Quantum Boltzmann Machines

Maria Kieferova [1,2]   Nathan Wiebe[3]

[1] IQC, University of Waterloo

[2]Macquarie University

[3]Microsoft Research

Can we encode complicated models into a relatively small neural network?

- quantum algorithm
- exploiting the full power of a quantum computer
- can handle quantum input/output

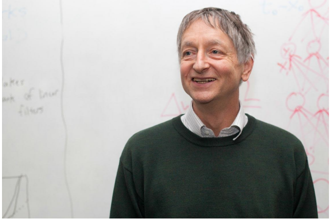
Can we encode complicated models into a relatively small neural network?

- quantum algorithm
- exploiting the full power of a quantum computer
- can handle quantum input/output

Boltzmann machines are a good candidate for quantizations [M. H. Amin, et al., arXiv:1601.02036]

What is a quantum Boltzmann machine?

Boltzmann machines are neural networks used for generative machine learning. [G. E. Hinton, 2012]

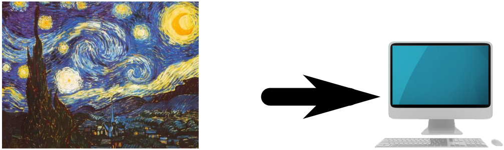


Generative Training

"What I cannot create,
I do not understand."

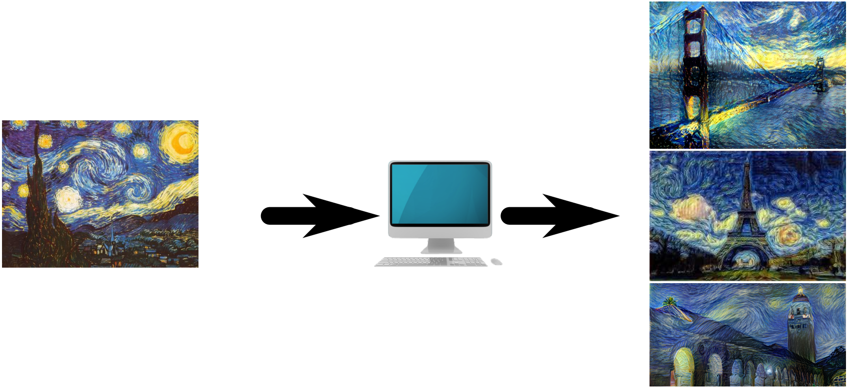
—Richard Feynman

Generative Training



[https://github.com/jcjohnson/neural-style]

Generative Training

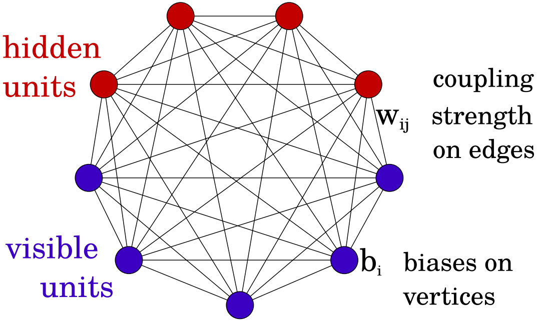


[https://github.com/jcjohnson/neural-style]

Algorithm learns a model that explains how the data were generated.

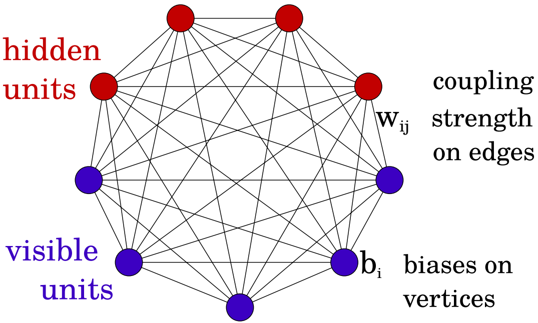
- Assume there is an underlaying distribution of the data parametrized by $\{\Theta_i\}$
- Find $\{\Theta_i\}$ that "explains" the data
- Generate similar examples using the model

