TUTORIAL PART 1
Unsupervised Learning

Marc'Aurelio Ranzato
Department of Computer Science – Univ. of Toronto
ranzato@cs.toronto.edu

Co-organizers: Honglak Lee, Yoshua Bengio, Geoff Hinton, Yann LeCun, Andrew Ng

Deep Learning and Unsupervised Feature Learning Workshop, 10 Dec. 2010
Feature Learning

Input

Input space

color

brightness

Motorbikes

“Non”-Motorbikes

Learning algorithm

kindly borrowed from Andrew Ng ECCV10
Feature Learning

Input space

- Motorbikes
- "Non"-Motorbikes

Input

- color
- brightness

Learning algorithm

"wheel"

"handle"

kindly borrowed from Andrew Ng ECCV10
How is computer perception done?

Object detection

Image

Low-level vision features

Recognition

Audio classification

Audio

Low-level audio features

Speaker identification

Helicopter control

Helicopter

Low-level state features

Action
Computer vision features

SIFT

Spin image

HoG

RIFT

Textons

GLOH

kindly borrowed from Andrew Ng ECCV10
Audio features

- Spectrogram
- MFCC
- Flux
- ZCR
- Rolloff

kindly borrowed from Andrew Ng ECCV10
Engineering features:

- Need expert knowledge
- Sub-optimal
- Time-consuming and expensive
- Does not generalize to other domain
The goal of Unsupervised Feature Learning

Unlabeled images → Learning algorithm → Feature representation

kindly borrowed from Andrew Ng ECCV10
Outline

- What is Unsupervised Learning?
- Unsupervised Learning Algorithms
- Comparing Unsupervised Learning Algorithms
- (Tutorial II) Deep Learning
Unsupervised Learning

Data points belonging to 3 classes
Unsupervised Learning

No labels are provided during training
Unsupervised Learning

Fit mixture of 3 Gaussians: use responsibility to represent a data point (indicative of its class)
Unsupervised Learning

- Unsupervised Learning
  - Density estimation
  - Latent variables (→ features, possibly useful for discrimination)
Unsupervised Learning

- Unsupervised Learning
- Density estimation
- Latent variables (→ features, possibly useful for discrimination)

- Energy-based interpretation
  - Each data-point $x$ has associated energy $E(x)$
  - Training has to make $E(x)$ lower for $x$ in training set
Unsupervised Learning

- Unsupervised Learning
- Density estimation
- Latent variables (→ features, possibly useful for discrimination)

- Energy-based interpretation
  - Each data-point $x$ has associated energy $E(x)$
  - Training has to make $E(x)$ lower for $x$ in training set

BEFORE TRAINING
Unsupervised Learning

- Unsupervised Learning
- Density estimation
- Latent variables (→ features, possibly useful for discrimination)

Energy-based interpretation
- Each data-point $x$ has associated energy $E(x)$
- Training has to make $E(x)$ lower for $x$ in training set

AFTER TRAINING
Principal Component Analysis

\[ E(X, Z; W) = \|X - WZ\|^2 \]

Feature: \( Z = W'X \), it must be lower dimensional

Training: minimize \( E \) s.t. orthogonality constraint
Principal Component Analysis

\[ E(X, Z; W) = \| X - WZ \|^2 \]

Feature: \( Z = W'X \), it must be lower dimensional

Training: minimize \( E \) s.t. orthogonality constraint
### Principal Component Analysis

$$E(X, Z; W) = \|X - W Z\|^2$$

Feature: $$Z = W'X$$

<table>
<thead>
<tr>
<th><strong>PROS</strong></th>
<th><strong>CONS</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple training (tuning free)</td>
<td>Feature must be lower dimensional</td>
</tr>
<tr>
<td>Unique solution</td>
<td>Features are linear</td>
</tr>
<tr>
<td>Fast</td>
<td></td>
</tr>
</tbody>
</table>
Auto-encoder Neural Network

\[ E(X, Z; W) = \| X - f(WZ) \|^2 \]

Feature: \( Z = g(AX) \), lower dimensional

Training: minimize \( E \)
Auto-encoder Neural Network

\[ E(X, Z; W) = \| X - f(WZ) \|^2 \]

