Energy Based Models

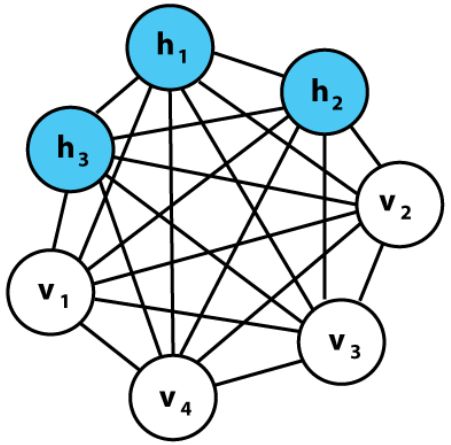
Introduction

- Energy-based models (EBMs) are a class of probabilistic models in machine learning that define probabilities using an energy function.
- These models try to minimize a predefined energy function.
  - They assign lower energy values to configurations that are more likely and higher energy values to those that are less likely.
  - Ie. High energy means bad compatibility
- EBMs are particularly useful for generative modeling, where the goal is to learn the probability distribution of the data.
- Examples of EBM models:
  - Restricted Boltzmann Machines
  - Deep Belief Networks

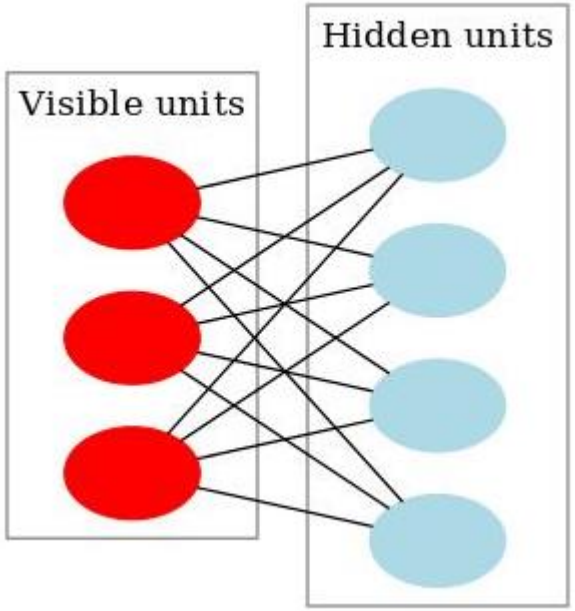
Restricted Boltzmann Machines (RBM)

- RBM is the neural network that belongs to the energy-based model
- It is a probabilistic, unsupervised, generative deep machine learning algorithm.
- RBM's objective is to find the joint probability distribution that maximizes the log-likelihood function.
- RBMs have a **bipartite** architecture with two layers:
  - **Visible layer (v)**: This represents the input data (e.g., pixels in an image).
  - **Hidden layer (h)**: This represents the learned features (latent features).
- There are:
  - **No connections between units within the same layer (visible nodes).**
  - **Full connections between visible and hidden units. (**All visible nodes are connected to all the hidden nodes**)**
- The original Boltzmann machine had connections between all the nodes. Since RBM restricts the intralayer connection, it is called a Restricted Boltzmann Machine.

Boltzmann Machine



Restricted Boltzmann Machine



Training an RBM

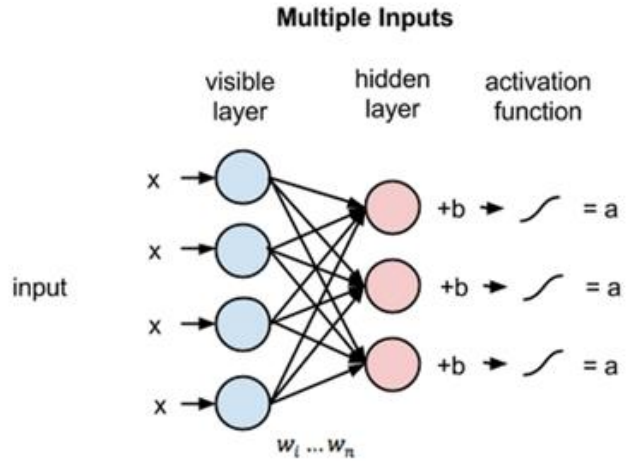
- The goal of training an RBM is to learn the weights and biases of the visible and hidden layers so that the model captures the underlying patterns in the data with the intention of recreating/ generating the data.
- It does this by adjusting parameters to minimize the difference between the actual data distribution and the model's internal (reconstructed) distribution
- The most commonly used algorithm to train RBMs is Contrastive Divergence (CD). This method approximates the gradient of the log-likelihood and updates the weights and biases to maximize the likelihood of the training data under the model.
- The training involves the following steps:
  - **Feed in real data to the visible layer — like an image or a vector.**
  - **The network activates hidden units, guessing which hidden features are likely turned on by the input.**
  - **Then, it uses those hidden features to reconstruct the input**
  - **It compares the reconstructed input with the real input using** Contrastive Divergence (CD)
  - **Based on the difference, it adjusts the weights so that it gets better next time.**