The Neural Network



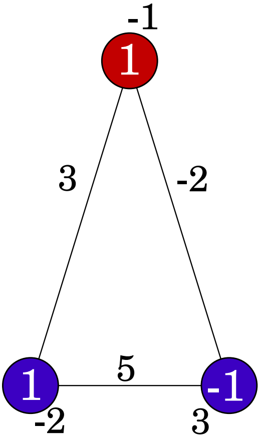
visible units serve as input/output
hidden units provide extra degrees of freedom

The Neural Network



weights and biases are learned during training

Boltzmann Machine



Units s_i can take values -1 and 1

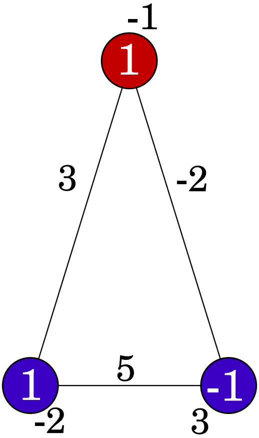
Energy of a configuration

$$E(v,h) = \sum_{\text{vertices } i} b_i s_i + \sum_{\text{edges } <i,j>} w_{i,j} s_i s_j$$

Motivated by the Ising model

$$\begin{aligned} E(v,h) &= (-1)\cdot 1 + (-2)\cdot 1 + 3\cdot(-1) + \\ &+ 3\cdot 1\cdot 1 + (-2)\cdot 1\cdot(-1) + 5\cdot 1\cdot(-1) \\ &= -6 \end{aligned}$$

Probability of a configuration on visible (v) and hidden (h) units

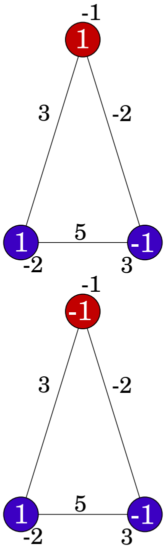


$$p(v, h) = \frac{1}{Z} e^{-E(v,h)}$$

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

Boltzmann distribution
favors low-energy states

Probability of a configuration on visible (v) units is the marginal distribution



$$p(v) = \sum_h p(v,h) = \sum_h \frac{1}{Z} e^{-E(v,h)}$$

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

The distribution $p(v)$ should be close to the distribution over the data $q(v)$ for a properly trained Boltzmann machine.

"Distance" Measure

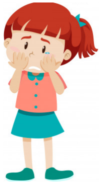
The distribution $p(v) = \sum_h p(v,h)$ should be **close** to the distribution over the data $q(v)$ for a properly trained Boltzmann machine

KL divergence

$$\mathcal{L}_{\mathcal{KL}} = \sum_{v \in data} q(v) \left[\log\Big(q(v)\Big) - \log\Big(\sum_h p(v,h)\Big) \right]$$

Negative log-likelihood

$$\mathcal{L} = -\sum_{v \in data} q(v) \log\Big(\sum_h p(v,h)\Big)$$



$\mathcal{L}$ is difficult to compute

Minimize KL divergence using gradient descent

Gradient of $\mathcal{L}$ is easy* to compute
Requires only expectation values of single vertices or edges

$$\frac{\partial \log p(v)}{\partial w_{i,j}} = \langle s_i s_j \rangle_{data} - \langle s_i s_j \rangle_{model}$$

$$\frac{\partial \log p(v)}{\partial b_i} = \langle s_i \rangle_{data} - \langle s_i \rangle_{model}$$

*Given an approximation of a thermal state

Boltzmann Machine Training: The Algorithm

1. decide on the graph
2. generate starting weights and biases
3. construct the energy function
4. **repeat** for number of epochs:

Boltzmann Machine Training: The Algorithm

1. decide on the graph
2. generate starting weights and biases
3. construct the energy function
4. **repeat** for number of epochs:
    4.1 measure expectation values in the thermal state

