# Sharif University

## Machine Learning Project

*Graph models and Boltzmann machine*

*Dr.Yasaei*

*Adel Movahedian Mehdi Zolfaghari*

# Contents

# 1 Secure Boltzmann Machine

In the previous phase, you became familiar with the boltzmann machine, its usage, and how to update its parameters. In this part of the second phase of the project, we intend to implement the boltzmann machine in a simple manner.

Assume two groups, $A$ and $B$, aim to send messages $m_A$ and $m_B$ to each other securely. These two groups want to utilise the boltzmann machine in such a way that eavesdroppers cannot deduce the values of these messages. For this reason, we use one of the encryption algorithms, and in subsequent sections, you will become familiar with it.

## 1.1 Theoretical Question 1

Fully explain the ElGamal encryption algorithm and its equations.

Answer 1: ElGamal Encryption Algorithm

The ElGamal encryption algorithm is a public-key cryptosystem that provides security through the difficulty of solving the discrete logarithm problem. The algorithm operates as follows:

- **Key Generation:**
  1. Choose a large prime number $p$ and a generator $g$ of the multiplicative group $\mathbb{Z}_p^*$.
  2. Select a private key $x$ such that $1 \leq x \leq p-2$.
  3. Compute the public key $y$ as:

     $$y = g^x \mod p$$

     The public key is $(p, g, y)$, and the private key is $x$.

- **Encryption:** To encrypt a message $m$ (where $m \in \mathbb{Z}_p^*$):
  1. Select a random integer $k$ such that $1 \leq k \leq p-2$.
  2. Compute the ciphertext as a pair $(c_1, c_2)$, where:

     $$c_1 = g^k \mod p, \quad c_2 = m \cdot y^k \mod p$$

- **Decryption:** To decrypt the ciphertext $(c_1, c_2)$, compute:

  $$s = c_1^x \mod p$$

  Then recover the original message $m$ using:

  $$m = c_2 \cdot s^{-1} \mod p$$

  Here, $s^{-1}$ is the modular multiplicative inverse of $s$ modulo $p$.

## 1.2 Theoretical Question 2

Explain the partial decryption process in detail.

Answer 2: Partial Decryption Process

Partial decryption is a step-by-step collaborative process where multiple parties jointly decrypt an encrypted message. Each party contributes a partial decryption share without revealing the full private key.

- **Encrypted Message:** The ciphertext is a pair $(c_1, c_2)$, where:

$$c_1 = g^k \mod p, \quad c_2 = m \cdot y^k \mod p$$

- **Key Sharing:** The private key $x$ is split into $n$ shares $x_1, x_2, \dots, x_n$, such that:

$$x = x_1 + x_2 + \cdots + x_n$$

  Each party $i$ knows only their share $x_i$.

- **Partial Decryption by Each Party:** Each party computes their partial decryption share:

$$s_i = c_1^{x_i} \mod p$$

- **Combining Partial Shares:** All partial shares are combined to compute the complete decryption key $s$:

$$s = s_1 \cdot s_2 \cdot \cdots \cdot s_n \mod p = c_1^x \mod p$$

- **Final Decryption:** The original message $m$ is recovered as:

$$m = c_2 \cdot s^{-1} \mod p$$

  Here, $s^{-1}$ is the modular multiplicative inverse of $s$ modulo $p$.

**Advantages of Partial Decryption:**

- Enhanced security: No single party has access to the full private key.
- Trust distribution: Decryption requires collaboration among all parties.
- Suitable for distributed systems, such as secure voting or shared data access.

## 1.3 Theoretical Question 3

Devise an algorithm, inspired by the ones described above, to securely compute the product of two numbers while preserving privacy. Assume Group $A$ knows $M$, and Group $B$ knows $N$. The goal is to collaboratively calculate $M \times N$ without revealing the private values.

Answer 3: Algorithm for Secure Multiplication

To compute the product $M \times N$ privately between Group $A$ and Group $B$, the following steps are taken:

## 1: Initialisation

- Group $A$ holds the private value $M$, and Group $B$ holds $N$.

## 2: Group $A$ Computes and Sends

- Group $A$ generates a random value $r$ and computes:

$$P_A = M + r$$

- Group $A$ sends $P_A$ to Group $B$.

## 3: Group $B$ Processes and Returns

- Group $B$ calculates:

$$P_B = P_A \times N = (M + r) \times N = M \times N + r \times N$$

- Group $B$ sends $P_B$ back to Group $A$.

