Restricted Boltzmann Machine and Deep Belief Network

Jarod Klion

CAP5771

April 11, 2022

Table of Contents

Boltzmann Distribution (1868)

Definition

The probability that a given system is in a specific state based on the energy of that state and the system's temperature.

If we have several particles $\{x_i\}_{i=1}^{d}$treated as random variables with each particle having a random state, the probability mass function and corresponding variables are given as:

$$\mathbb{P}(x) = \frac{e^{-\beta E(x)}}{Z}, \qquad Z := \sum_{x \in \mathbb{R}^d} e^{-\beta E(x)}, \qquad \beta := \frac{1}{k_B T} \tag{1}$$

where E(x) is the energy of variable x, Z is the canonical partition function used for normalization, and β is the reciprocal of thermodynamic temperature.

Ising Model (1925)

- Simplest explanation of ferromagnetism
  - Particles have either spin +1 or -1 and are assumed to be interacting with nearest neighbors.
  - $x_i \in \{-1, +1\}, \forall i \in \{1, ..., d\}$
- Uses Boltzmann distribution with energy E(x) given by:

$$E(x) = -\sum_{(i,j)} J_{ij} x_i x_j, \tag{2}$$

  where J_{ij} is a coupling parameter dependent on the model.
- Ferromagnetic: $J_{ij} \geq 0$, so parallel spins
- Anti-ferromagnetic: $J_{ij} < 0$, so antiparallel spins
- BM and RBM are Ising models
  - Coupling parameter considered as a weight
  - Energy-based learning methods

Hopfield Network (1982)

- Ising model of a neural network
  - Network of neurons $\{x_i\}_{i=1}^d$
  - States and outputs are binary: $x_i \in \{-1, +1\}$
- Weights between neurons i and j denoted w_{ij}
  - Learned through Hebbian learning rule:

$$w_{ij} := \begin{cases} x_i \times x_j & \text{if } i \neq j, \\ 0 & \text{otherwise.} \end{cases}$$

- Outputs determined from input if threshold θ passed:

$$x_i := \begin{cases} +1 & \text{if } \sum_{j=1}^d w_{ij} x_j \geq \theta, \\ -1 & \text{otherwise.} \end{cases}$$

- RBM model is a Hopfield network
  - weights learned with MLE, not Hebbian learning

Gibbs Sampling (1984)

- Use d conditional distributions to draw samples from a d-dimensional multivariate distribution $\mathbb{P}(x)$
  - Assumes it is simple to draw samples from
- Follows an algorithm:
  1. Start from random d-dimensional vector in range of data
  2. Sample 1st dimension of 1st sample from the distribution of the 1st dimension conditioned on the others
     - j-th dimension: $x_j \sim P(x_j | x_1, ..., x_{j-1}, x_{j+1}, ..., x_d)$
     - Do this for all dimensions until all dimensions of 1st sample are drawn
  3. Repeat this iteratively for all samples

Structure of the BM/RBM

Generative model named after the distribution used in it.

- Visible layer:
  - Layer we can see such as data
  - $\mathbf{v} = [v_1, ..., v_d] \in \mathbb{R}^d$
- Hidden layer:
  - Layer of latent variables
    - Meaningful features
    - Embeddings of the visual data
  - $\mathbf{h} = [h_1, ..., h_d] \in \mathbb{R}^p$
- Meaningful connection between visible and hidden layers

Graphical Connection Links

- $\mathbf{W} = [w_{ij}] \in \mathbb{R}^{d \times p}$:
  - w_{ij} is the link between v_i and h_j
  - Symmetric, i.e., $w_{ij} = w_{ji}$
- $\mathbf{L} = [l_{ij}] \in \mathbb{R}^{d \times d}$:
  - l_{ij} is the link between v_i and v_j
  - $l_{ii} = 0 \forall i$
- $\mathbf{J} = [j_{ij}] \in \mathbb{R}^{p \times p}$:
  - j_{ij} is the link between h_i and h_j
  - $j_{ii} = 0 \forall i$
- $\mathbf{b} = [b_1, ..., b_d] \in \mathbb{R}^{d}$:
  - b_i is the bias link for v_i
- $\mathbf{c} = [c_1, ..., c_d] \in \mathbb{R}^{p}$:
  - c_i is the bias link for h_i

Difference between BM and RBM

Restricted Boltzmann machines are a specific form of the Boltzmann machine.

- Restricts links to be $L = J = 0$

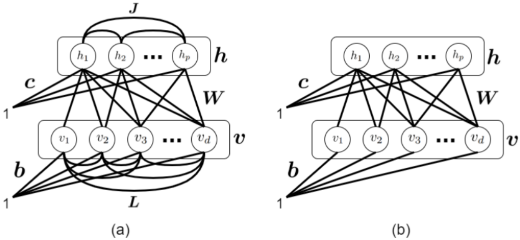

Figure: The structures of (a) a Boltzmann machine and (b) a restricted Boltzmann machine.

