Performance comparison on CFRBM between GPU and Quantum Annealing

Shitian Ni & Shota Nagayama

shitian.ni@mercari.com, shota.nagayama@mercari.com

Mercari, Inc.

Goal: Fast and accurate approach to train Restricted Boltzmann Machine for Collaborative Filtering

Short result: Quantum annealing faster than GPU but immature for real-world large RBMs

Background

RBM Restricted Boltzmann Machine

Structure of RBM

- RBM is a kind of network that has 2 groups, ***visible units*** and ***hidden units*** as Fig. 1
- Visible units work as both input and output as Fig. 2
- By clamping some visible unit values to be the same as input, we can sample the other unclamped units defined as predictions/output in the reconstruction phase.

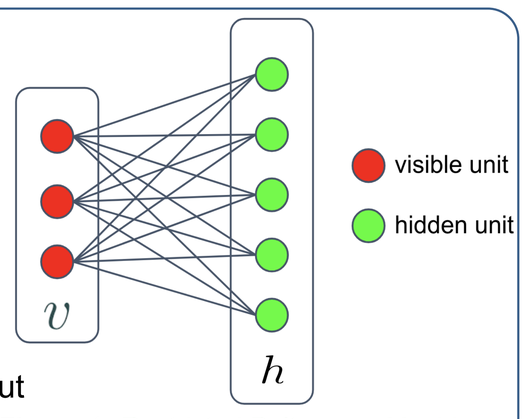


Figure 1: Restricted Boltzmann Machine

Energy defined by a set of binary values of visible units and hidden units

$$E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i w_{i,j} h_j \quad (1)$$

a : weight of visible units
v : binary value of visible units
b : weight of hidden units
h : binary value of hidden units
w : weight of network edges

Partition function Z

$$Z = \sum_{v,h} e^{-E(v,h)} \quad (2)$$

Probability for a set of binary values (v, h) to occur

$$P(v,h) = \frac{1}{Z} e^{-E(v,h)} \quad (3)$$

Training of RBM

- RBM is trained to increase the probability for a set of visible unit values to occur.
- In each epoch we update RBM weights as:
  - $positive\_gradient = v \times h$
  - $negative\_gradient = v' \times h'$
  - $\Delta w = lr \times (positive\_gradient - negative\_gradient)$
  - $\Delta a = lr \times (v - v')$
  - $\Delta b = lr \times (h - h')$

  $lr : learning\_rate$
- After the weights are trained, we can sample and get the most probable visible unit values as outputs.

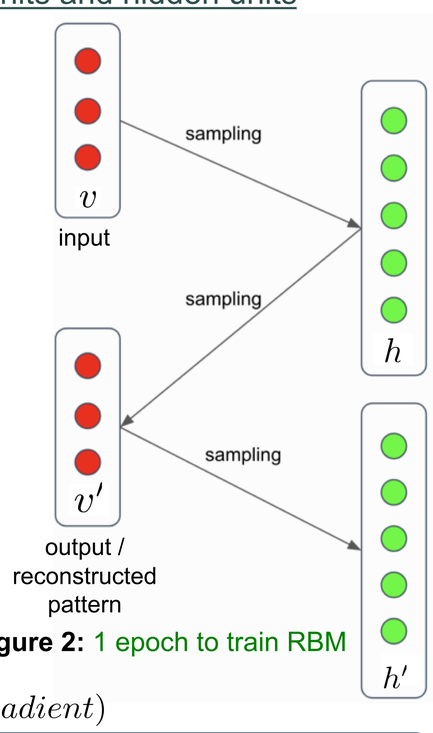


Figure 2: 1 epoch to train RBM

Quantum annealing (QA)

Find the global minimum of a objective function over a set of candidate solutions utilizing ***quantum effects***.

Background Mechanics

- Quanta have a different ground state for a different transverse field they are put in.
- *Adiabatic theorem* tells that quanta in the ground state keep in ground states though the field changes as far as slowly.

Process

1. Design a longitudinal field in which the ground state of quanta corresponds to the optimal answer of the optimization problem.
2. Start from a transverse field which has quantum in a trivial ground state, quantum superposition of all possible states.
3. Change the field, gradually making the longitudinal field dominant slowly.
4. Measure quanta to get the optimal answer.

Objective function

- Ising model variable ∈ {+1,-1}

$$H(\sigma) = -\sum_{i,j} J_{ij}\sigma_i\sigma_j - \mu \sum_j h_j \sigma_j$$
$$\sigma_k \in \{+1, -1\}$$

- Quadratic unconstrained binary optimization variable ∈ { 0, 1 }

Quantum annealing RBM

Approximate the following intractable calculations with *quantum annealing*.

- Partition function *Z* defined in (2)
- Joint probability distribution of visible and hidden units.

GPU RBM

- GPU accelerates sampling and reconstruction Matrix-Matrix Multiplications in conventional RBM implementations.
- In this work, we used TensorFlow to implement RBM instead of directly touching CUDA.

CFRBM Restricted Boltzmann Machines for Collaborative Filtering

RBM based collaborative filtering is more capable to handle large datasets and give better accuracy than traditional collaborative filtering approaches such as SVD.

Training of CFRBM

- Train sub-RBM X, Y based on item A and item B ratings from user 1.
- Train sub-RBM Y, Z based on item B and item C ratings from user 2.