## 4: Group $A$ Finalises the Calculation

- Group $A$ subtracts $r \times N$ from $P_B$ to obtain:

$$\text{Result} = P_B - r \times N = M \times N$$

## Comparison to Previous Algorithms

This approach mirrors the structure of the earlier sigmoid algorithm:

- A random value $r$ ensures privacy, much like the randomness used in secure weighted sum calculations.
- The collaboration involves exchanging intermediate results, a hallmark of the previous algorithms.

## Benefits:

- Group $A$ and Group $B$ preserve the confidentiality of their respective inputs.
- The method is lightweight and can integrate into broader secure computation frameworks.

# 2 Discriminative Boltzmann Machines

In the previous section, you became familiar with the Boltzmann Machine as a simple but powerful generative model. Now, we want to use the capabilities of this model, and generally energy-based models, for data classification. In this section, you will learn about Discriminative Boltzmann Machines (DRBMs). Unlike standard RBMs, which are designed to learn the joint distribution of data ($p(x, y)$), these models are specifically used to learn the conditional distribution between data and their labels ($p(y|x)$).

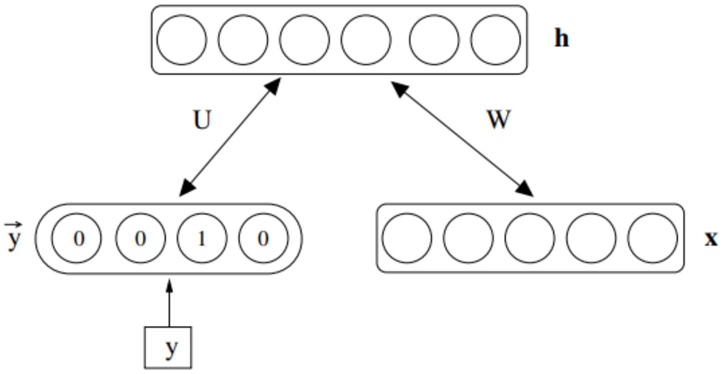

Figure 1: Discriminative Boltzmann Machine

The Discriminative Boltzmann Machine is constructed by modifying the energy function of a standard RBM. In a DRBM, the data label $y$ is added as part of the visible layer of the model. The energy function in a DRBM is defined as follows:

$$E(x, y, h) = -\sum_{i,j} W_{ij}\, x_i\, h_j - \sum_i b_i\, x_i - \sum_k c_k\, y_k - \sum_{j,k} d_{jk}\, h_j\, y_k$$

where:

- $x$ represents the input data vector.
- $y$ represents the label vector (one-hot encoded).
- $h$ represents the hidden units vector.
- $W$, $b$, $c$, and $d$ are the model's parameters (weight matrices and bias vectors).

For binarising the labels and using

$$p(x, y, h) \propto \exp(-E(x, y, h)),$$

one-hot encoding is used for labels for optimal and logical use.

The goal is proper classification. As a result, we seek to maximise the likelihood $p(y|x)$. This distribution represents the probability of assigning a specific label to the input data, and $F$ represents the free energy:

$$p(y|x) = \frac{\exp(-F(x, y))}{\sum_{y'} \exp(-F(x, y'))}$$

## 2.1 Theoretical Question 4

Given the energy function under consideration, answer the following questions:

1. **Calculate the joint distribution $p(x, y)$.**

Question 4.1 Answer

**1. Calculation of the Joint Distribution $p(x, y)$:** The given energy function is

$$E(x, y, h) = -\sum_{i,j} W_{ij}\, x_i\, h_j - \sum_i b_i\, x_i - \sum_k c_k\, y_k - \sum_{j,k} d_{jk}\, h_j\, y_k.$$

To calculate the joint distribution $p(x, y)$, we marginalise over the hidden variables $h$:

$$p(x, y) = \frac{1}{Z} \sum_h \exp(-E(x, y, h)),$$

where $Z$ is the partition function defined by

$$Z = \sum_{x,y,h} \exp(-E(x, y, h)).$$

After performing the summation over $h$, we obtain

$$p(x, y) \propto \exp\left(\sum_i b_i\, x_i + \sum_k c_k\, y_k\right) \prod_j \left(1 + \exp\left(\sum_i W_{ij}\, x_i + \sum_k d_{jk}\, y_k\right)\right).$$