Feature: \( Z = g(A \times X) \), lower dimensional

<table>
<thead>
<tr>
<th>PROS</th>
<th>CONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ Non-linear features</td>
<td>▪ Feature must be lower dimensional</td>
</tr>
<tr>
<td>▪ Pretty fast training</td>
<td>▪ A few hyper-parameters</td>
</tr>
<tr>
<td></td>
<td>▪ Optimization becomes hard</td>
</tr>
<tr>
<td></td>
<td>▪ if highly non-linear</td>
</tr>
</tbody>
</table>
Denoising Auto-encoder

\[ E(X, Z; W) = \|X - f(WZ)\|^2 \]

Feature: \( Z = g(A(X + n)) \), \( n \) is noise

Training: minimize \( E \)
Denoising Auto-encoder

\[ E(X, Z; W) = \|X - f(WZ)\|^2 \]

Feature: \[ Z = g(A(X + n)) \]

**PROS**
- Non-linear features
- Pretty fast training
- Robustness to noise in the input
- Possibly higher dimensional features
- Can check convergence

**CONS**
- A few hyper-parameters
- Optimization becomes hard if highly non-linear
- Choice of noise distribution
K-Means

\[ E(X, Z; W) = \| X - WZ \|^2 \]

Feature: \( Z \in \text{1-of-N code} \)

Training: minimize \( E \)
K-Means

\[ E(X, Z; W) = \| X - W Z \|^2 \]

Feature: \( Z \in 1\text{-of-}N \) code

**PROS**
- Simple training (tuning free)
- Fast

**CONS**
- One might need lots of prototypes to cover high-dimensional space
- Representation is too sparse
Sparse Coding

$$E(X, Z; W) = \|X - WZ\|^2 + \lambda |Z|_1$$

Feature: $Z$ sparse

Training: minimize $E$ (coordinate descent)
Sparse Coding

\[ E(X, Z; W) = \|X - WZ\|^2 + \lambda |Z|_1 \]

Feature: \( Z \) sparse

**PROS**
- Possibly higher-dimensional features
- It often yields more interpretable features
- Biologically plausible (?)

**CONS**
- Expensive training
- Expensive inference
- Need to tune \( \lambda \)
Predictive Sparse Coding

\[ E(X, Z; W) = \|X - WZ\|^2 + \lambda |Z|_1 + \|Z - g(A'X)\|^2 \]

Feature: \( Z \) sparse

**PROS**
- Possibly higher-dimensional features
- Fast inference

**CONS**
- Expensive training
- Need to tune \( \lambda \)
Restricted Boltzmann Machine

\[ E(X, Z; W) = -Z' W' X \]

All variables are binary: \( X_i \in \{0, 1\}, Z_j \in \{0, 1\} \)

\[ E = -w_{11} X_1 Z_1 - w_{12} X_1 Z_2 - w_{21} X_2 Z_1 - \ldots \]
Restricted Boltzmann Machine

\[ E(X, Z; W) = -Z'W'X \]

\[ p(X, Z; W) = \frac{\exp(-E(X, Z; W))}{\sum_x \sum_z \exp(-E(x', z'; W))} \]
Restricted Boltzmann Machine

\[ E(X, Z; W) = -Z'W'X \]

\[ p(X, Z; W) = \frac{\exp(-E(X, Z; W))}{\sum_x \sum_z \exp(-E(x', z'; W))} \]

\[ \text{INTRACTABLE} \]
Restricted Boltzmann Machine

\[ E(X, Z; W) = -Z'W'X \]

\[ p(X = 1|Z; W) = \prod_j \sigma(W_jZ), \quad \sigma(u) = \frac{1}{1 + \exp(-u)} \]

\[ p(Z = 1|X; W) = \prod_i \sigma(W_i'X) \]
Restricted Boltzmann Machine

\[ E(X, Z; W) = -ZW'X \]

\[ p(X|Z; W) = \sigma(WZ) \]

\[ p(Z|X; W) = \sigma(W'X) \]

Easy conditionals: efficient Gibbs sampling
Restricted Boltzmann Machine

\[ E(X, Z; W) = -Z'W'X \]
\[ p(X|Z; W) = \sigma(WZ) \]
\[ p(Z|X; W) = \sigma(W'X) \]

