RG inspired Machine Learning for lattice field theory

Yannick Meurice

The University of Iowa

yannick-meurice@uiowa.edu

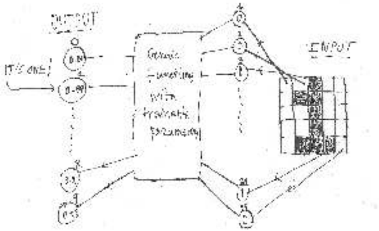
With Sam Foreman, Joel Giedt, and Judah Unmuth-Yockey

Lattice 2017, June 23

Content of the talk

1. What is Machine Learning?
2. RG ideas applied to digit recognition using the MNIST data
   - The MNIST data
   - The multiple layer perceptron
   - Principal Components Analysis (PCA): relevant features
   - Coarse graining procedures
3. The 2D Ising model near T_c (Sam Foreman)
4. Work in progress (Ising near the Tensor RG fixed point; Restricted Boltzmann Machines)
5. Conclusions

Rosenblatt perceptron (1958) and its digital version



Using outputs functions of the form $y_l = \sigma(\sum_j W_{lj} v_j)$ you can recognize 91 percent of the digits of the MNIST data (v_j: pixels, W_{lj} tunable, $\sigma(x)$ the sigmoid function).

Machine Learning Course (Andrew Ng, Stanford)

About this course: Machine learning is the science of getting computers to act without being explicitly programmed.

In the past decade, machine learning has given us self-driving cars, practical speech recognition, effective web search, and a vastly improved understanding of the human genome.

Topics include:

(i) Supervised learning (parametric/non-parametric algorithms, support vector machines, kernels, neural networks).

(ii) Unsupervised learning (clustering, dimensionality reduction, recommender systems, deep learning).

(iii) Best practices in machine learning (bias/variance theory; innovation process in machine learning and AI). [...]

MNIST data

60,000 handwritten digits; the “correct" answer is known.

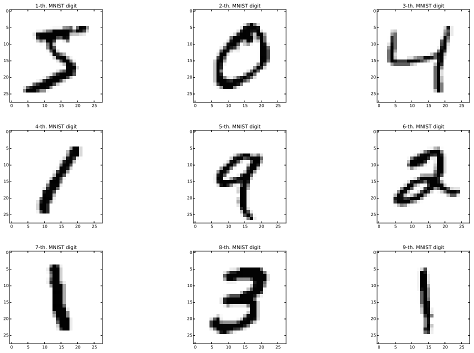
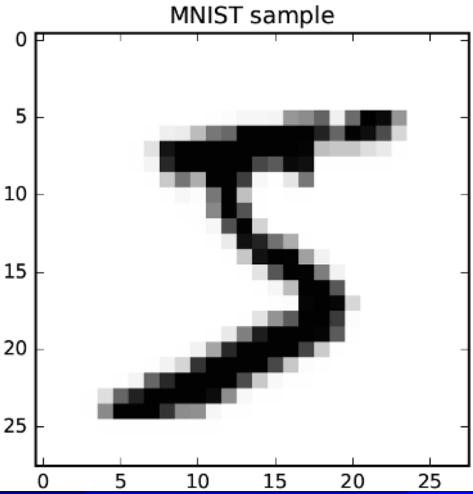


FIG. 1. First 9 MNIST data.

MNIST data: 28×28 grayscale pixels

There is a UV cutoff (pixels are uniform), an IR cutoff close to the overall size and a typical size (width of lines).



A perceptron for the MNIST data (one hidden layer)

We consider a perceptron model where the visible variables v_i are the $28 \times 28 = 784$ pixels of the MNIST digits (in grayscale values between 0 and 1). We decided to take 196=784/4 hidden variables h_k. Later (see hierarchical approximations below), we will “attach" them rigidly to 2×2 blocks of pixels. The hidden variables are:

$$h_k = \sigma(\sum_{j=1}^{784} W^{(1)}_{kj} v_j), k = 1, 2 \ldots 196.$$

We choose the sigmoid function $\sigma(x) = 1/(1 + \exp(-x))$.

