Efficient Learning of Sparse Overcomplete Representations with an Energy-Based Model

Marc'Aurelio Ranzato

C. Poultney
S. Chopra (NYU – Courant Institute)
Y. LeCun

CIAR Summer School – Toronto 2006
Why Extracting Features?

Preprocessing in image analysis systems

- Extract information from high dimensional data (e.g. compression, visualization)
- Map data into higher dimensional where features become linearly separable (e.g. kernel methods for classification)
- Find representation with a more “meaningful” interpretation (e.g. NMF applied to face images finds “parts”)

Why Extracting Features?
Why Extracting Features?

Biological motivation

- The brain circuitry extracts information from highly redundant sensory signals

- **Barlow**: the goal of sensory coding is to transform the input reducing the redundancy among elements in the input stream

  [“Possible principle underlying the transformation of sensory messages” Sensory Communication 1961]

- **Hubel, Wiesel**: receptive fields in area V1 of visual cortex code edges at different scales, orientations and spatial locations

  [“Receptive Fields of single neurones in cat's striate cortex” J. Physiol. 1959]
Why sparse & overcomplete codes?

Surface area (mm^2) ...........................................190,000
Thickness (mm) ..................................................2.5
Neurons/mm^3 ...................................................40,000
Synapses/mm^3 ................................................7x10^8
Energy available for spikes/min .............................4x10^20 ATP
Energy for 1 spike ...........................................2.2x10^9 ATP
Average rate (spike/s/neuron) ..............................0.16

“... in strongly driven visual cortex, only a small fraction of neurons is working at any one time – between 1 in 25 and 1 in 63, with the latter the more probable value”

This might be interpreted as evidence for the use of a sparse and over-complete code.

[P. Lennie “The cost of cortical computation” Current Biology 2003]
Sparse and overcomplete representations are also efficient:

- compression
- robustness to noise
- better tiling of joint space of location and frequency
Unsupervised Feature Extractor

- PCA
- Kernel-PCA  \([\text{Scholkopf, Smola, Muller} - 1998]\)
- Auto-encoders
- ICA  \([\text{Bell, Sejnowski} - 1995]\)
- NMF  \([\text{Lee, Seung} - 1999]\)
Unsupervised Feature Extractor

- PCA
- Kernel-PCA  [Scholkopf, Smola, Muller - 1998]
- Auto-encoders
- ICA  [Bell, Sejnowski - 1995]
- NMF  [Lee, Seung - 1999]
Unsupervised Feature Extractor

- PCA
- Kernel-PCA \([Scholkopf, Smola, Muller - 1998]\) \(\rightarrow\) COMPACT + NON LINEAR
- Auto-encoders
- ICA \([Bell, Sejnowski - 1995]\)
- NMF \([Lee, Seung - 1999]\)
Unsupervised Feature Extractor

- PCA
- Kernel-PCA \([\text{Scholkopf, Smola, Muller - 1998}]\)
- Auto-encoders
- ICA \([\text{Bell, Sejnowski - 1995}]\)
- NMF \([\text{Lee, Seung - 1999}]\)
Unsupervised Feature Extractor

- PCA
- Kernel-PCA [Scholkopf, Smola, Muller - 1998]
- Auto-encoders
- ICA [Bell, Sejnowski - 1995]
- NMF [Lee, Seung – 1999]
- Gabor-Wavelets [Simoncelli, Freeman, Adelson, Heeger - 1998]
non causal model

system has energy:  \( E = \sum_i \gamma_i \log(1 + z_i^2) \)

Loss is:  \( L = \sum_j E(X_j) + \log \int_X e^{-E(X)} \)

Update weights:

\[
\frac{dL}{dw_{ij}} = \langle \frac{dE}{dw_{ij}} \rangle_{p^0} - \langle \frac{dE}{dw_{ij}} \rangle_{p^\infty}
\]

replaced by Contrastive Divergence:

\[
\Delta w_{ij} = \langle \frac{dE}{dw_{ij}} \rangle_{p^0} - \langle \frac{dE}{dw_{ij}} \rangle_{p^1}
\]
STRENGTHS
• Fast and easy inference
• Works for any number of code units
• Can be extended to multi-layer architectures

LIMITATIONS
• Training is expensive because of sampling
• Preprocessing is necessary to improve convergence
• How can we reconstruct a patch given its code?

