A Fast Learning Algorithm for Deep Belief Nets

Geoffrey E. Hinton, Simon Osindero & Yee-Whye Teh

Presented by Zhiwei Jia
Deep Belief Nets

- Stacked Restricted Boltzmann Machine (RBM)
- RBM
  - Nice property: given one side, easy to sample the other.

\[
E(v, h) = - \sum_{i \in \text{visible}} a_i v_i - \sum_{j \in \text{hidden}} b_j h_j - \sum_{i,j} v_i h_j w_{ij} \\
p(v, h) = \frac{1}{Z} e^{-E(v, h)} \\
p(v) = \frac{1}{Z} \sum_h e^{-E(v, h)} \\
p(h_j = 1 | v) = \sigma(b_j + \sum_i v_i w_{ij}) \\
p(v_i = 1 | h) = \sigma(a_i + \sum_j h_j w_{ij})
\]
Training a RBM

- Maximum Likelihood Estimation (MLE) via Stochastic Gradient Ascent
  \[
  \frac{\partial \log p(v)}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}
  \]
  - \(<x>_p\) means the expectation of \(x\) w.r.t. the distribution \(p\)

- Expectation of \(v_i h_j\) w.r.t. the model distribution is hard to obtain
  - Via Contrastive Divergence, which performs \(n\) pass of alternating Gibbs Sampling in each gradient ascent step.

\[
p(h_j = 1 \mid v) = \sigma(b_j + \sum_i v_i w_{ij})
\]
\[
p(v_i = 1 \mid h) = \sigma(a_i + \sum_j h_j w_{ij})
\]
Training in Directed Belief Network

● Key Challenge: hard to infer the hidden variables on directed belief nets
  ○ Posterior distribution cannot be easily factored as $P(y|x) = \prod_j P(y_j|x)$
  ○ MCMC is too slow
● Approach
  ○ With a stacked RBM structure, we can
    ■ First perform greedy based per-layer training
    ■ Follow with ‘backfitting’, i.e., jointly fine-tune all layers
Greedy Learning in DBN

1. First train a layer of features that receive input directly from the input images.
2. Then treat the activations of the trained features as if they were pixels and learn features of features in the next hidden layer.
Why Greedy Learning Works?

● Each time we add another layer of features we improve a variational lower bound on the log probability of the training data.

● Given energy function  

\[ E(v^0, h^0) = -[\log p(h^0) + \log p(v^0|h^0)] \]

● We have the following variational lower bound

\[
\log p(v^0) \geq \sum_{\text{all } h^0} Q(h^0|v^0) \left[ \log p(h^0) + \log p(v^0|h^0) \right] \\
- \sum_{\text{all } h^0} Q(h^0|v^0) \log Q(h^0|v^0)
\]

● This becomes an equality iff \( Q(\cdot|v^0) \) is the true posterior distribution.

● Contrastive Divergence learning of this layer makes the bound tighter
Why Greedy Learning Works? (cont.)

- When weights of the first RBM is fixed, maximizing
  \[ \sum_{\text{all } h^0} Q(h^0|v^0) \left[ \log p(h^0) + \log p(v^0|h^0) \right] - \sum_{\text{all } h^0} Q(h^0|v^0) \log Q(h^0|v^0) \]
  is equivalent to maximizing
  \[ \sum_{\text{all } h^0} Q(h^0|v^0) \log p(h^0) \]

- To train the next RBM is to maximize \( \log p(h^0) \), which in turn increases \( \log p(v^0) \)
Back-fitting Lower Layers via Up-Down Algorithm

- Learning the weight matrices in a greedy way is efficient but not optimal.
  - Given learned weights in higher layers, the weights are not optimal for the lower layers.
  - We can fine-tune the features to improve the overall model.
- Untie ‘recognition’ (bottom-up) and ‘generative’ (top-down) weights of all but the last RBM.
  1. Do a bottom-up pass: for each ‘untied’ RBM, fix bottom-up weights and train top-down weights by MLE.
  2. Use contrastive divergence to train the top-level RBM.
  3. Do a top-down pass: for each ‘untied’ RBM, fix top-down weights and train bottom-up weights by MLE.
Supervised Training on MNIST

1. Use greedy algorithm was used to train each layer of weights separately, with each layer trained for epochs (batch size is 100).
2. During training of the top RBM, labels are provided as extra inputs; sampling label neurons according to a softmax probability.
3. 300 epochs of training via the up-down algorithm with increasing number of Gibbs sampling in top RBM’s training.
4. Further training process on training & validation set.
Performance on MNIST

● How to test?
  ○ Start with a neutral state of the label units
  ○ Do an up-pass from the input image
  ○ Turn on each of the label units in turn and compute the corresponding exact energy
  ○ Making stochastic up-pass deterministic reduces noise and thus boosts performance

● With 1.25% test error.
Generate Image Samples

- Perform Gibbs sampling in the top RBM and do a single down pass
- Can clamp label neurons to a particular class.

With 1000 steps of the alternating Gibbs sampling.
Another Way to Look At DBN

- In general, directed belief nets are hard to train/inference.
- Undirected graphical model RBM has nice algorithm (CD).
- An infinite logistic belief net with tied weights is equivalent to an (undirected) RBM.
  - For infinite logistic belief net:
    \[
    \frac{\partial \log p(v^0)}{\partial w_{ij}} = \langle h_j^0(v_i^0 - v_i^1) \rangle \\
    + \langle v_i^1(h_j^0 - h_j^1) \rangle \\
    + \langle h_j^1(v_i^1 - v_i^2) \rangle \\
    + \ldots
    \]
  - For RBM:
    \[
    \frac{\partial \log p(v^0)}{\partial w_{ij}} = \langle v_i^0 h_j^0 \rangle - \langle v_i^\infty h_j^\infty \rangle
    \]
Another Way to Look At DBN

- Greedily train a layer in an infinite logistic belief net is equivalent to greedily train an RBM in a deep belief net.
- Maximizing the log probability of the data is exactly the same as minimizing \( KL(P^0||P_\phi^\infty) \).
- Contrastive divergence learning, in which only \( n \) number of Gibbs sampling is performed, corresponds to minimizing

\[
KL(P^0||P_\phi^\infty) - KL(P_\phi^n||P_\phi^\infty)
\]