Boltzmann Machine Training: The Algorithm

1. decide on the graph
2. generate starting weights and biases
3. construct the energy function
4. **repeat** for number of epochs:
   - 4.1 measure expectation values in the thermal state
   - 4.2 **repeat** for each training example
     - 4.2.1 set visible units to the example vector
     - 4.2.2 create a thermal state on hidden units
     - 4.2.3 measure expectation values

Boltzmann Machine Training: The Algorithm

1. decide on the graph
2. generate starting weights and biases
3. construct the energy function
4. **repeat** for number of epochs:
   - 4.1 measure expectation values in the thermal state
   - 4.2 **repeat** for each training example
     - 4.2.1 set visible units to the example vector
     - 4.2.2 create a thermal state on hidden units
     - 4.2.3 measure expectation values
   - 4.3 compute gradients
   - 4.4 update weights and biases
   - 4.5 construct the new energy function

Quantum Boltzmann Machine

Quantum version of:

Energy function

Objective function

Training data

Energy Function

Classical thermal distribution:

$$p = \frac{1}{Z} e^{-E}, \quad Z = \sum_z e^{-E}$$

Quantum thermal state:

$$\rho = \frac{1}{Z} e^{-H}, \quad Z = Tr\left[e^{-H}\right]$$

Hamiltonian for Ising model

$$H = \sum_i b_i \sigma_i^z + \sum_{i<j} w_{i,j} \sigma_i^z \sigma_j^z$$

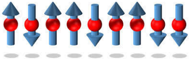
energy $H\left|\psi\right\rangle = E\left|\psi\right\rangle$

Hamiltonian

Arbitrary QBM Hamiltonian

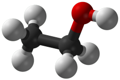
$$H = \sum_i \theta_i H_i, \qquad \|H_i\| = 1$$

Transverse Ising Model

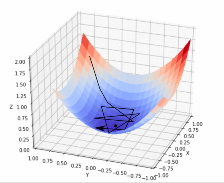


Pauli Basis

Fermionic Hamiltonian



Objective Function



KL divergence

$$\mathcal{L}_{\mathcal{KL}} = \sum_{v \in data} q(v) \left[\log \Big(q(v) \Big) - \log \Big(\sum_{h} p(v,h) \Big) \right]$$

Quantum Relative Entropy

$$S = \mathrm{Tr}\left[\rho_{data} \left(\log \rho_{data} - \log \sigma_{model} \right) \right]$$

Objective Function

Minimize the objective function

$$\mathcal{O}_\rho = -\mathrm{Tr}\left[\rho_{data} \log \rho_{model}\right]$$

For no hidden units:

$$\partial_\theta \mathcal{O}_\rho = \langle \partial_\theta H \rangle_{qbm} - \langle \partial_\theta H \rangle_{data}$$

Learning Quantum States

$$\langle \partial_\theta H \rangle_{model} = \mathrm{Tr}\left[\partial_\theta H \frac{e^{-H}}{Tr[e^{-H}]}\right]$$

estimated by sampling from the QBM

$$\langle \partial_\theta H \rangle_{data} = Tr[\partial_\theta H \rho_{data}]$$

created by simulation

Training data is a density matrix

Creating a representation of a quantum state is the goal of tomography

Tomography

unknown state

ρ

local measurements

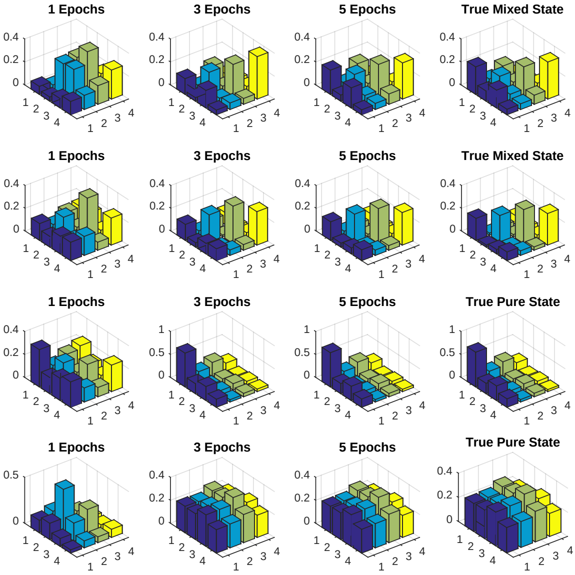
classical description

$\tilde{\rho}$

Tomography

- local measurements give gradients of relative entropy
- QBM describes the unknown state as a thermal state of a Hamiltonian
- Hamiltonian gives a classical approximate description of a state
- bonus: QBM works as an approximate cloning device