Easy conditionals: efficient Gibbs sampling
Restricted Boltzmann Machine

\[ E(X, Z; W) = -Z'W'X \]

\[ p(X|Z; W) = \sigma(WZ) \]

\[ p(Z|X; W) = \sigma(W'X) \]

Easy conditionals: efficient Gibbs sampling
Restricted Boltzmann Machine

\[ E(X, Z; W) = -Z'W'X \]

\[ p(X|Z; W) = \sigma(WZ) \]

\[ p(Z|X; W) = \sigma(W'X) \]

Easy conditionals: efficient Gibbs sampling
\[ E(X, Z; W) = -Z'W'X \]
\[ p(X|Z; W) = \sigma(WZ) \]
\[ p(Z|X; W) = \sigma(W'X) \]

Subsequently used as features

Restricted Boltzmann Machine

\[ Z_1 \quad Z_2 \]
\[ W \]
\[ X_1 \quad X_2 \quad X_3 \]
Restricted Boltzmann Machine

\[ E(X, Z; W) = -Z'W'X \]

\[ \text{Loss} = -\log(p(X; W)) \]

\[ W \leftarrow W - \eta \left( \sum_z p(z|X^t) \frac{\partial E(X^t, z; W)}{\partial W} - \sum_{x, z} p(x, z) \frac{\partial E(x, z; W)}{\partial W} \right) \]
Restricted Boltzmann Machine

\[ E(X, Z; W) = -Z'W'X \]

Loss = \(-\log(p(X; W))\)

\[ W \leftarrow W - \eta \left( \sum_z p(z|X^t) \frac{\partial E(X^t, z; W)}{\partial W} - \sum_{x, z} p(x, z) \frac{\partial E(x, z; W)}{\partial W} \right) \]
Restricted Boltzmann Machine

\[ E(X, Z; W) = -Z'W'X \]

Loss = \(-\log(p(X; W))\)

\[ W \leftarrow W - \eta (\sum_z p(z|X^t) \frac{\partial E(X^t, z; W)}{\partial W} - \sum_{x,z} p(x, z) \frac{\partial E(x, z; W)}{\partial W}) \]
Restricted Boltzmann Machine

\[ E(X, Z; W) = -Z'W'X \]

\[ \text{Loss} = -\log(p(X; W)) \]

\[ W \leftarrow W - \eta \left( \sum_z p(z|X^t) \frac{\partial E(X^t, z; W)}{\partial W} - \sum_z p(z|X^m) \frac{\partial E(X^m, z; W)}{\partial W} \right) \]

USING MCMC
Restricted Boltzmann Machine

\[ E(X, Z; W) = -Z'W'X \]

\[ \text{Loss} = -\log(p(X; W)) \]

\[ W \leftarrow W - \eta \left( \sum_z p(z|X^t) \frac{\partial E(X^t, z; W)}{\partial W} - \sum_z p(z|X^m) \frac{\partial E(X^m, z; W)}{\partial W} \right) \]

In practice, Gibbs sampler takes too long to converge:

- Contrastive Divergence
- Persistent Contrastive Divergence
- Fast Persistent Contrastive Divergence
- Score Matching
- Ratio Matching
- Margin-Based Losses
- Variational Methods
**Restricted Boltzmann Machine**

\[ E(X, Z; W) = -Z'W'X \]

Feature: \( Z = g(W'X) \)