- Yields energy $E(\mathbf{v}, \mathbf{h}) := -\mathbf{b}^\top \mathbf{v} - \mathbf{c}^\top \mathbf{h} - \mathbf{v}^\top \mathbf{W} \mathbf{h}$
- Joint Boltzmann distribution: $\mathbb{P}(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v}, \mathbf{h})}$

Sampling Variables

Generate hidden and visible units in training and evaluation phases of RBM as follows:

- Input visible dataset $\mathbf{v}$
- Get initialization or do random initialization of $\mathbf{v}$
- Iteratively sample until burn-out convergence
  - **for** *j from 1 to p* **do** $h_j^{(\nu)} \sim \mathbb{P}(h_j \mid \mathbf{v}^{(\nu)})$
  - **for** *i from 1 to d* **do** $v_i^{(\nu+1)} \sim \mathbb{P}(v_i \mid \mathbf{h}^{(\nu)})$

BM and RBM are generative models.

- Can represent d-dimensional observation through generation of any number of p-dimensional hidden variables using Gibbs sampling.

Training with Maximum Likelihood Estimation

- Need to learn weights of links to generate new units
- Log-likelihood:

$$\ell(\mathbf{W}, \mathbf{b}, \mathbf{c}) = \sum_{i=1}^{n} \log\left(\sum_{\mathbf{h}\in\mathbb{R}^p} e^{-E(\mathbf{v}_i,\mathbf{h})}\right) - n\log \sum_{\mathbf{v}\in\mathbb{R}^d} \sum_{\mathbf{h}\in\mathbb{R}^p} e^{-E(\mathbf{v},\mathbf{h})}$$

- Gradient of each parameter $(\mathbf{W}, \mathbf{b}, \mathbf{c})$ for gradient ascent using Maximum Likelihood Estimation:

$$\nabla_{\mathbf{W}}\ell = \sum_{i=1}^{n} \mathbf{v}_i \widehat{\mathbf{h}}_i^\top - n\mathbb{E}_{\sim\mathbb{P}(\mathbf{h},\mathbf{v})}[\mathbf{v}\mathbf{h}^\top], \tag{3}$$

$$\nabla_{\mathbf{b}}\ell = \sum_{i=1}^{n} \mathbf{v}_i - n\mathbb{E}_{\sim\mathbb{P}(\mathbf{h},\mathbf{v})}[\mathbf{v}], \tag{4}$$

$$\nabla_{\mathbf{c}}\ell = \sum_{i=1}^{n} \widehat{\mathbf{h}}_i^\top - n\mathbb{E}_{\sim\mathbb{P}(\mathbf{h},\mathbf{v})}[\mathbf{h}], \tag{5}$$

$$\text{where } \widehat{\mathbf{h}}_i := \mathbb{E}_{\sim\mathbb{P}(\mathbf{h}|\mathbf{v}_i)}[\mathbf{h}]$$

Contrastive Divergence

- Exact computation of MLE extremely difficult
  - Should approximate instead!
- Negative sampling
  - Train iteratively but less ambitiously each iteration
  - Learn which outputs are wrong to avoid
  - Generate correct observations by avoiding negative samples

Input: training data $\{\boldsymbol{x}_i\}_{i=1}^n$
Randomly initialize $\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}$
while *not converged* **do**
  Sample a mini-batch $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_m\}$ from training dataset $\{\boldsymbol{x}_i\}_{i=1}^n$ (n.b. we may set $m = n$)
  // Gibbs sampling for each data point:
  Initialize $\widehat{\boldsymbol{v}}_i^{(0)} \leftarrow \boldsymbol{v}_i$ for all $i \in \{1, \dots, m\}$
  for i *from* 1 *to* m **do**
    Algorithm 1 $\leftarrow \widehat{\boldsymbol{v}}_i^{(0)}$
    $\{h_i\}_{i=1}^p, \{v_i\}_{i=1}^d \leftarrow$ Last iteration of Algorithm 1
    $\widetilde{\boldsymbol{h}}_i \leftarrow [h_1, \dots, h_p]^\top$
    $\widetilde{\boldsymbol{v}}_i \leftarrow [v_1, \dots, v_d]^\top$
    $\widehat{\boldsymbol{h}}_i \leftarrow \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h}|\boldsymbol{v}_i)}[\boldsymbol{h}]$
  // gradients:
  $\nabla_{\boldsymbol{W}} \ell(\theta) \leftarrow \sum_{i=1}^m \boldsymbol{v}_i \widehat{\boldsymbol{h}}_i^\top - \sum_{i=1}^m \widetilde{\boldsymbol{h}}_i \widetilde{\boldsymbol{v}}_i^\top$
  $\nabla_{\boldsymbol{b}} \ell(\theta) \leftarrow \sum_{i=1}^m \boldsymbol{v}_i - \sum_{i=1}^m \widetilde{\boldsymbol{v}}_i$
  $\nabla_{\boldsymbol{c}} \ell(\theta) \leftarrow \sum_{i=1}^m \widehat{\boldsymbol{h}}_i - \sum_{i=1}^m \widetilde{\boldsymbol{h}}_i$
  // gradient descent for updating solution:
  $\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \nabla_{\boldsymbol{W}} \ell(\theta)$
  $\boldsymbol{b} \leftarrow \boldsymbol{b} - \eta \nabla_{\boldsymbol{b}} \ell(\theta)$
  $\boldsymbol{c} \leftarrow \boldsymbol{c} - \eta \nabla_{\boldsymbol{c}} \ell(\theta)$
Return $\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}$

