Restricted Boltzmann Machines

Patrick Huembeli

Build an RBM

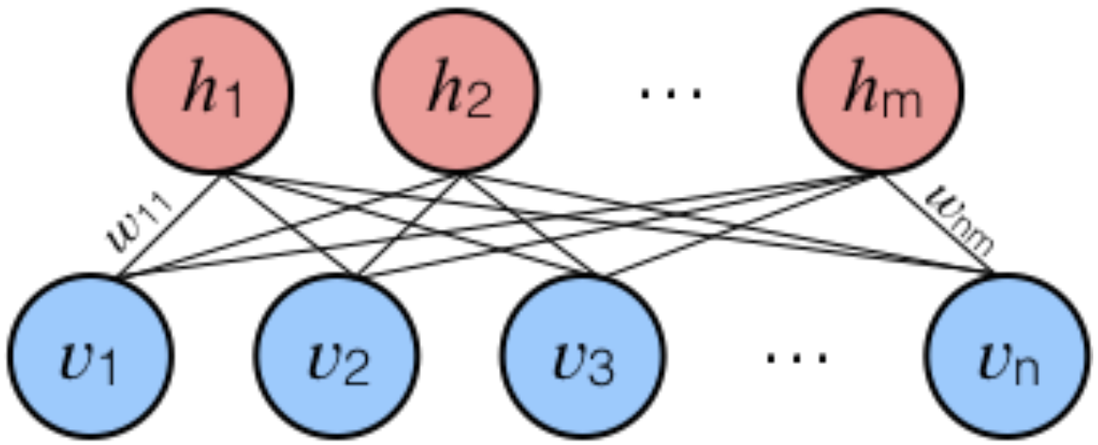
Ingredients:

- Bipartite graph: hidden and visible
- Binary units $\{0,1\}$ **or** $\{-1,1\}$
- Probabilistic model
  - Normal Neural Networks are deterministic
  - We want to find e.g.

$$p(v_i = 1)$$

$$p(h_i = 0)$$

https://medium.com/@MeTroFuN/python-mxnet-tutorial-1-restricted-boltzmann-machines-using-ndarray-f77578648ecf



Parametrise weights and biases with:

$$\boldsymbol{\lambda} = (\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c})$$

Build an RBM

$$E_{\lambda}(\boldsymbol{v},\boldsymbol{h}) = -\sum_{i,j} w_{ij} v_i h_j - \sum_i b_i v_i - \sum_j c_j h_j$$

Boltzmann distribution

$$p_{\lambda}(\boldsymbol{v},\boldsymbol{h}) = e^{-E_{\lambda}(\boldsymbol{v},\boldsymbol{h})}/Z$$

Partition Function

$$Z = \sum_{\boldsymbol{v},\boldsymbol{h}} p_{\lambda}(\boldsymbol{v},\boldsymbol{h})$$

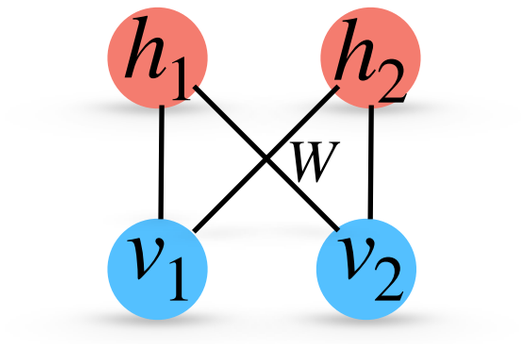
$$\rightarrow p_{\lambda}(\boldsymbol{v}) = \sum_{\boldsymbol{h}} p_{\lambda}(\boldsymbol{v},\boldsymbol{h}) = e^{-\mathscr{E}(\boldsymbol{v})}/Z$$

$$\mathscr{E}(v) = -\sum_i b_i v_i - \sum_j softplus(c_j + \sum_i w_{i,j} v_i)$$

$$softplus(x) = \ln(1 + \exp(x))$$

Min. Energy gives max. Probability !!!

Example:



$$b_i = c_i = 0$$

$$w_{i,j} = 1,\ \forall i, j$$

$$E = -v_1 h_1 - v_1 h_2 - v_2 h_1 - v_2 h_2$$

Which configuration is most likely??