Reverse Approach

In energy terms:
\[ E(X, Z, W) = \|X - WZ\|^2 + \lambda \sum S(Z_i) \]

Learning:
\[ \hat{W} = arg\!\min_W \left( \min_Z E(X, Z, W) \right) \]

1) find optimal Z, given X and W (inference, E-step)
2) update W given X and Z (learning, M-step)

But, the norm of each filter in W must be normalized by:
\[ \left[ \frac{\langle Z_i \rangle^2}{\sigma_{goal}^2} \right]^\alpha \]

STRENGTHS
- Simple learning algorithm
- Straightforward probabilistic interpretation
- Neural circuitry *might* work on the same principle of sparsity...

LIMITATIONS
- Normalization tweak
- Inference is expensive
- Preprocessing is required to get convergence
- A hierarchical approach might be needed
- Do real causes independently and linearly mix? No, they occlude one another.

Combining two approaches

- no normalization
- no slow inference
- no preprocessing

- easy learning

- no difficult learning
- no sampling
- no preprocessing

- fast inference

Diagram illustrating the combination of two approaches.
EBM for Sparse Representations

Decoder energy: $E_D(X, Z, W_D)$

Decoder reconstruction: $\|X - \text{Dec}(Z, W_D)\|^2$

Decoder: $\text{DECODER } W_D$

Rectified and sparsified code: $\widetilde{Z}$

Sp. Logistic

Encoder prediction: $\|Z - \text{Enc}(X, W_C)\|^2$

Encoder energy: $E_C(X, Z, W_C)$

Image $X$
EBM for Sparse Representations

\[ E_D(X, Z, W_D) \]

\[ \|X - \text{Dec}(Z, W_D)\|^2 \]

DECODER \( W_D \)

CODE \( Z \)

rectified and sparsified code

Sp. Logistic

[\text{Teh, Welling, Osindero, Hinton "Energy-Based Models for Sparse Overcomplete Representations" JMLR 2003}]

Direct module

ENCODER \( W_C \)

\[ \|Z - \text{Enc}(X, W_C)\|^2 \]

encoder energy

\[ E_C(X, Z, W_C) \]

\[ \text{decoder energy} \]

\[ \text{decoder reconstruction} \]

IMAGE \( X \)
EBM for Sparse Representations

\[ \|X - Enc(Z, W_D)\|^2 \]

Decoder Energy: \( E_D(X, Z, W_D) \)

DECODER \( W_D \)

Rectified and sparsified code

CODE \( Z \)

Decoder reconstruction

\[ \|Z - Enc(X, W_C)\|^2 \]

Encoder energy: \( E_C(X, Z, W_C) \)

Encoder prediction

EBM for Sparse Representations

\[ E_D(X, Z, W_D) \]

\[ \|X - Dec(Z, W_D)\|^2 \]

DECODER \( W_D \)

rectified and sparsified code

\[ Z \rightarrow \text{Sp. Logistic} \]

Non-linearity producing spikes

\[ \|Z - Enc(X, W_C)\|^2 \]

ENCODER \( W_C \)

encoder prediction

encoder energy

\[ E_C(X, Z, W_C) \]

decoder energy

\[ \text{decoder reconstruction} \]

IMAGE \( X \)