<table>
<thead>
<tr>
<th><strong>PROS</strong></th>
<th><strong>CONS</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Possibly higher-dimensional features</td>
<td>Exact learning is intractable</td>
</tr>
<tr>
<td>It can generate data</td>
<td>Approximations do not let easily assess convergence</td>
</tr>
<tr>
<td>Simple interpretation of learning rule</td>
<td>A few hyper-parameters to tune</td>
</tr>
<tr>
<td>Simple to extend variables to other distributions</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Linear Features</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>PCA</td>
<td>yes</td>
</tr>
<tr>
<td>Auto-encoder</td>
<td>no</td>
</tr>
<tr>
<td>K-Means</td>
<td>no</td>
</tr>
<tr>
<td>Sparse Coding</td>
<td>no</td>
</tr>
<tr>
<td>RBM</td>
<td>no</td>
</tr>
<tr>
<td>Denoising Auto-enc.</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>Sparse features</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td><strong>PCA</strong></td>
<td>no (but it can be added)</td>
</tr>
<tr>
<td><strong>Auto-encoder</strong></td>
<td>no (but it can be added)</td>
</tr>
<tr>
<td><strong>K-Means</strong></td>
<td>yes</td>
</tr>
<tr>
<td><strong>Sparse Coding</strong></td>
<td>yes</td>
</tr>
<tr>
<td><strong>RBM</strong></td>
<td>no (but it can be added)</td>
</tr>
<tr>
<td><strong>Denoising Auto-enc.</strong></td>
<td>no (but it can be added)</td>
</tr>
<tr>
<td></td>
<td>Energy “pull-up”</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>PCA</td>
<td>restriction on code</td>
</tr>
<tr>
<td>Auto-encoder</td>
<td>restriction on code</td>
</tr>
<tr>
<td>K-Means</td>
<td>restriction on code</td>
</tr>
<tr>
<td>Sparse Coding</td>
<td>restriction on code</td>
</tr>
<tr>
<td>RBM</td>
<td>partition function</td>
</tr>
<tr>
<td>Denoising Auto-enc.</td>
<td>noise to input</td>
</tr>
</tbody>
</table>
Comparison

PCA on 8x8 patches

ICA on 8x8 patches
Comparing Unsupervised Algorithms

- Properties of features
- Reconstruction error
- Likelihood on test data
- Discriminative performance of classifier trained on features
- Statistical dependency of components
- Denoising performance
- Other tasks
References: RBM


code
- http://www.cs.toronto.edu/~hinton/code/rbm.m (matlab)
- http://www.cs.toronto.edu/~tijmen/gnumpy_example.py (python using gnumpy module to run on a GPU)
References: sparse RBM

- Lee, Ekanadham, Ng “Sparse deep belief net model for visual area V2” NIPS 2008
- Lee, Grosse, Ranganath, Ng “Convolutional Deep Belief Networks for scalable unsupervised learning of hierarchical representations”, ICML 2009
References: S-RBM

- Osindero, Hinton “Modeling image patches with a directed hierarchy of markov random fields” NIPS 2008
References: mcRBM

- Ranzato, Hinton “Modeling pixel means and covariances using factorized third-order Boltzmann machines” CVPR 2010
- Ranzato, Mnih, Hinton “Generating more realistic images using gated MRF” NIPS 2010

code
References: Sparse Coding

- Olshausen, Field “Sparse coding with an overcomplete basis set: a strategy employed by V1?” Vision Research 1997
- Lee, Battle, Raina, Ng “Efficient sparse coding algorithms” NIPS 2007
- Lee, Raina, Teichman, Ng “Exponential family sparse coding with applications to self-taught learning” IJCAI 2009

code
- http://www.eecs.umich.edu/~honglak/softwares/nips06-sparsecoding.htm
- http://www.di.ens.fr/willow/SPAMS/
  (Julien Mairal's work on fast sparse coding methods with several extensions)
- http://www.cs.technion.ac.il/~ronrubin/software.html
  (see also work by M. Elad et al. on K-SVD, another sparse coding algorithm)
References: Predictive Sparse Coding

- Kavukcuoglu, Sermanet, Boureau, Gregor, Mathieu, LeCun, "Learning Convolutional Feature Hierarchies for Visual Recognition" NIPS 2010

**code**
- http://cs.nyu.edu/~koray/wp/?page_id=29
  (basic algorithm for predictive sparse decomposition)
References: Local Coordinate Coding

- Yu, Zhang, Gong “Nonlinear learning using local coordinate coding” NIPS 2009
References: Product of Student's t

References: Denoising Auto-encoders

- Pascal, Larochelle, Bengio, Manzagol “Extracting and composing robust features with denoising autoencoders” ICML 2008

**code**
- [http://www.deeplearning.net/tutorial/dA.html#daa](http://www.deeplearning.net/tutorial/dA.html#daa)
  code written in Theano, a python library with interface to GPU, developed in Y. Bengio's lab, see more at: [http://www.deeplearning.net/tutorial/](http://www.deeplearning.net/tutorial/)
End of Part 1

Any questions?