mathbb{E}_{\epsilon^1, \ldots, \epsilon^L \sim \mathcal{N}(0, I)} \left[ \log \frac{p(x, h(\epsilon, x, \theta)|\theta)}{q(h(\epsilon, x, \theta)|x, \theta)} \right] \]

\[ = \mathbb{E}_{\epsilon^1, \ldots, \epsilon^L \sim \mathcal{N}(0, I)} \left[ \nabla_{\theta} \log \frac{p(x, h(\epsilon, x, \theta)|\theta)}{q(h(\epsilon, x, \theta)|x, \theta)} \right] \]

Gradients can be computed by backprop

The mapping \( h \) is a deterministic neural net for fixed \( \epsilon \).
Importance Weighted Autoencoders

• Can improve VAE by using following k-sample importance weighting of the log-likelihood:

\[
\mathcal{L}_k(x) = \mathbb{E}_{h_1, \ldots, h_k \sim q(h|x)} \left[ \log \frac{1}{k} \sum_{i=1}^{k} \frac{p(x, h_i)}{q(h_i | x)} \right]
\]

\[
= \mathbb{E}_{h_1, \ldots, h_k \sim q(h|x)} \left[ \log \frac{1}{k} \sum_{i=1}^{k} w_i \right]
\]

where \( h_1, \ldots, h_k \) are sampled from the recognition network.

Burda, Grosse, Salakhutdinov, 2015
Generators of Images from Captions

• **Generative Model:** Stochastic Recurrent Network, chained sequence of Variational Autoencoders, with a single stochastic layer.

• **Recognition Model:** Deterministic Recurrent Network.

Gregor et. al. 2015

(Mansimov, Parisotto, Ba, Salakhutdinov, 2015)
Motivating Example

• Can we generate images from natural language descriptions?

A **stop sign** is flying in blue skies

A **pale yellow school bus** is flying in blue skies

A **herd of elephants** is flying in blue skies

A **large commercial airplane** is flying in blue skies

(Mansimov, Parisotto, Ba, Salakhutdinov, 2015)
Flipping Colors

A yellow school bus parked in the parking lot

A red school bus parked in the parking lot

A green school bus parked in the parking lot

A blue school bus parked in the parking lot

(Mansimov, Parisotto, Ba, Salakhutdinov, 2015)
Qualitative Comparison

A group of people walk on a beach with surfboards

Our Model

LAPGAN (Denton et. al. 2015)

Conv-Deconv VAE

Fully Connected VAE
Novel Scene Compositions

A toilet seat sits open in the bathroom

A toilet seat sits open in the grass field

Ask Google?
Neural Story Telling

Sample from the Generative Model (recurrent neural network):

We were barely able to catch the breeze at the beach, and it felt as if someone stepped out of my mind.

She was in love with him for the first time in months, so she had no intention of escaping. The sun had risen from the ocean, making her feel more alive than normal. She is beautiful, but the truth is that I do not know what to do. The sun was just starting to fade away, leaving people scattered around the Atlantic Ocean.

Kiros et.al., NIPS 2015
One-Shot Learning: Humans vs. Machines

(Lake, Salakhutdinov, Tenenbaum, Science, 2015)
Talk Roadmap

• Basic Building Blocks:
  - Sparse Coding
  - Autoencoders

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

• Generative Adversarial Networks
Generative Adversarial Networks

• There is no explicit definition of the density for \( p(x) \) – Only need to be able to sample from it.

• No variational learning, no maximum-likelihood estimation, no MCMC. How?

• By playing a game!
Generative Adversarial Networks

• Set up a game between two players:
  - Discriminator D
  - Generator G

• **Discriminator D** tries to discriminate between:
  - A sample from the data distribution.
  - And a sample from the generator G.

• The **Generator G** attempts to “fool” D by generating samples that are hard for D to distinguish from the real data.
Generative Adversarial Networks

D tries to output 1

Differentiable function D

x sampled from data

Slide Credit: Ian Goodfellow
Generative Adversarial Networks

D tries to output 1

Differentiable function D

x sampled from data

x sampled from model

Differentiable function G

Input noise Z

Slide Credit: Ian Goodfellow
Generative Adversarial Networks

D tries to output 1
Differentiable function D
x sampled from data

D tries to output 0
Differentiable function D
x sampled from model

Input noise
Z

Slide Credit: Ian Goodfellow
Generative Adversarial Networks

- Minimax value function
- Generator: generate samples that D would classify as real
- \( \min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \)

Discriminator:
- Pushes up
- Classify data as being real
- Classify generator samples as being fake

- Optimal strategy for Discriminator is:
\[
D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}
\]
DCGAN Architecture

(Radford et al 2015)
LSUN Bedrooms: Samples

(Radford et al 2015)
CIFAR

Training Samples

Samples

(Salimans et. al., 2016)
IMAGENET

Training

Samples

(Salimans et. al., 2016)
ImageNet: Cherry-Picked Results

• **Open Question:** How can we quantitatively evaluate these models!

Slide Credit: Ian Goodfellow
Summary

- Efficient learning algorithms for Deep Unsupervised Models

Deep models improve the current state-of-the-art in many application domains:

- Object recognition and detection, text and image retrieval, handwritten character and speech recognition, and others.
Thank you