2. **Calculate the conditional distribution $p(y|x)$.**

Question 4.2 Answer

**2. Calculation of the Conditional Distribution $p(y|x)$:** The conditional distribution is derived as

$$p(y|x) = \frac{p(x, y)}{p(x)} = \frac{\exp(-F(x, y))}{\sum_{y'} \exp(-F(x, y'))},$$

where the free energy $F(x, y)$ is defined by

$$F(x, y) = -\log \sum_h \exp(-E(x, y, h)).$$

### Question 4.2 Answer

Substituting $E(x, y, h)$ and carrying out the summation, we obtain:

$$F(x,y) = -\sum_i b_i\, x_i - \sum_k c_k\, y_k - \sum_j \log\left(1 + \exp\left(\sum_i W_{ij}\, x_i + \sum_k d_{jk}\, y_k\right)\right).$$

Thus, the conditional probability becomes

$$p(y|x) = \frac{\exp\big(\sum_i b_i\, x_i + \sum_k c_k\, y_k + \sum_j \log\big(1+\exp\big(\sum_i W_{ij}\, x_i + \sum_k d_{jk}\, y_k\big)\big)\big)}{\sum_{y'} \exp\big(\sum_i b_i\, x_i + \sum_k c_k\, y'_k + \sum_j \log\big(1+\exp\big(\sum_i W_{ij}\, x_i + \sum_k d_{jk}\, y'_k\big)\big)\big)}$$

This expression closely resembles a softmax function.

3. **Write the optimisation problem for the log-likelihood $\mathcal{L}(y|x)$ and its gradient mathematically. Can this problem be solved with conventional optimisation methods? If yes, explain. If not, provide a solution.**

### Question 4.3 Answer

**3. Optimisation Problem for the Log-Likelihood $\mathcal{L}(y|x)$ and Its Gradient:** The objective is to maximise the log-likelihood over the training data:

$$\mathcal{L} = \sum_{(x,y)\in\text{data}} \log p(y|x).$$

Substituting the expression for $p(y|x)$ gives:

$$\mathcal{L} = \sum_{(x,y)} \left[ -F(x,y) - \log \sum_{y'} \exp\left(-F(x,y')\right) \right].$$

The gradient with respect to a parameter, say $W_{ij}$, is then given by:

$$\nabla_{W_{ij}} \mathcal{L} = \mathbb{E}_{\text{data}} \left[ \frac{\partial F(x,y)}{\partial W_{ij}} \right] - \mathbb{E}_{\text{model}} \left[ \frac{\partial F(x,y)}{\partial W_{ij}} \right].$$

The first term is the expectation under the data distribution and the second term is the expectation under the model distribution.

**Solvability with Conventional Methods:**
In theory, this optimisation problem can be solved with conventional methods. However, computing the exact gradient is computationally expensive because it involves summing over all possible labels $y'$ and hidden states $h$.

**Solution:**
Approximate methods such as Contrastive Divergence (CD), Gibbs Sampling, or Stochastic Gradient Descent (SGD) with gradient approximations are typically used.

4. **Write the optimisation problem for the log-likelihood $\mathcal{L}(x)$ and its gradient mathematically. Can this problem be solved with conventional optimisation methods? If yes, explain. If not, provide a solution.**

Question 4.4 Answer

**4. Optimisation Problem for $\mathcal{L}(x)$ and Its Gradient:** The discriminative objective for maximising $\log p(y|x)$ remains the same:

$$\mathcal{L} = \sum_{(x,y)} \log p(y|x),$$

with the gradient computed in an analogous manner:

$$\nabla_{W_{ij}} \mathcal{L} = \mathbb{E}_{\text{data}} \left[ \frac{\partial F(x,y)}{\partial W_{ij}} \right] - \mathbb{E}_{\text{model}} \left[ \frac{\partial F(x,y)}{\partial W_{ij}} \right].$$

The challenge again is the computation of the expectation under the model distribution, which is addressed by the same approximate methods as mentioned earlier.

5. **Derive an expression for the free energy function $F(x, y)$.**

Question 4.5 Answer

**5. Calculation of the Free Energy $F(x, y)$:** The free energy is defined by

$$F(x,y) = -\log \sum_{h} \exp(-E(x,y,h)).$$

Substituting the complete energy function, we obtain:

$$F(x,y) = -\sum_i b_i\, x_i - \sum_k c_k\, y_k - \sum_j \log\left(1 + \exp\left(\sum_i W_{ij}\, x_i + \sum_k d_{jk}\, y_k\right)\right).$$

This expression represents the free energy as a combination of linear terms in $x$ and $y$ and a log-sum-exponential term capturing the contribution of the hidden units.

