Efficient Learning of Sparse Representations with an Energy-Based Model

Marc’Aurelio Ranzato, Christopher Poultney, Sumit Chopra, Yann Le Cun
Presented by Pascal Lamblin

February 14th, 2007
1 Introduction
   - Pre-processors and feature extractors
   - Coding and decoding

2 The Model
   - Energy Function and Architecture
   - The Sparsifying Logistic

3 Learning

4 Experiments
   - Feature extraction
   - Initialization of a Convolutional Neural Net
   - Hierarchical Extension: Learning Topographic Maps
Unsupervised Learning of Representations

- Methods like PCA, ICA, Wavelet decompositions...
- Usually, dimensionality is reduced
- Not necessary: *sparse overcomplete representations*
  - Improved separability of classes
  - Better interpretation (sum of basic components)
  - Biological parallel (early visual areas)
Usually, an encoder and a decoder (possibly sharing parameters)

Architecture for auto-encoders, restricted Boltzmann machines, PCA, ...

Sometimes, the encoder or decoder is absent (e.g., replaced by a sampling or minimization procedure)

Here we present a model with an encoder and a decoder
Learning Procedure

- Usually (PCA, auto-encoders, ...), we minimize a reconstruction error criterion
- Here, we also want sparsity in the code: another constraint
- Use of a Sparsifying Logistic module, between code and decoder
- Hard to learn through backprop only: optimize a global energy function, which depends also on the codes
- Iterative coordinate descent optimization (like EM)
Notation and Components

- The **input**: an image patch, $X$, as a vector
- The **encoder**: set of linear filters, rows of $W_C$
- The **code**: a vector $Z$
- The **Sparsifying Logistic**: transforms $Z$ into $\tilde{Z}$
- The **sparse code vector**: a vector $\tilde{Z}$ with components in $[0, 1]$
- The **decoder**: reverse linear filters, columns of $W_D$
Energy of the System

We want to minimize the global energy of the system, function of the model’s parameters $W_C$ and $W_D$, the free parameter $Z$, and the input $X$.

- Code prediction energy:
  \[
  E_C(X, Z, W_C) = \frac{1}{2} \| Z - W_C X \|^2
  \]
- Reconstruction energy:
  \[
  E_C(X, Z, W_D) = \frac{1}{2} \| X - W_D \bar{Z} \|^2
  \]

We have no hard equality constraint between $Z$ and $W_C X$, nor on $X$ and $W_D \bar{Z}$.

- $Z \neq W_C X$
- $W_D \bar{Z} \neq X$

Attention, accrochez-vous : qu’est-ce qui fait “Toin ! Toin !” ?
Energy of the System

We want to minimize the global energy of the system, function of the model’s parameters $W_C$ and $W_D$, the free parameter $Z$, and the input $X$.

- Code prediction energy:
  $$E_C(X, Z, W_C) = \frac{1}{2} \| Z - W_C X \|^2$$
- Reconstruction energy:
  $$E_C(X, Z, W_D) = \frac{1}{2} \| X - W_D \tilde{Z} \|^2$$

We have no hard equality constraint between $Z$ and $W_C X$, nor on $X$ and $W_D \tilde{Z}$.

- $Z \neq W_C X$
- $W_D \tilde{Z} \neq X$
Architecture of the energy-based model
In Theory

Let’s consider the $k$-th training sample

- $\bar{z}_i(k) = \frac{\eta e^{\beta z_i(k)}}{\zeta_i(k)}$
- $\zeta_i(k) = \eta e^{\beta z_i(k)} + (1 - \eta)\zeta_i(k - 1)$

Like a weighted softmax applied through time

- High values of $\beta$ makes the values more binary
- High values of $\eta$ increases the “firing rate”

The Sparsifying Logistic enforces sparsity through the examples for each individual component. There is no constraint of sparsity between the units of a code.
In Practice

- $\bar{z}_i(k) = \frac{\eta e^{\beta z_i(k)}}{\zeta_i(k)}$

- $\bar{z}_i(k) = \frac{1}{1 + (1 - \eta)\zeta_i(k - 1)}$

- $\bar{z}_i(k) = \left[1 + \frac{(1 - \eta)}{\eta} \zeta_i(k - 1)e^{-\beta z_i(k)}\right]^{-1}$

*We learn* $\zeta_i$ across the training set and fix it.