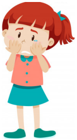
Relative Entropy Training



Relative Entropy Training

- low number of epochs
- “compact” models for quantum states
- general method but can be tailored
- ability to create copies of a quantum state

Relative Entropy Training



- low number of epochs
- "compact" models for quantum states
- general method but can be tailored
- ability to create copies of a quantum state

- each epoch requires estimation of several expectations values
- approximating a thermal state can be costly

Relative Entropy Training

- learn classical or quantum data
- gradient descent algorithm
- unorthodox approach to tomography
- approximate cloning

Objective Function - Part Deux

log-likelihood

$$\mathcal{L} = \sum_{v \in data} q(v) \log \left(\frac{\sum_h e^{-E(v,h)}}{\sum_{v',h'} e^{-E(v',h')}} \right)$$

quantum log-likelihood

$$\mathcal{O}_\Lambda(H) = \sum_{\mathbf{v}} P_{\mathbf{v}} \log \left(\frac{\mathrm{Tr}\left[\Lambda_v e^{-H}\right]}{\mathrm{Tr}\left[e^{-H}\right]} \right)$$

where Λ_v is a projector on state v on visible units

[M. H. Amin, et al., arXiv:1601.02036]

Golden-Thompson Training

The gradient cannot be directly computed but can be bounded using Golden-Thomson inequality

$$\mathcal{L} \geq \sum_v P_v \log \left(\frac{Tr[\Lambda_v e^{-H_v}]}{Tr[e^{-H}]} \right)$$

"clamped" Hamiltonian $H_v = H - ln\Lambda_v$
cannot learn states where $\mathrm{Tr}[e^{-H_v}\partial_\theta H] = 0$

[M. H. Amin, et al., arXiv:1601.02036]

How to extend the training set beyond classical states?

The set $\{\Lambda_v\}$ must be POVM

The choice for classical data is ambiguous

Example: Training Set

- classical

$$\Lambda_n = |2n\rangle\langle 2n| \text{ for } 1 \leq n \leq 8$$

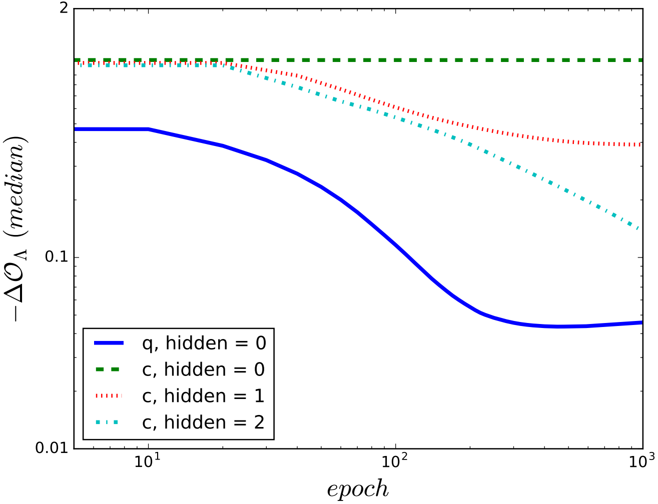
$$\Lambda_0 = \mathbb{1} - \sum_{n=1}^{8} \Lambda_n, \qquad P_v = (1 - \delta_{v,0})/8.$$