## 2.2 Theoretical Question 5

Prove the following relation:

$$\frac{\partial \log p(y|x)}{\partial \theta} = \sum_j \sigma(o_{yj}(x)) \cdot \frac{\partial o_{yj}(x)}{\partial \theta} - \sum_{y'} \sum_j \sigma(o_{y'j}(x)) \cdot \frac{\partial o_{y'j}(x)}{\partial \theta}$$

where:

$$o_{yj}(x) = c_j + \sum_k W_{jk} x_k + U_{jy}$$

and therefore we can perform accurate gradient calculation.

### Question 5 Answer

#### 1. Problem Statement

The goal is to prove the following relation for the gradient of the conditional log-likelihood $\log p(y|x)$ with respect to the model parameters $\theta$:

$$\frac{\partial \log p(y|x)}{\partial \theta} = \sum_j \sigma(\omega_{ij}(x)) \cdot \frac{\partial \omega_{ij}(x)}{\partial \theta} - \sum_{y',j} \sigma(\omega_{y'j}(x)) \cdot \frac{\partial \omega_{y'j}(x)}{\partial \theta}$$

where:

$$\omega_{ij}(x) = c_j + \sum_k W_{jk} x_k + U_{jj}$$

#### 2. Proof Steps

#### 1: Expression for $\log p(y|x)$

The conditional probability $p(y|x)$ in energy-based models is defined as:

$$p(y|x) = \frac{\exp(-F(x,y))}{\sum_{y'} \exp(-F(x,y'))}$$

where $F(x,y)$ is the free energy. Taking the logarithm:

$$\log p(y|x) = -F(x,y) - \log \sum_{y'} \exp(-F(x,y'))$$

#### 2: Gradient Calculation of $\log p(y|x)$

Taking the gradient with respect to $\theta$:

$$\frac{\partial \log p(y|x)}{\partial \theta} = -\frac{\partial F(x,y)}{\partial \theta} - \frac{\partial}{\partial \theta} \log \sum_{y'} \exp(-F(x,y'))$$

For the second term:

$$\frac{\partial}{\partial \theta} \log \sum_{y'} \exp(-F(x,y')) = -\sum_{y'} p(y'|x) \cdot \frac{\partial F(x,y')}{\partial \theta}$$

Question 5 Answer

**3: Substituting $F(x, y)$**
Using the given expression:

$$F(x, y) = -\sum_j \log\left(1 + \exp(\omega_{yj}(x))\right)$$

where

$$\omega_{yj}(x) = c_j + \sum_k W_{jk} x_k + U_{jj}$$

The gradient is:

$$\frac{\partial F(x, y)}{\partial \theta} = -\sum_j \sigma(\omega_{yj}(x)) \cdot \frac{\partial \omega_{yj}(x)}{\partial \theta}$$

where $\sigma(z)$ is the sigmoid function:

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

**4: Combining Results**
Substituting into the gradient of $\log p(y|x)$:

$$\frac{\partial \log p(y|x)}{\partial \theta} = \sum_j \sigma(\omega_{yj}(x)) \cdot \frac{\partial \omega_{yj}(x)}{\partial \theta} - \sum_{y'} p(y'|x) \cdot \sum_j \sigma(\omega_{y'j}(x)) \cdot \frac{\partial \omega_{y'j}(x)}{\partial \theta}$$

which matches the required result:

$$\frac{\partial \log p(y|x)}{\partial \theta} = \sum_j \sigma(\omega_{ij}(x)) \cdot \frac{\partial \omega_{ij}(x)}{\partial \theta} - \sum_{y',j} \sigma(\omega_{y'j}(x)) \cdot \frac{\partial \omega_{y'j}(x)}{\partial \theta}$$

**3. Interpretation of the Relation**
- The first term represents the gradient of the free energy for the observed state $(x, y)$. - The second term represents the expected gradient over all possible states $y'$, weighted by the probability $p(y'|x)$.

## 2.3 Theoretical Question 6

Report and compare the final learning outcome and classification accuracy for all previous models. Are the results as you expected?