The output variables are

$$y_l = \sigma(\sum_{k} W^{(2)}_{lk} h_k)).$$

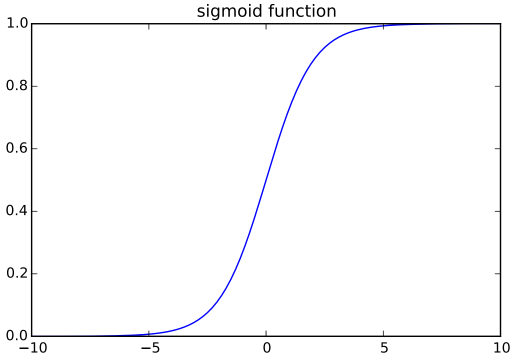
with $l = 0, 1, \ldots 9$ which we want to associate with the MNIST characters by having $y_l \simeq 1$ for $l =$digit while $y_l \simeq 0$ for the 9 others.

The sigmoid function

The sigmoid function

$$\sigma(x) = 1/(1 + \exp(-x))$$

is a popular choice of activation function (0 at large negative input, 1 at large positive input). We have $\sigma(x)' = \sigma(x) - \sigma(x)^2$ which allows simple algebraic manipulations for the gradients.



Gradient search

Given a learning set $\{v_i^{(n)}\}$ with $n = 1, 2, \ldots N \simeq 60,000$ with the corresponding target vectors $\{t_l^{(n)}\}$, we minimize the loss function:

$$\mathcal{E}(W^{(1)}, W^{(2)}) = (1/2) \sum_{n=1}^{N} \sum_{l=0}^{9} (y_l^{(n)} - t_l^{(n)})^2$$

with

$$y_l^{(n)} \equiv \sigma(\sum_k W_{lk}^{(2)} \sigma(\sum_j W_{kj}^{(1)} v_j^{(n)})).$$

The weights matrices $W^{(1)}$ and $W^{(2)}$ are initialized with random numbers following a normal distribution. They are then optimized using a gradient method with gradients:

$$\mathcal{G}_{lk}^{(2)} \equiv \frac{\partial \mathcal{E}}{W_{lk}^{(2)}} = (y_l - t_l)(y_l - y_l^2) h_k$$

$$\mathcal{G}_{kj}^{(1)} \equiv \frac{\partial \mathcal{E}}{W_{kj}^{(1)}} = \sum_l (y_l - t_l)(y_l - y_l^2) W_{lk}^{(2)} (h_k - h_k^2) v_j$$

Performance

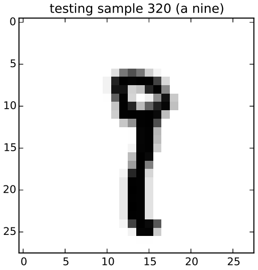
After one “learning cycle" (going through the entire MNIST training data), we get a performance of about 0.95 (number of correct identifications/number of attempts), after 10 learning cycles, the performance saturates near 0.98 (with 196 hidden variables and learning parameters 0.1 and 0.5).

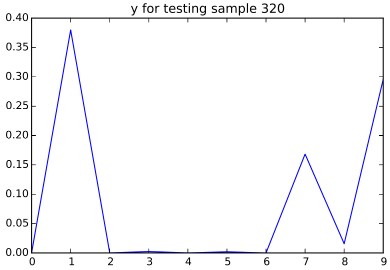
It is straightforward to introduce more hidden layers $y_l^{(n)} \equiv \sigma(W^{(m)}....\sigma(W_{lk}^{(2)}\sigma(W_{kj}^{(1)}v_j^{(n)}))$. With two hidden layers, the performance improves (but only very slightly).

On the other hand, if we remove the hidden layer (as in Rosenblatt experiment), the performance goes down to about 0.91.

Failures (2 percent)

It is instructive to look at the outputs for the 2 percent of cases where the algorithm fails to identify the correct digits. Here, the outputs suggest 1, 7 and 9 with a slightly larger value for 1 than for 9 which is the correct answer according to the MNIST data. In the following, I will focus more on getting comparable performance with less learning parameters rather than reducing the number of failures.





Recent attempts to use RG ideas

- It has been suggested by Methap and Schwab (1410.3831) that the hidden variables can be related to the RG “block variables" and that an hierarchical organization inspired by physics modeling could drastically reduce the number of learning parameters. This may be called “cheap" learning (Lin and Tegmark, 1608.08225).
- The notion of criticality or fixed point has not been identified precisely on the ML side. It is not clear that the technical meaning of “relevant", as used in the RG context to describe unstable directions of the RG transformation linearized near a nontrivial fixed point, can be used generically in ML context.
- We are exploring this question for the MNIST data (which is not “critical") and the more tunable Ising model.