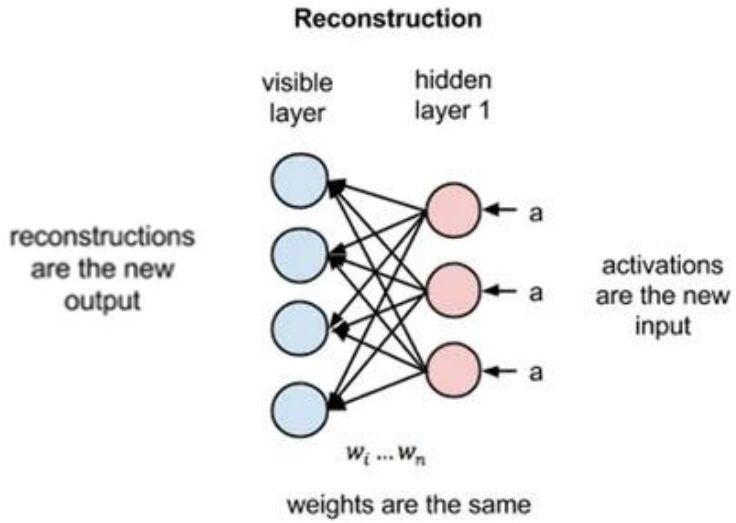
    3. **Weight update**:

    The difference between the two outer products gives the weight update:

    $$\Delta W = \epsilon(v_0 h_0^T - v_1 h_1^T)$$

    Here, ϵ is the learning rate.
- This loop is repeated over and over on many pieces of data until the RBM is good at representing the kinds it sees.





- RBMs use an **energy function** to model the joint probability of the visible and hidden layers:

$$E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i w_{ij} h_j$$

Where:

- v_i: i-th visible unit
- h_j: j-th hidden unit
- w_{ij}: weight between v_i and h_j
- a_i, b_j: biases for visible and hidden units

The training of RBM consists of the finding of parameters for given input values so that the energy reaches a minimum.
At the start of this process, weights for the visible nodes are randomly generated and used to generate the hidden nodes. These hidden nodes then use the same weights to reconstruct visible nodes. The weights used to reconstruct the visible nodes are the same throughout. However, the generated nodes are not the same because they aren't connected to each other.

Once trained, the RBM can:

- Help understand the underlying pattern in the data
- Reduce the number of features (dimensionality reduction)
- Act as a building block for deep learning models like **Deep Belief Networks**

Training an RBM: mathematical details (For those who are interested in the mathematical aspects)

Step 1: Positive Phase (Data-driven pass)

Given an input vector v (a training example):

1.1 Compute hidden probabilities:

Each hidden unit gets activated based on the input:

$$P(h_j = 1|v) = \sigma\left(b_j + \sum v_i w_{ij}\right)$$

- v: visible units (input data)
- h: hidden units (latent features)
- w_{ij}: weight between visible unit v_i and hidden unit h_j
- a_i: bias of visible unit v_i
- b_j: bias of hidden unit h_j
- $\sigma(x) = \frac{1}{1+e^{-x}}$: sigmoid function

Step 2: Negative Phase (Reconstruction pass)

Using the sampled (or probability-based) hidden units:

2.1 Reconstruct the visible units:

$$P(v'_i = 1|h) = \sigma\left(a_i + \sum_j h_j w_{ij}\right)$$

Again, you can **sample** or use the probabilities to get reconstructed visible vector v'.

2.2 Recompute hidden probabilities from reconstructed data:

$$P(h'_j = 1|v') = \sigma\left(b_j + \sum_i v'_i w_{ij}\right)$$

Step 3: Update Weights and Biases

The idea here is to **reinforce associations that the input supports**, and **penalize those that the reconstruction supports**.