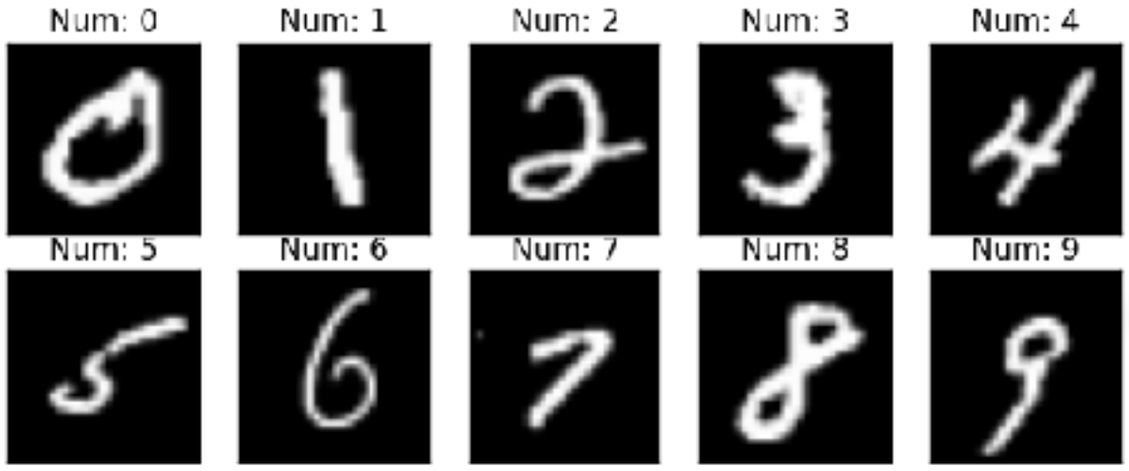
Homework:

Calculate Probabilities for this case for all configurations.
To do so you will have to calculate the partition function and the energies for all possible configurations.
Do the same with $w_{i,j} = -1,\ \forall i, j$

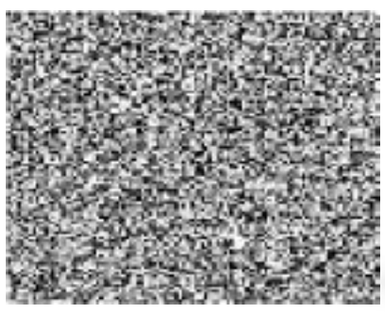
Why is this useful?

- RBM can learn probability distributions $p_{\lambda}(v)$ by finding the parameters that:

$v =$



Make this configuration very likely



Make this configuration very unlikely

Make this configuration quite likely

- Learning is unsupervised, no labels needed

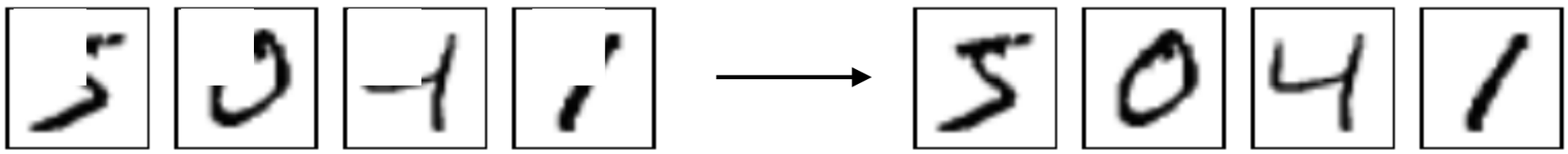
Why is this useful?

- We can sample from the learned distribution via Gibbs sampling

- Get new configurations

- Generate new data

- Restore ‘defect’ data (Homework)



1st Summary

- RBMs are probabilistic models
- We here only look at binary RBMs, which means all nodes are either 0 or 1
- We define energy and adjust parameters such that the configurations of our data have the smallest energy
    - Which is the highest probability
    - This is inspired by statistical physics (e.g. classical Spins)
    - Preferred configurations have lowest energy
- Think of training data as probability distribution.
    - Distribution of pixels that we want to learn

Build an RBM

Probabilities:

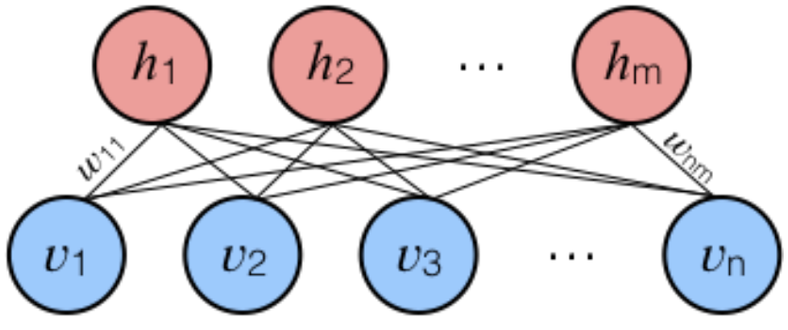
$$p_\lambda(\boldsymbol{v}, \boldsymbol{h}) = p_\lambda(\boldsymbol{v} \mid \boldsymbol{h}) p_\lambda(\boldsymbol{h})$$

$$p_\lambda(\boldsymbol{h}, \boldsymbol{v}) = p_\lambda(\boldsymbol{h} \mid \boldsymbol{v}) p_\lambda(\boldsymbol{v})$$

Because there are no connections between the units of the same layer, the conditional distributions factorise:

$$p_\lambda(\boldsymbol{v} \mid \boldsymbol{h}) = \prod_i p_\lambda(\boldsymbol{v}_i \mid \boldsymbol{h})$$

$$p_\lambda(\boldsymbol{h} \mid \boldsymbol{v}) = \prod_j p_\lambda(\boldsymbol{h}_j \mid \boldsymbol{v})$$



Build an RBM

- Calculate conditional probabilities

$$p_{\boldsymbol{\lambda}}(\boldsymbol{v}_i = 1 \,|\, \boldsymbol{h}) = \mathcal{S}\left(b_i + \sum_j h_j w_{i,j} \right)$$

$$p_{\lambda}(\boldsymbol{h}_i = 1 \,|\, \boldsymbol{v}) = \mathcal{S}\left(c_j + \sum_i v_j w_{i,j} \right)$$

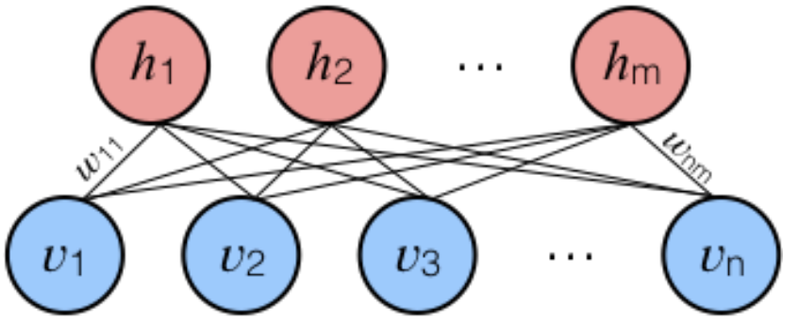
$$\mathcal{S}(x) = \frac{1}{1 + e^{-x}}$$

Derivation:

http://www.iro.umontreal.ca/~bengioy/ift6266/H14/ftml-sec5.pdf

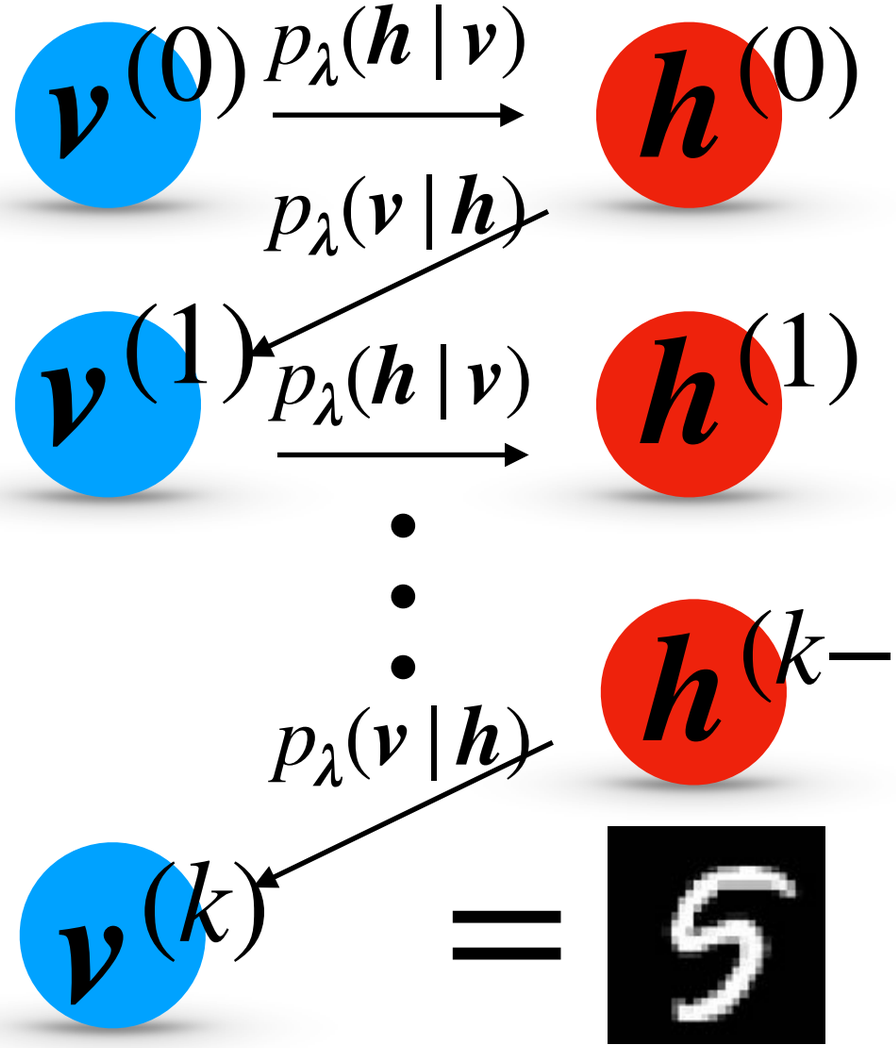
Build an RBM: Example

- After RBM is trained on MNIST, the configurations of MNIST are most likely!



Input v is given

$v^{(0)}$



Markov chain converges to stationary solution!
Which means low energy solution

Train an RBM

‘Cost function’:

$$C_\lambda = D_{KL}(q||p_\lambda) = \sum_{\boldsymbol{v}} q(\boldsymbol{v}) \log\left(\frac{q(\boldsymbol{v})}{p_\lambda(\boldsymbol{v})}\right)$$

q is the data distribution, for example MNIST. How are the pixels normally arranged in a MNIST data set? What is the probability $q(v)$ of a configuration $\mathcal{V}$ in your data set?

p_λ is the distribution learned by the RBM. The parameters λ determine the probability of a certain configuration $\mathcal{V}$.

Train an RBM

- Minimize Kullback-Leibler (KL) divergence
- To calculate gradient of KL-Divergence we need to calculate expectation values over the model distribution.

Problem!

$$\nabla_\lambda C_\lambda \approx = \langle \nabla_\lambda \mathscr{E}_\lambda(\boldsymbol{v}) \rangle_{\mathscr{D}} - \langle \nabla_\lambda \mathscr{E}_\lambda(\boldsymbol{v}) \rangle_{p_\lambda}$$

- Partition function Z generally not accessible

$$p_\lambda(\boldsymbol{v}) = e^{-\mathscr{E}(\boldsymbol{v})}/Z$$

- Derivation from : https://qucumber.readthedocs.io/en/stable/_static/RBM_tutorial.pdf

Train RBM

- Derivation on https://github.com/PatrickHuembeli/QML-Course-UPC-2018/blob/master/RBM_Slides_and_Notes/RBM_gradient_derivation.pdf