Principal Component Analysis (PCA, 19th cent. tech.)

Identifying “relevant directions" may allow a drastic reduction of the information necessary to calculate observables.

Relevant directions: directions with largest variance

$v_i^{(n)}$: greyscale value of the i-th pixel in the n-th sample
$\bar{v}_i$: average greyscale value of the i-th pixel

covariance matrix: $C_{ij} = (1/N)\sum_{n=1}^{N}(v_i^{(n)} - \bar{v}_i)(v_j^{(n)} - \bar{v}_j)$

We can project the original data in a small dimensional subspace corresponding to the largest eigenvalues of C_{ij}

Can we keep the crucial information in a small subspace?

For the MNIST data, the 784 pixels digits are recognizable using projections in dimension 10-20 subspace

MNIST averages

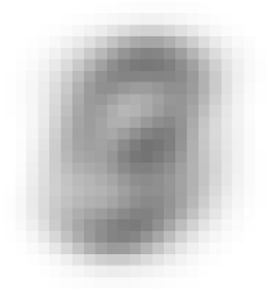
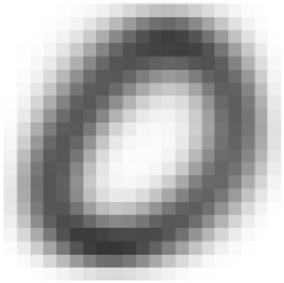
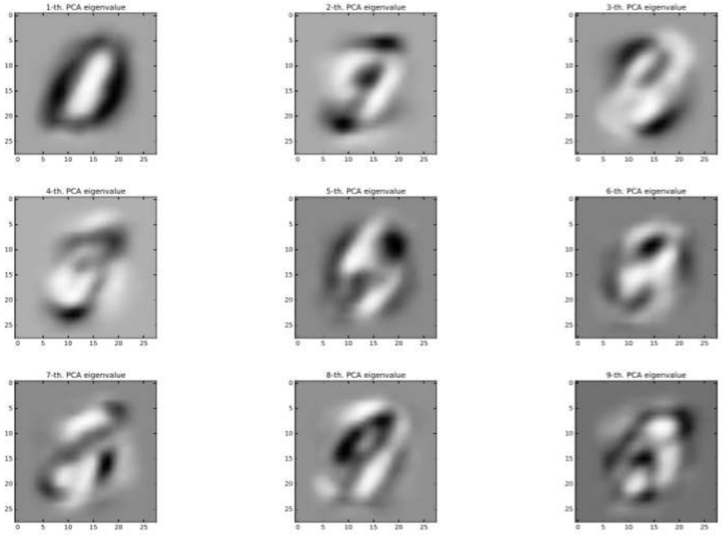



Figure: Empirical probability for 60,000 digits (includes all digits, left) and 5,923 zeros (right).

PCA eigenvectors (“eigenfaces")



PCA projections of a MNIST digit

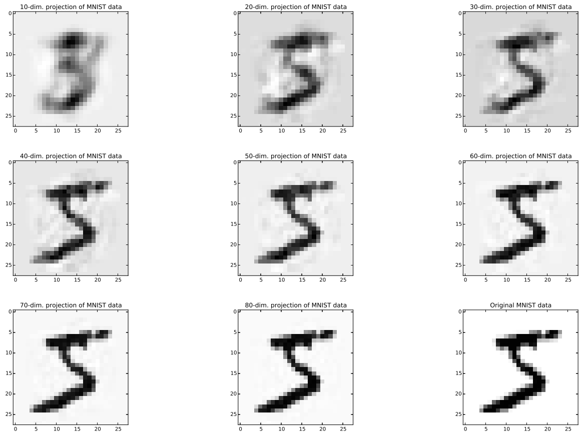


Figure 1: PCA projections in subspaces of dimensions 10, 20, ... 80 compared to the original 784-dimensional data.

PCA projections in subspaces of dimensions 10, 20, ... 80 compared to the original (lower right) 784-dimensional MNIST digit.