[Pillow, Paninski, Uzzell, Simoncelli, Chichilnisky “Prediction and decoding of RGC responses with a probabilistic spiking model” J.Neurosc. 05]
EBM for Sparse Representations

\[ P(X, Z | W_c, W_d) \propto \exp(-\beta E(X, Z, W_c, W_d)) \]

\[ E(X, Z, W_c, W_d) = E_C(X, Z, W_c) + E_D(X, Z, W_d) \]

\[ E_C(X, Z, W_c) = \frac{1}{2} \| Z - W_c X \|^2 = \frac{1}{2} \sum (z_i - W^i_c X)^2 \]

\[ E_D(X, Z, W_d) = \frac{1}{2} \| X - W_d Z \|^2 = \frac{1}{2} \sum (x_i - W^i_d Z)^2 \]
Inference

\[ \hat{Z} = \arg\min_Z E(X, Z, W) = \arg\min_Z [E_C(X, Z, W) + E_D(X, Z, W)] \]

- let \( Z(0) \) be the encoder prediction
- find code which minimizes total energy
- gradient descent optimization

Learning

\[ W \leftarrow W - \frac{\partial E(X, \hat{Z}, W)}{\partial W} \]

- using the optimal code, minimize \( E \) w.r.t. the weights \( W \)
- gradient descent optimization
Inference - step 1

\[ Z(0) = Wc X \]
Inference - step 1

Forward propagation

\[ \|X - \text{Dec}(Z, W_D)\|^2 \]

DECODER \( W_D \)

\( Z \)

Sp. Logistic

ENCODER \( W_C \)

\[ \|Z - \text{Enc}(X, W_C)\|^2 \]

CODE \( Z \)

\( E_D(X, Z, W_D) \)

decoder energy

\( E_C(X, Z, W_C) \)

encoder energy

\( W_C \times X \)

IMAGE \( X \)

\( X \)

ED decoder reconstruction
Inference - step 1

Back propagation of gradients w.r.t. $Z$

Decoder energy $E_D(X, Z, W_D)$

Encoder prediction $E_C(X, Z, W_C)$

Encoder prediction $\|Z - Enc(X, W_C)\|^2$

Decoder reconstruction $\|X - Dec(Z, W_D)\|^2$

Rectified and sparsified code $\tilde{Z}$
Learning - step 2

Forward propagation

\[ \| X - \text{Dec}(Z, W_D) \|^2 \]

**Decoder energy**

**Ed**

**Encoder energy**

**Ec**

**Image X**

**CODE Z**

**Optimal Z**

**Enc(x, WC)**

**Dec(Z, WD)**

**Sp. Logistic**

**DECODER W_D**

**ENCODER W_C**
Learning - step 2

Back propagation of gradients w.r.t $W$

- Encoder $W_c$ prediction
  $\|Z - \text{Enc}(X, W_c)\|^2$
  encoder energy $E_C(X, Z, W_c)$

- Decoder $W_d$ reconstruction
  $\|X - \text{Dec}(Z, W_d)\|^2$
  decoder energy $E_D(X, Z, W_d)$

- Sp. Logistic
  rectified and sparsified code $\bar{Z}$
  update $W_d$

- Encoder $W_c$ prediction
  encoder energy $E_C(X, Z, W_c)$

- Decoder $W_d$ reconstruction
  decoder energy $E_D(X, Z, W_d)$
  update $W_c$
Sparsifying non linearity mapping a code vector into a sparse code vector with components between 0 and 1.

- $Z_i$ input unit
- $\tilde{Z}_i$ corresponding output unit

$$\tilde{z}_i(k) = \frac{\eta e^{\beta z_i(k)}}{\xi_i(k)}, \quad i \in [1..m], \quad k \in [1..P]$$

$$\xi_i(k) = \eta e^{\beta z_i(k)} + (1 - \eta) \xi_i(k - 1) \quad \text{(recursive equation)}$$

Expanding the denominator, we have:

$$\tilde{z}_i(k) = \frac{\eta e^{\beta z_i(k)}}{\eta e^{\beta z_i(k)} + \eta (1 - \eta) e^{\beta z_i(k - 1)} + \eta (1 - \eta)^2 e^{\beta z_i(k - 2)} + ...}$$