- superposition

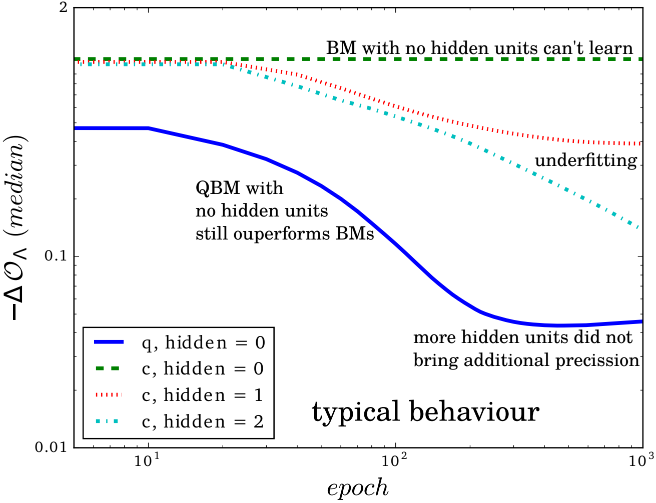
$$\Lambda_1 = \frac{1}{8}\big(\,|2\rangle + \cdots + |16\rangle\,\big)\big(\,\langle 2| + \cdots + \langle 16|\,\big),$$

$$\Lambda_0 = \mathbb{1} - \Lambda_1, \qquad P_v = \delta_{v,1}.$$

5 visible units



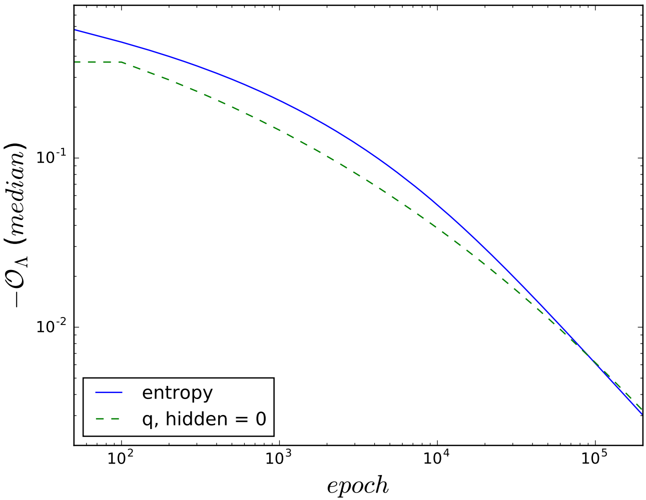
5 visible units



Golden-Thompson Training

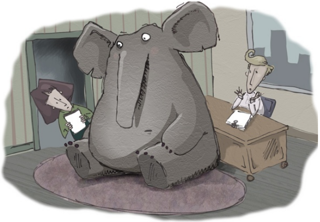
- natural generalization of log-likelihood
- bound on the gradient can provide high-precision results
- compact models of data

Quantum Relative Entropy vs Golden-Thompson Training



The thermal state is difficult to compute

Boltzmann machines face the same limitations but work well with a weak approximation of the thermal state (contrastive divergence for RBM)



Thermal State

In practice we do not compute expectations values in the thermal state

- finite number of samples n
- approximation of the thermal state error ϵ_H

Error in estimating each component of the gradient

$$\mathcal{O}\left(\sqrt{1/n + \epsilon_H^2}\right)$$

Existing algorithms:
[A. N. Chowdhury and R. D. Somma, arXiv:1603.02940]
[M. Yung and A. Aspuru-Guzik, Proc. Natl. Acad. Sci]

Complexity

QBM is BQP hard

- can’t be simulated efficiently unless BPP=BQP
- likely to provide quantum advantage

To Do:

- larger scale needed to show performance in practice
- unclear how approximations affect the convergence in practice

Summary

- Potential application for medium size quantum computers
- Richer models, better approximation than classical Boltzmann machines
- A new type of tomography - thermal state representation

Summary

- Potential application for medium size quantum computers
- Richer models, better approximation than classical Boltzmann machines
- A new type of tomography - thermal state representation

Thank you!