Success rate of (drastic) PCA Projections

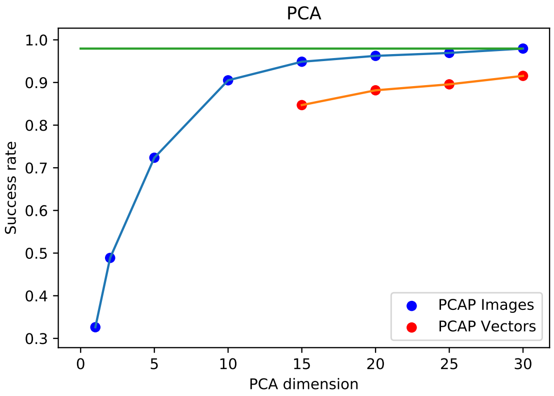
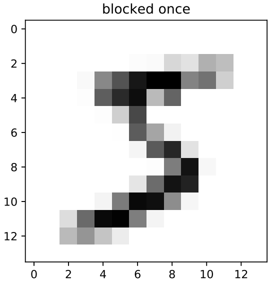
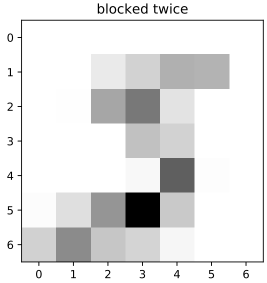


Figure: Success rate for PCA projections of the 28×28 images (blue, using the PCA eigenvectors=images) or using the PCA coordinates only (red, no images) as a function of the dimension of the subspace. The green line represents the asymptotic value (98 percent) for the original version (dimension 784).

Blocking

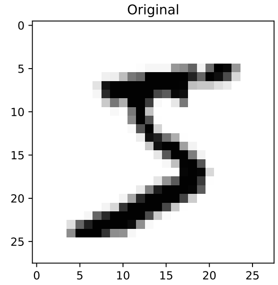
We have used the one layer perceptron with blocked images. First we replaced squares of 4 four pixels by a single pixel carrying the average value of the four blocked pixels. Using the 14×14 blocked pictures with 49 hidden variables, the success rate goes down slightly (97 percent). Repeating once, we obtain 7×7 images. With 25 hidden variables, we get a success rate of 92 percent.

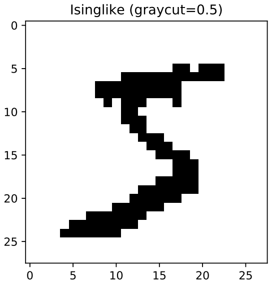




Ising projection

We can replace the grayscale pixels by black and white pixels. This barely affects the performance (97.6 percent) but diminishes the configuration space from 256^{784} to 2^{784} and allows a Restricted Boltzmann Machine treatment (not discussed here).





Cheap learning (hierarchical approximations)

We have considered the hierarchical approximation where each hidden variable is only connected to a single 2×2 block of visible variables (pixels):

$$W^{(1)}_{lj} v_j \rightarrow W^{(1)}_{l,\alpha} v_{l,\alpha}$$

with α = 1, 2, 3, 4 are the position in the 2×2 block and l = 1, ...,196 the labeling of the blocks. Even though the number of parameters that we need to determine with the gradient method is significantly smaller (by a factor 196), the performance remains 0.92. A generalization with 4x4 blocks leads to a 0.90 performance with 1/4 as many weights. This simplified version can be used as a starting point for a full gradient search (pretraining), but the hierarchical structure (sparcity of W_{ij}) is robust and remains visible during the training. This pretraining breaks the huge permutation symmetry of the hidden variables.

Transition to the 2D Ising model

- The MNIST data has a typical size built-in and UV details can be erased until the relevant information contained at that size is reached.
- One can think that the various digits are “phases", but there is nothing like a critical point.
- We now consider worm configurations for the 2D Ising model at different $\beta = 1/T$, some close to the critical value.
- The graphs for the Ising model calculations have been made by Sam Foreman.