Figure: Training RBM using contrastive divergence

Stacking RBM Models

- Vanishing gradients typical problem before ReLu and dropout methods
  - Random initial weights unsuitable for backpropagation
  - RBM training yields good weight initialization
- Consider a neural network of ℓ layers with p_ℓ neurons in the ℓ-th layer:
  - Every 2 successive layers are considered as one RBM
  - Train using belief propagation (RBM training)
  - Network can get as deep as desired
    - Hence, referred to as *Deep Belief Network*

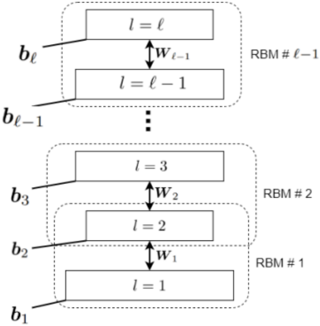

Figure: Pre-training a DBN by considering every pair of layers as an RBM.

Training a Deep Belief Network

- Training dataset used as the visible variable of the 1st pair of layers
  - Train weights and biases with algorithm 2
- Generate n hidden variables with Gibbs sampling
  - These will be visible variables in 2nd RBM
- Repeat process until all layer pairs are trained
  - Greedy approach to training
  - Produces good initialized weights and biases
- Fine-tune weights and biases using backpropagation

Input: training data $\{\boldsymbol{x}_i\}_{i=1}^n$
// pre-training:
for l *from* 1 *to* $\ell - 1$ **do**
  if $l = 1$ **then**
    $\{\boldsymbol{v}_i\}_{i=1}^n \leftarrow \{\boldsymbol{x}_i\}_{i=1}^n$
  else
    // generate n hidden variables of previous RBM:
    $\{\boldsymbol{h}_i\}_{i=1}^n \leftarrow$ Algorithm 1 for $(l-1)$-th RBM $\leftarrow \{\boldsymbol{v}_i\}_{i=1}^n$
    $\{\boldsymbol{v}_i\}_{i=1}^n \leftarrow \{\boldsymbol{h}_i\}_{i=1}^n$
  $\boldsymbol{W}_l, \boldsymbol{b}_l, \boldsymbol{b}_{l+1} \leftarrow$ Algorithm 2 for l-th RBM $\leftarrow \{\boldsymbol{v}_i\}_{i=1}^n$
// fine-tuning using backpropagation:
Initialize network with weights $\{\boldsymbol{W}_l\}_{l=1}^{\ell-1}$ and biases $\{\boldsymbol{b}_l\}_{l=2}^{\ell}$.
$\{\boldsymbol{W}_l\}_{l=1}^{\ell-1}, \{\boldsymbol{b}_l\}_{l=1}^{\ell} \leftarrow$ Backpropagate the error of loss fro several epochs.

Figure: Training a deep belief network

Improvements over RBM and DBN

- Convolutional DBN
- Recurrent RBM
  - Handle temporal information of data
- Other energy-based models
  - Helmholtz machine
- Deep Boltzmann Machine (DBM)
  - Document processing
  - Face modeling

Conclusion

- Background of BM, RBM, and DBN
  - Boltzmann Distribution
  - Ising Model
  - Hopfield Network
- Overview of BM and RBM
  - Structure
  - Generating variables with Gibbs sampling
  - Training with contrastive divergence
- Overview of DBN
  - Stack of RBMs
  - Pre-trained then fine-tuned

References

- Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, Mark Crowley: Restricted Boltzmann Machine and Deep Belief Network: Tutorial and Survey. CoRR abs/2107.12521 (2021)
- Cai, W. (2011, February 25). Ising model. Retrieved April 10, 2022, from http://micro.stanford.edu/ caiwei/me334/Chap12_Ising_Model_v04.pdf
- Hopfield J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences of the United States of America, 79(8), 2554–2558. https://doi.org/10.1073/pnas.79.8.2554