Quantum Boltzmann Machines

D-Wave implementation for Image Reconstruction

Summary

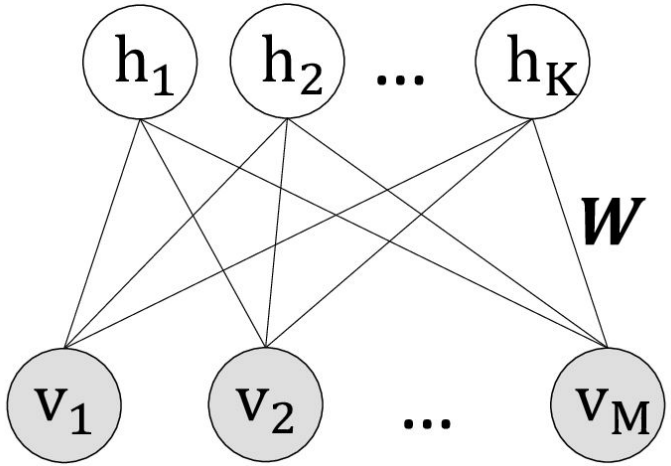
1. What is a Restricted Boltzmann Machine
2. Structure
3. Energy model
4. Partition function
5. Training procedure
6. Embedding on D-Wave
7. The specific problem

RBM is Restricted Boltzmann Machine

- Generative stochastic artificial neural network
- It can learn a probability distribution function over an input dataset
- Restricted because they can form only a bipartite graph
- Invented by Paul Smolensky in 1986, became famous thanks to Geoffrey Hinton
- Applied in:
    - Dimensionality reduction
    - Classification
    - Collaborative filtering
    - etc...

RBM structure

- Visible layer → Input
- Map through W to the Hidden Layer
- We also have biases for the visible units and for the hidden ones



Energy model

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

$$E(v,h) = -a^{\mathrm{T}} v - b^{\mathrm{T}} h - v^{\mathrm{T}} W h$$

a: weight of visible units

b: weight of hidden units

v: binary value of visible units

h: binary value of hidden units

w: weight of network edges

Partition function

- This is the intractable part of the problem
- Especially for P(v,h)

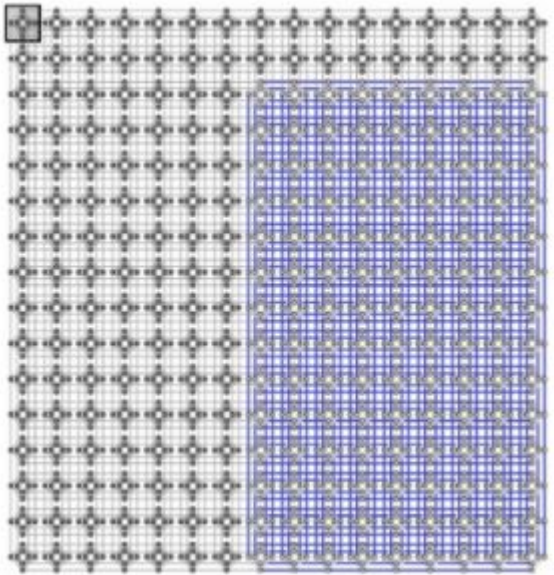
$$Z = \sum_{v,h} e^{-E(v,h)} \quad \Rightarrow \quad P(v) = \frac{1}{Z} \sum_{h} e^{-E(v,h)}$$

Training procedure

- Goal: increase the probability for a set of visible unit values to occur
- Each epoch → update RBM weights as:
  - positive_gradient = v x h
  - negative_gradient = v' x h'
  - Δw = lr x (positive_gradient - negative_gradient)
  - Δa = lr x (v - v')
  - Δb = lr x (h - h')
- Parameter → lr : learning_rate
- After training the weights we can sample the most probable visible unit values as outputs

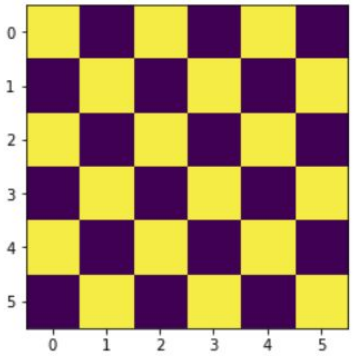
Embedding on D-Wave

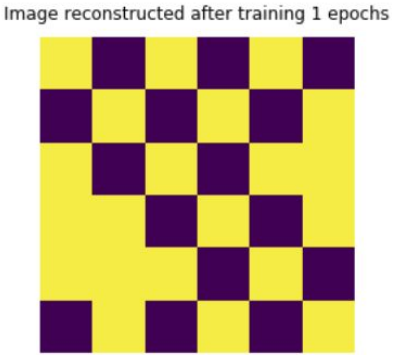
- Visible and Hidden units connected together
- Extremely dependant on the D-Wave chip size
- Suffer from noise & transverse field problems

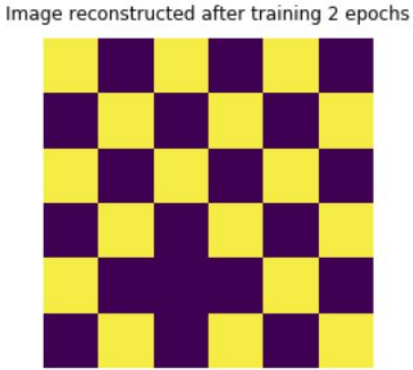


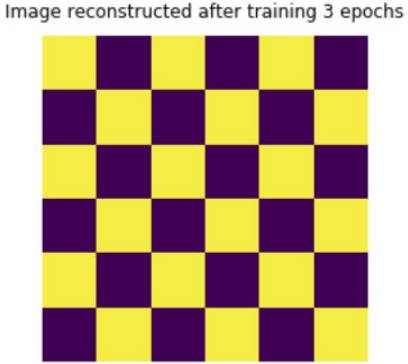
Collaborative Filtering

- Used for Recommender Systems
- Can be also used in Imaging problems
  - Image Reconstruction









The Code!

- Python → Ocean SDK
- D-Wave Quantum Annealer

References

- https://cloud.dwavesys.com/leap/
- https://arxiv.org/pdf/1606.06123.pdf
- https://arxiv.org/abs/2005.03247
- https://github.com/mercari/CFQIRBM
- https://www.nature.com/articles/s41598-021-82197-1
- https://github.com/mercari/CFQIRBM/blob/master/doc/QRBM_vs_GPU_Poster.pdf