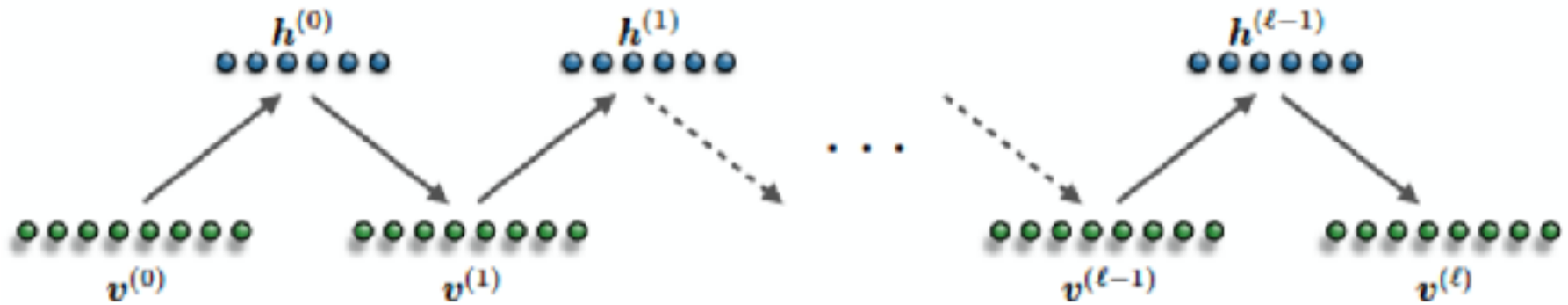
Train RBM

- The problem is we cannot calculate $p_\lambda(\boldsymbol{v})$
- Because to calculate $Z = \sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h})$ sum grows exponentially
- Approximate expectation value with estimator sampled from model using Gibbs sampling

$$\langle \nabla_{\boldsymbol{\lambda}} \mathscr{E}_{\boldsymbol{\lambda}}(\boldsymbol{v}) \rangle_{p_\lambda} = \sum_{\boldsymbol{v}} p_{\boldsymbol{\lambda}}(\boldsymbol{v}) \mathscr{E}_{\boldsymbol{\lambda}}(\boldsymbol{v}) \approx \frac{1}{M} \sum_{\boldsymbol{v}^{(l)} \in \mathscr{M}} \mathscr{E}_{\boldsymbol{\lambda}}(\boldsymbol{v}^{(l)})$$

Gibbs sampling?

- Ideally Markov chain converges to stationary solution



- If we sample back and forth, we approximate the distribution of the RBM

- Initial vector is a training sample (Homework)

- Empirics show, that one Gibbs step is enough

Contrastive divergence

- Instead of taking the model distribution $p_\lambda(\boldsymbol{v})$
- We calculate the expectation value with an estimator

$$\langle \nabla_{\boldsymbol{\lambda}} \mathscr{E}_{\boldsymbol{\lambda}}(\boldsymbol{v}) \rangle_{p_\lambda} = \sum_{\boldsymbol{v}} p_{\boldsymbol{\lambda}}(\boldsymbol{v}) \mathscr{E}_{\boldsymbol{\lambda}}(\boldsymbol{v}) \approx \frac{1}{M} \sum_{\boldsymbol{v}^{(k)} \in \mathscr{M}} \mathscr{E}_{\boldsymbol{\lambda}}(\boldsymbol{v}^{(k)})$$

- The number of Gibbs steps we take is k
- The more steps we take the closer we converge the the actual model distribution
- But it also takes more time
- Contrastive divergence is often referred to as CD-k, where k gives the number of steps

Calculate gradients explicitly

$$\frac{\partial \mathscr{E}_{\boldsymbol{\lambda}}(\boldsymbol{v})}{\partial w_{i,j}} = -\, v_i \cdot p_{\boldsymbol{\lambda}}(h_j = 1 \,|\, \boldsymbol{v})$$

$$\frac{\partial \mathscr{E}_{\boldsymbol{\lambda}}(\boldsymbol{v})}{\partial c_j} = -\, p_{\boldsymbol{\lambda}}(h_j = 1 \,|\, \boldsymbol{v})$$

$$\frac{\partial \mathscr{E}_{\boldsymbol{\lambda}}(\boldsymbol{v})}{\partial b_i} = -\, v_i$$

Check as Homework!

Parameter update

- Like in stochastic gradient descent $\lambda \leftarrow \lambda - \eta \nabla_\lambda C_\lambda$
- Reminder: $\nabla_\lambda C_\lambda \approx = \langle \nabla_\lambda \mathscr{E}_\lambda(\boldsymbol{v}) \rangle_{\mathscr{D}} - \langle \nabla_\lambda \mathscr{E}_\lambda(\boldsymbol{v}) \rangle_{p_\lambda}$
- E.g. weight update

$$w_{i,j} \leftarrow w_{i,j} - \eta \langle \frac{\partial \mathscr{E}_\lambda(\boldsymbol{v})}{\partial w_{i,j}} \rangle_{\mathscr{D}} + \eta \langle \frac{\partial \mathscr{E}_\lambda(\boldsymbol{v})}{\partial w_{i,j}} \rangle_{p_\lambda}$$