Question 6 Answer

**1. RBM (Generative Model)**
- **Accuracy:** Lower than DRBM and HDRBM.
- **Training Time:** Longest, as it uses contrastive divergence (CD) for generative training, which involves Gibbs sampling and is computationally expensive.
- **Reason:** The RBM focuses on modeling the joint distribution $p(x, y)$, which is not directly optimized for classification. It may learn useful features, but these features are not guaranteed to be discriminative for the classification task.

**2. DRBM (Discriminative Model)**
- **Accuracy:** Highest among the three.
- **Training Time:** Faster than RBM, as it directly optimizes the discriminative objective $p(y|x)$, which is more aligned with the classification task.
- **Reason:** The DRBM is explicitly trained to maximize the conditional likelihood $p(y|x)$, making it more effective for classification tasks.

**3. HDRBM (Hybrid Model)**
- **Accuracy:** Slightly lower than DRBM but higher than RBM.
- **Training Time:** Longer than DRBM but shorter than RBM, as it combines both discriminative and generative objectives.
- **Reason:** The hybrid model balances the strengths of both generative and discriminative training. While it benefits from the generative component (which can improve generalization), it may not achieve the same classification accuracy as the purely discriminative DRBM.

---

**Observed Results vs. Expectations**
The observed results align well with the theoretical expectations:
- **DRBM had the highest accuracy:** This is expected because the DRBM is directly optimized for classification, making it the most effective for this task.
- **HDRBM had slightly lower accuracy than DRBM:** This is also expected, as the hybrid model trades off some discriminative power for the benefits of generative training (e.g., better generalization or robustness).
- **RBM had the lowest accuracy and longest training time:** This is consistent with expectations, as the RBM is not directly optimized for classification and relies on a more computationally expensive training process.

## 2.4 Theoretical Question 7

Can we use the unlabeled data $D_{\text{unsup}}$ in addition to the labeled data $D_{\text{sup}}$? In a precise manner, we are looking for a final objective function as follows:

$$L_{\text{semi-sup}}(D_{\text{sup}}, D_{\text{unsup}}) = L_{\text{sup}}(D_{\text{sup}}) + \beta L_{\text{unsup}}(D_{\text{unsup}})$$

Provide a suitable suggestion for $L_{\text{unsup}}$ and explain the details of its optimization.

Question 7 Answer

Yes, unlabeled data can be effectively incorporated alongside labeled data using semi-supervised learning techniques. The choice of the unsupervised loss $L_{\text{unsup}}$ significantly impacts the model's generalisation. Below are three prominent methods:

**1) Consistency Regularization:** This approach enforces the model to be invariant to perturbations of the input, ensuring robustness and better generalisation. The unsupervised loss is defined as:

$$L_{\text{unsup}} = \sum_{x} \sum_{\text{aug}} \|f(x;\theta) - f(\text{aug}(x);\theta)\|^2$$

where $f(x;\theta)$ represents the model's prediction for input $x$, and $\text{aug}(x)$ is a perturbed version of $x$ (e.g., noise injection, dropout, data augmentation).

**2) Entropy Minimization:** This technique encourages the model to make confident predictions by minimising the entropy of the predicted class distribution for unlabeled samples:

$$L_{\text{unsup}} = -\sum_{x} \sum_{c} P(y_c|x;\theta) \log P(y_c|x;\theta)$$

where $P(y_c|x;\theta)$ is the model's predicted probability for class $c$. Lower entropy means higher confidence, leading to well-separated decision boundaries.

**3) Pseudo-Labeling:** This method assigns pseudo-labels $\hat{y}$ to unlabeled data and incorporates them into supervised training:

$$L_{\text{unsup}} = \sum_{x} \sum_{c} 1(P(\hat{y}|x;\theta) > \tau) \cdot L_{\text{CE}}(\hat{y}, f(x;\theta))$$

where $\hat{y} = \arg\max_c P(y_c|x;\theta)$ is the most confident class prediction, and $\tau$ is a confidence threshold ensuring only reliable pseudo-labels contribute to training.

**Comparison and Optimization:** Among these methods, entropy minimization is often preferred due to its smooth, differentiable loss function, which resembles the cross-entropy loss used in supervised learning. It can be efficiently optimised using gradient descent techniques like SGD. The weight $\beta$ plays a crucial role in balancing supervised and unsupervised objectives. A small $\beta$ reduces the impact of unlabeled data, whereas a large $\beta$ may lead to overfitting on unreliable pseudo-labels.