3.1 Compute weight update:

$$\Delta w_{ij} = \eta \left(\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{reconstruction}} \right)$$

Where:

- η: learning rate
- $\langle \cdot \rangle$ denotes expectation (can be approximated using actual values in mini-batch)

3.2 Bias updates:

$$\Delta a_i = \eta (v_i - v_i')$$
$$\Delta b_j = \eta (h_j - h_j')$$

Then, **apply updates**:

$$w_{ij} \leftarrow w_{ij} + \Delta w_{ij}$$
$$a_i \leftarrow a_i + \Delta a_i$$
$$b_j \leftarrow b_j + \Delta b_j$$

Repeat for Each Batch and Epoch

You iterate this process over many **mini-batches** and **epochs**, gradually tuning weights to reduce the reconstruction error and improve the learned feature representations.

Deep Belief Networks DBN)

A **Deep Belief Network** is a **generative** and **deep neural network**, composed of **stacked Restricted Boltzmann Machines (RBMs)**.

- Each RBM learns to represent the input in terms of **features**. Stacking them allows DBNs to learn **hierarchical representations** — where each layer captures higher-level patterns than the one before.
- Structure of a DBN:

Input → RBM1 → RBM2 → RBM3 → ... → Classifier (e.g. Softmax)

•**RBM1**: Learns basic features from the input (e.g., edges in images)
•**RBM2**: Learns patterns from the features of RBM1
•**RBM3**: Learns even higher-order features
•**Classifier**: Final supervised fine-tuning
Each RBM is trained to model the distribution of its input.

Training DBN

🔁 1. **Pretraining (Unsupervised Layer-wise Training)**

- Train one RBM at a time, **greedily**.
- After training RBM1 on the input data, freeze its weights.
- Pass the transformed data (hidden activations) to train RBM2.
- Repeat for all layers.

Greedy training = **unsupervised pre-training of each RBM one by one**

Step-by-step:

1. **Train RBM 1** on the input data.
2. Freeze RBM 1, get hidden representations from it.
3. **Train RBM 2** on the output of RBM 1.
4. Freeze RBM 2, and repeat...
5. After stacking all layers, you can fine-tune the full network.

🏁 2. **Fine-tuning (Supervised Training)**

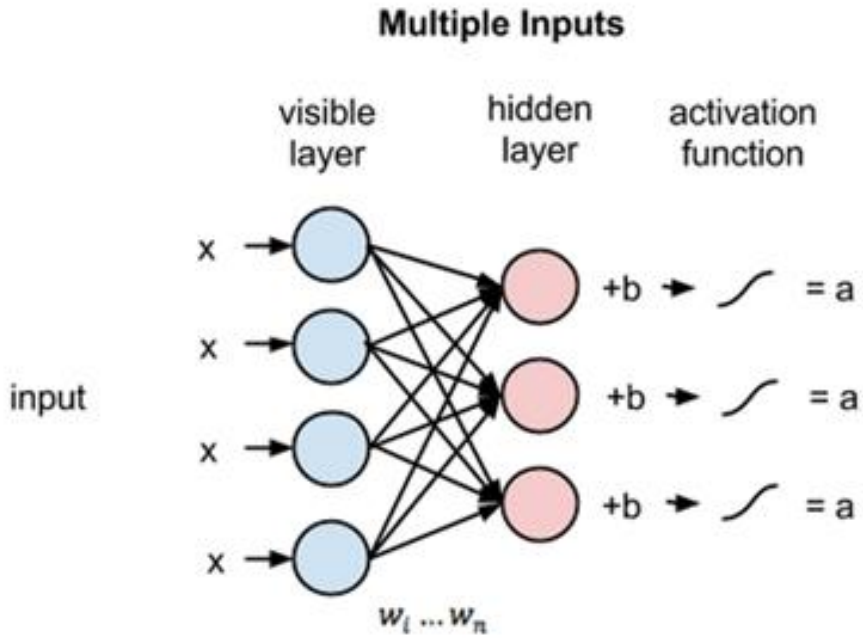
- After stacking RBMs, **add a final classifier layer** (e.g., softmax for classification).
- Train the **entire network** end-to-end using **backpropagation** on labeled data.

> During this step, all weights get fine-tuned to improve performance on the task (e.g., digit classification).

Appendix

Working of RBM

- RBM is a Stochastic Neural Network which means that each neuron will have some random behavior when activated.
- In addition to the Input layer (visible layer), and hidden layer, there are two other layers of bias units (hidden bias and visible bias) in an RBM. This is what makes RBMs different from autoencoders.
- The hidden bias RBM produces the activation on the forward pass and the visible bias helps RBM to reconstruct the input during a backward pass.
- The reconstructed input is always different from the actual input as there are no connections among the visible units and therefore, no way of transferring information among themselves.



- https://medium.com/machine-learning-researcher/boltzmann-machine-c2ce76d94da5
- Energy-based Models Definition | DeepAI