$$w_{i,j} \leftarrow w_{i,j} + \eta \langle v_i \cdot p_\lambda(h_j = 1 \,|\, \boldsymbol{v}) \rangle_{\mathscr{D}} - \eta \langle v_i \cdot p_\lambda(h_j = 1 \,|\, \boldsymbol{v}) \rangle_{p_\lambda}$$

$$w_{i,j} \leftarrow w_{i,j} + \eta \frac{1}{|\,\mathscr{D}\,|} \sum_{v_i \in \mathscr{D}} v_i \cdot p_\lambda(h_j = 1 \,|\, \boldsymbol{v}) - \eta \frac{1}{|\,\mathscr{M}\,|} \sum_{v_i^{(k)} \in \mathscr{M}} v_i \cdot p_\lambda(h_j = 1 \,|\, \boldsymbol{v})$$

Sum over batch from data

Sum over batch of Gibbs samples

Restricted Boltzmann Machines in Physics

RBM as ansatz for wavefunctions

$$\Psi = \sum_{\sigma} \Psi(\sigma)\,|\sigma\rangle$$

$$P(\sigma) \propto \left|\Psi(\sigma)\right|^2$$

$$\Psi(\sigma) = \langle \sigma | \Psi \rangle$$

$$\psi_\lambda(\sigma) = \sqrt{\frac{p_\lambda(\sigma)}{Z_\lambda}}$$

- Variational Monte Carlo and minimize e.g. energy (Troyer et al.)

- Learn state from samples / measurements

- Ansatz is efficient.

  ➡Grows polynomial in system size

RBM as ansatz for wavefunctions

Wavefunction of physical system $\Psi(\sigma) = \langle \sigma | \Psi \rangle$

Take RBM ansatz

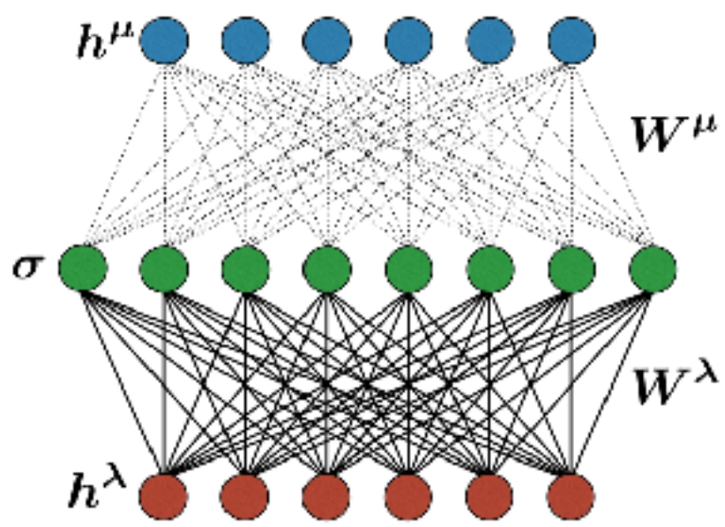
Real-positive wavefunction

$$\psi_\lambda(\sigma) = \sqrt{\frac{p_\lambda(\sigma)}{Z_\lambda}}$$

Complex wavefunction

$$\psi_{\lambda,\mu}(\sigma) = \sqrt{\frac{p_\lambda(\sigma)}{Z_\lambda}} e^{i\phi_\mu(\sigma)}$$

$$\phi_\mu(\sigma) = \log p_\mu(\sigma)$$



State tomography

- For phases we need to change basis.

- Make measurements on system in different basis
  b = 0, 1, …

$$P_b(\boldsymbol{\sigma}^{[b]}) \propto \left|\Psi(\boldsymbol{\sigma}^{[b]})\right|^2$$

- Learn this distribution with RBM

$$\psi_{\boldsymbol{\lambda},\boldsymbol{\mu}}(\boldsymbol{\sigma}^{[b]}) = \sum_{\boldsymbol{\sigma}} U_b(\boldsymbol{\sigma}, \boldsymbol{\sigma}^{[b]}) \psi_{\boldsymbol{\lambda},\boldsymbol{\mu}}(\boldsymbol{\sigma})$$

- In contrast to positive-real wavefunction, the unitaries will be part of the training.

Results

- RBMs are efficient representation for quantum states
- We can sample from it
- We can calculate any expectation value via estimators
- Open source software for QST

➡https://github.com/PIQuIL/QuCumber