Reducing the Dimensionality of Data with Neural Networks

Giorgi Pertaia, Charles Hernandez

Industrial and Systems Engineering

September 29, 2017
Outline

1. Auto-encoder
2. Restricted Boltzmann Machine
3. Contrastive Divergence
Autoencoder-decoder

Deep Autoencoder

Encoding DBN

Decoding DBN

Input

Output

Compressed Feature Vector
Restricted Boltzmann Machine

A Symmetrical, Bipartite, Bidirectional Graph with Shared Weights

\[ w_i \ldots w_n \]
Properties of a Restricted Boltzmann Machine

Definitions

\( a \in \mathbb{R}^n \) are the bias constants for the visible units.

\( b \in \mathbb{R}^m \) are the bias constants for hidden units.

\( W \in \mathbb{R}^{n \times m} \) is the weight matrix for the connections.

Energy Function

\[
E(v, h) = - \sum_i a_i v_i - \sum_j b_i h_j - \sum_{i,j} v_i W_{i,j} h_j
\]

Note: \( \frac{\partial E(v, h)}{\partial W_{i,j}} = -v_i h_j \)

Joint probability of configuration

\[
P(v, h) = \frac{1}{Z} e^{-E(v, h)}
\]

where

\[
Z = \sum_{v,h} e^{-E(v,h)}
\]
Deriving the gradient descent formula

What we need:

\[ \frac{\partial}{\partial W_{ij}} \log(P(v)) \]

Start from joint probability

\[ P(v, h) = \frac{1}{Z} e^{-E(v, h)} \]

Sum over \( h \)

\[ P(v) = \frac{1}{Z} \sum_{h} e^{-E(v, h)} \]

Take the log of the probability

\[ \log(P(v)) = \log\left(\frac{1}{Z} \sum_{h} e^{-E(v, h)}\right) \]
Log-Probabilities of visible unit configuration

\[
\log(P(v)) = \log \left( \frac{1}{Z} \sum_h e^{-E(v,h)} \right)
\]

Substitute for Z and simplify

\[
\log(P(v)) = \log \left( \sum_h e^{-E(v,h)} \right) - \log \left( \sum_{v,h} e^{-E(v,h)} \right)
\]

Take the Derivative

\[
\frac{\partial}{\partial W_{ij}} \log(P(v)) = \frac{\sum_h e^{-E(v,h)} v_i h_j}{\sum_h e^{-E(v,h)}} - \frac{\sum_{v,h} e^{-E(v,h)} v_i h_j}{\sum_{v,h} e^{-E(v,h)}}
\]
Substitute the following formulas to simplify the derivative

Probability formula

\[ P(v) = \frac{1}{Z} \sum_h e^{-E(v,h)} \]

Formula for Z

\[ Z = \sum_{v,h} e^{-E(v,h)} \]

Joint probability formula

\[ P(v, h) = \frac{1}{Z} e^{-E(v,h)} \]

Derivative of Log probability

\[
\frac{\partial}{\partial W_{ij}} \log(P(v)) = \frac{\sum_h e^{-E(v,h)} v_i h_j}{\sum_h e^{-E(v,h)}} - \frac{\sum_{v,h} e^{-E(v,h)} v_i h_j}{\sum_{v,h} e^{-E(v,h)}}
\]

Derivative of Log probability after substitutions

\[
\frac{\partial}{\partial W_{ij}} \log(P(v)) = \frac{\sum_h Z*P(v,h)*v_i h_j}{Z*P(v)} - \frac{\sum_{v,h} Z*P(v,h)*v_i h_j}{Z}
\]
Derivative of Log probability after substitutions

\[
\frac{\partial}{\partial W_{ij}} \log(P(v)) = \frac{\sum_h P(v,h) * v_i h_j}{P(v)} - \frac{\sum_{v,h} P(v,h) * v_i h_j}{1}
\]

Simplified Derivative of Log probability

\[
\frac{\partial}{\partial W_{ij}} \log(P(v)) = \sum_h P(h|v) v_i h_j - \sum_{v,h} P(v, h) v_i h_j
\]

Final Derivative of Log probability

\[
\frac{\partial}{\partial W_{ij}} \log(P(v)) = \mathbb{E}[v_i h_j | v] - \mathbb{E}[v_i h_j]
\]
Sampling

- It is easy to calculate \( v_i \mathbb{E}[h_i | v] \)
- It is hard to calculate \( \mathbb{E}[v_i h_i] \) sampling \( v_i h_i \) directly is hard.

Gibbs sampling

- We can use Gibbs sampling
- we have \( p(v|h) \) and \( p(h|v) \).
Contrastive Divergence

Much faster alternative to Gibbs sampling

Algorithm:
1. initialize the visible units to training vectors.
2. calculate the binary states of hidden units.
3. "reconstruct" the binary states of visible units given hidden units.
4. go to step 1

$$\Delta w = \epsilon (v_i \mathbb{E}(h|v) - \mathbb{E}(\hat{v}_i \hat{h}_i))$$
Unrolling the Autoencoder-decoder
The Autoencoder-decoder is initialized by training a sequence of Restricted Boltzmann Machines using Contrastive Divergence.

This process was shown to work even with real valued data.

Once initialized, the weights and biases are fine tuned using gradient descent on the entire autoencoder-decoder network.