Inference of CFRBM

1. Sample joint RBM X+Z based on item A rating from user 3
2. Predicted item C rating is in the lowest energy sample whose item A rating == rating from user 3

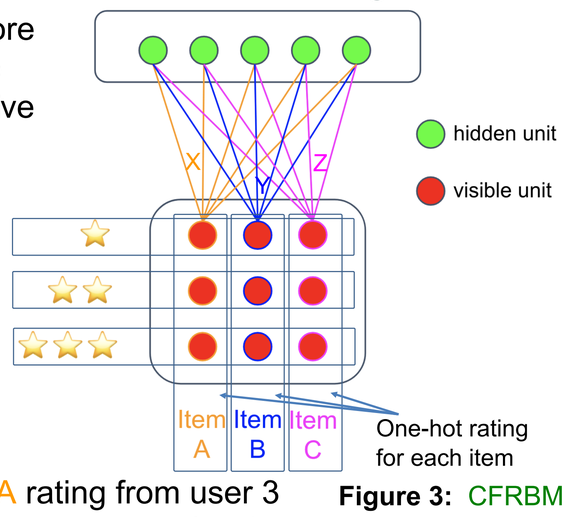


Figure 3: CFRBM

Performance Comparisons

Training input v		
	TensorFlow	**QA**
Default		
Epoch 1		
Epoch 2		
Epoch 3		

Table 1: v' reconstructed from training iterations

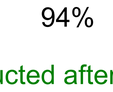
Test random input v		
	TensorFlow	**QA**
Test 1		
Test 2		
Test 3		
Accuracy	100%	94%

Table 2: v' reconstructed after training

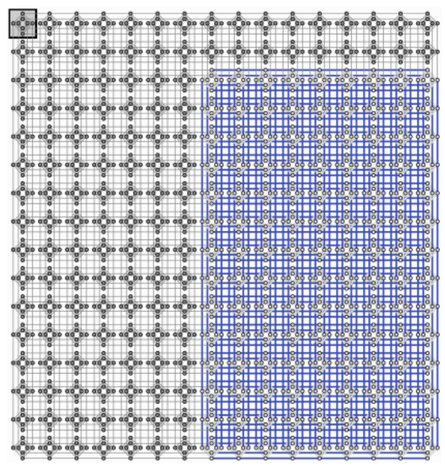


Figure 4: Emdedding of a 36x56 RBM on a D-Wave 2000Q chip powered by D-Wave web interface

Our programs

https://github.com/mercari/CFQIRBM
https://github.com/shitian-ni/tensorflow-rbm
https://github.com/shitian-ni/tensorflow-rbm/blob/master/measure_time.py

Measure

- Calculation time of RBM, to sample and reconstruct visible and hidden units.
- This measure of RBM should be proportional to that of CFRBM because CFRBM is composed from RBMs.

Approaches to Train RBM

- Quantum Annealing
- Bernoulli sampling in TensorFlow with GPU
- Bernoulli sampling in TensorFlow with CPU

Test Data

6x6 lattice pattern.

Result

On AWS p3.2xlarge instance

TensorFlow_gpu v1.10.1 Tesla V100	≈ 0.005 sec
TensorFow v1.10.1 Intel Xeon E5 (8 cores)	≈ 0.004 sec
Quantum Annealing D-Wave 2000Q	0.000164 sec (164 μs)

Table 3: Speed comparison on elapsed time for a 36x56 RBM to perform one sampling

Noticeable time command outputs

The parallelism is not clear because of the too small size of the problem.

Percent of CPU	114%
User time (sec)	2.22
System time (sec)	1.23
Elapsed time (sec)	3.01

GPU sampling

Percent of CPU	136%
User time (sec)	1.83
System time (sec)	0.70
Elapsed time (sec)	1.86

CPU sampling

Considerations

- Data transfer overhead causing GPU time longer than CPU in this small sized RBM.
- QA needs more iterations to train and is less accurate because of its probabilistic nature to give the optimal answer.
- Sampling by QA is the fastest in small RBMs.
- Chimera graph topology of QA chip by D-wave is sufficient for RBM.

Conclusions

- If we ignore access and data transfer overheads suffered by both QA and GPU, QA is the fastest approach to use RBM.
- However, current QA hardware by D-Wave can only hold at most 2048 qubits so QA RBM applications are extremely limited.
- Larger RBMs could show GPU advantage over CPU. So we conclude that GPU based RBM is still the best approach to solve real world problems.

Future Works

- RBMs with narrower but deeper hidden unit layers
- Use more complex datasets to compare KL-Divergence differences
- Try reverse annealing to enhance the performance

References

1. Tijmen Tieleman, "Training Restricted Boltzmann Machines using Approximations to the Likelihood Gradient", ICML, 2008.
2. Ali Ghodsi, "Deep Learning - Restricted Boltzmann Machines (RBM)", 2015.
3. Marcello Benedetti, John Realpe-Gomez, Rupak Biswas, Alejandro Perdomo-Ortiz, "Quantum-Assisted Learning of Hardware-Embedded Probabilistic Graphical Models", arXiv:1609.02542, 2016.
4. Steven H. Adachi, Maxwell P. Henderson, "Application of Quantum Annealing to Training of Deep Neural Networks", arXiv:1510.06356, 2015.
5. Mohammad H. Amin, Evgeny Andriyash, Jason Rolfe, Bohdan Kulchytskyy, Roger Melko, "Quantum Boltzmann Machine". arXiv:1601.02036, 2016.
6. Cai, Xianggao, Xu, Zhanpeng, Lai, Guoming, Wu, Chengwei, Lin, Xiaola, "GPU-Accelerated Restricted Boltzmann Machine for Collaborative Filtering", 2012.
7. Ruslan Salakhutdinov, Andriy Mnih, Geoffrey Hinton, "Restricted Boltzmann Machines for Collaborative Filtering", ICML, 2007.