This is a logistic function with an adaptive bias:

$$\tilde{z}_i(k) = \frac{1}{1 + e^{-\beta [z_i(k) - \frac{1}{\beta} \log \left( \frac{1 - \eta}{\eta} \xi_i(k - 1) \right)]}}$$
Sparsifying Logistic

\[ \bar{z}_i(k) = \frac{\eta e^{\beta z_i(k)}}{\xi_i(k)}, \quad i \in [1..m], \quad k \in [1..P] \]

\[ \xi_i(k) = \eta e^{\beta z_i(k)} + (1 - \eta) \xi_i(k - 1) \]

EXAMPLE
Input: random variable uniformly distributed in \([-1,1]\)
Output: a Poisson process with firing rate determined by \(\eta\) and \(\beta\).

- Increasing \(\beta\) the gain is increased and the output takes almost binary values.
- Increasing \(\eta\) more importance is given to the current sample, a spike will be more likely to occur.
"across samples" vs. "spatial" sparsity => no normalization is required

- $\zeta_i(k) = \eta e^{\beta z_i(k)} \xi_i(k)$, $i \in [1..m]$, $k \in [1..P]$

- $\xi_i(k) = \eta e^{\beta z_i(k)} + (1-\eta)\xi_i(k-1)$

- $\zeta$ is treated as a learned parameter after training

- $\zeta$ is saturated during training to allow units to have different sparsity.

- it is biologically inspired

[Pillow, Paninski, Uzzell, Simoncelli, Chichilnisky
"Prediction and Decoding of Retinal Ganglion Cell Responses with a Probabilistic Spiking Model"
J. Neuroscience 2005]

[Foldiak “Forming Sparse Representations by Local Anti-Hebbian Learning” Biological Cybernetics 1990]
Natural image patches - Berkeley

Berkeley data set

- 100,000 12x12 patches
- 200 units in the code
- $\eta = 0.02$
- $\beta = 1$
- learning rate 0.001
- L1, L2 regularizer 0.001
- fast convergence: < 30min.
Natural image patches - Berkeley

200 decoder filters (reshaped columns of matrix $W_d$)
Natural image patches - Berkeley

Encoder \textit{direct} filters (rows of }W_c\textit{)

Decoder \textit{reverse} filters (cols. of }W_d\textit{)
Handwritten digits - MNIST

- 60,000 28x28 images
- 196 units in the code
- $\eta$ 0.01
- $\beta$ 1
- learning rate 0.001
- L1, L2 regularizer 0.005

Encoder direct filters
Handwritten digits - MNIST

Forward propagation through encoder and decoder after training there is no need to minimize in code space.

Original: 7
Reconstructed without minimization: 7
Difference: 0.8

Original: 7
Reconstructed without minimization: 7
Difference: 0.8
CLASSIFICATION EXPERIMENTS

sparse representations & \textit{lenet6} (1->50->50->200->10)

• The baseline: \textit{lenet6} initialized randomly
  Test error rate: 0.70%. Training error rate: 0.01%.

• \textbf{Experiment 1}
  - Train on 5x5 patches to find 50 features
  - Use the scaled filters in the encoder to initialize the kernels in the first convolutional layer
  Test error rate: 0.60%. Training error rate: 0.00%.

• \textbf{Experiment 2}
  - Same as experiment 1, but training set augmented by elastically distorted digits (random initialization gives test error rate equal to 0.49%).
  Test error rate: 0.39%. Training error rate: 0.23%.
**STRENGTHS**
- Simple and fast learning algorithm
- Fast inference
- Simple preprocessing (no whitening)
- No normalization
- Can generate data from codes

**LIMITATIONS**
- No rigorous probabilistic interpretation
- No guarantee to find optimal solution
FUTURE WORK

- Multi-layer encoder and decoder
- Hierarchical architectures
- Applications: classification, clustering, density estimation, denoising
- Theoretical analysis of convergence
THANK YOU