Logistic function, with fixed gain and learnt bias.

This version of the Sparsifying Logistic module is deterministic, and does not depend on the ordering of the samples.
Learning Procedure

We want to minimize:

\[ E(W_C, W_D, Z^1, \ldots, Z^P) = \sum_{i=1}^{P} \left( E_C(X^i, Z^i, W_C) + E_D(X^i, Z^i, W_D) \right) \]

by the procedure:

\[ \{W^*_C, W^*_D\} = \arg\min_{\{W_C, W_D\}} \left( \min_{Z^1, \ldots, Z^P} E(W_C, W_D, Z^1, \ldots, Z^P) \right) \]

1. Find the optimal \( Z^i \), given \( W_C \) and \( W_D \)
2. Update the weights \( W_C \) and \( W_D \), given \( Z^i \) found at step 1, in order to minimize the energy
3. Iterate until convergence
We consider only one sample $X$ at a time. The cost to minimize is

$$C = E_C(X, Z, W_C) + E_D(X, Z, W_D)$$

1. Initialize $Z$ by $Z_{init} = W_C X$
2. Minimize $C$ wrt $Z$, by gradient descent initialized at $Z_{init}$
3. Compute the gradient of $C$ wrt $W_C$ and $W_D$, and perform one gradient step

We iterate over all samples, until convergence.
So, What Happens?

- Only a few steps of gradient descent are necessary to minimize $Z$
- At the end of the process, even $Z_{init} = W_C X$ is accurate enough
- So $E_C(X, Z, W_C) = \frac{1}{2} \| Z - W_C X \|^2$ is minimized
- The reconstruction errors from $\tilde{Z}_{init}$ are also low
- So $E_D(X, Z, W_D) = \frac{1}{2} \| X - W_D \tilde{Z} \|^2$ is also minimized

The minimization procedure manages to minimize both energy terms

- Imposing the hard constraint $W_C X = Z$ does not work, because of the saturation of the sparsifying module
- An $L_1$ penalty term is added to $W_C$, and an $L_2$ penalty term to $W_D
Natural Image Patches

- 12 × 12 patches from the Berkeley segmentation data set
- Codes of length 200
- 30 minutes on a 2 GHz processor for 200 filters on 100,000 12 × 12 patches

Filters learnt by the decoder

- Spatially localized filters, similar to Gabor wavelets, like receptive fields of V1 neurons
- $W_C$ and $W_D'$ are really close after the optimization
On MNIST Digit Recognition Data Set

- Input is the whole $28 \times 28$ image (not a patch)
- Codes of length 196

Some encoder filters, and an example of digit reconstruction

- Stroke detectors are learnt
- Reconstruction: sum of a few “parts”
On MNIST

- Train filters on $5 \times 5$ image patches
- Codes of length 50
- Initialize a network with 50 features on layer 1 and 2, 50 on layer 3 and 4, 200 on layer 5, and 10 output units.

<table>
<thead>
<tr>
<th>Misclassification</th>
<th>Random</th>
<th>Pre-training</th>
</tr>
</thead>
<tbody>
<tr>
<td>No distortions</td>
<td>0.70%</td>
<td>0.60%</td>
</tr>
<tr>
<td>Distortions</td>
<td>0.49%</td>
<td>0.39%</td>
</tr>
</tbody>
</table>
Natural Image Patches

- 12 × 12 patches from the Berkeley segmentation data set
- Codes of length 400
- Close filters learn similar weights
Conclusion

- Energy-based model for unsupervised learning of sparse overcomplete representations
- Fast and accurate processing after learning
- Sparsification of each unit across the dataset seems easier than sparsification of each example across the code units
- Can be extended to non-linear encoder and decoder
- Sparse code can be used as input for another feature extractor
Questions?

Hopefully not...