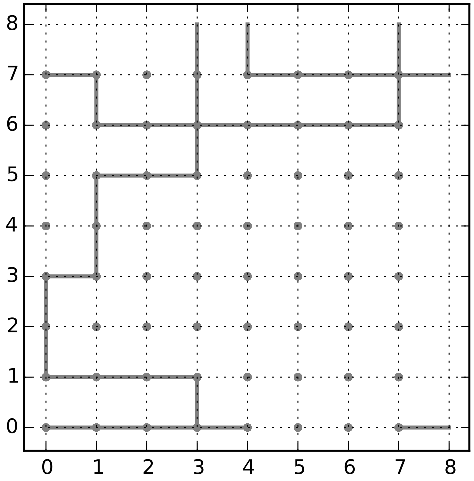
Worm Algorithm

1. Randomly select starting point on lattice, set head $=$ tail $= j$. Choose nearest neighbor site at random for possible update, j^*.
2. If *random*() $\leq R$: assign head to new site, j^*, and update the bond number, i.e. the number of currently active bonds $n_b = n_b \pm 1$, where R is the acceptance ratio, defined by

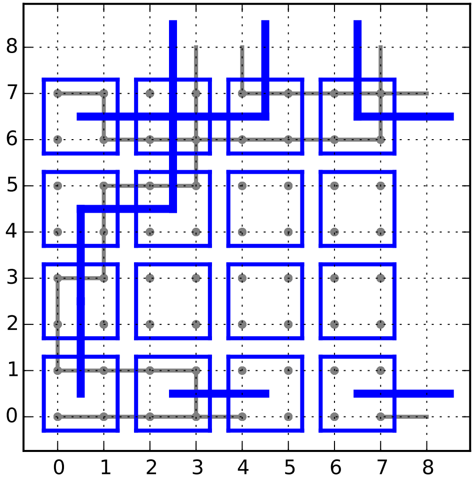
$$R = \begin{cases} \frac{\beta J}{n_b+1}, & \text{when } n_b = n_b + 1 \\ \frac{n_b}{\beta J}, & \text{when } n_b = n_b - 1 \end{cases} \tag{1}$$

3. While head $\neq$ tail, repeat 2. If head $=$ tail, update partition function $Z = Z + 1$, and go to 1.
4. Note that when this update is accepted, the global number of active bonds N_b is updated by one, i.e. $N_b = N_b + 1$.
5. For the Ising model, one can use resummed versions with links occupied 0 or 1 time with a weight $th(\beta)$.

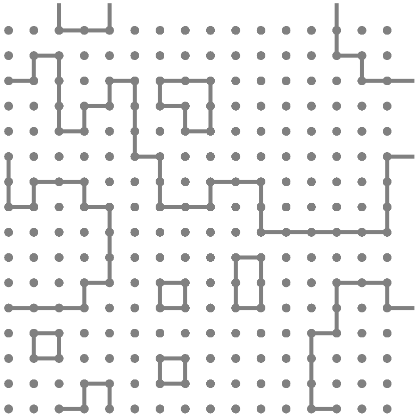
Sampling of Ising HT contributions (“worms")



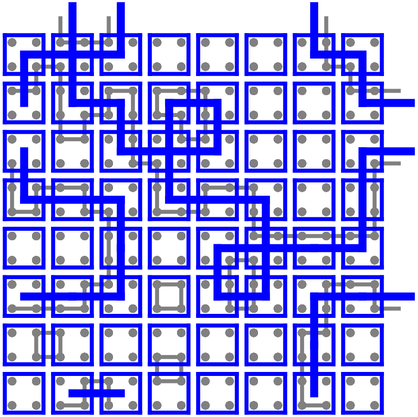
Blocked worms (8x8)



Worms (16x16)



Blocked worms (16x16)

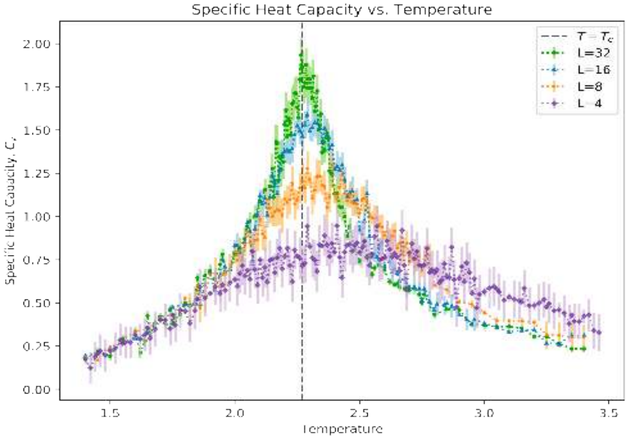


Blocking

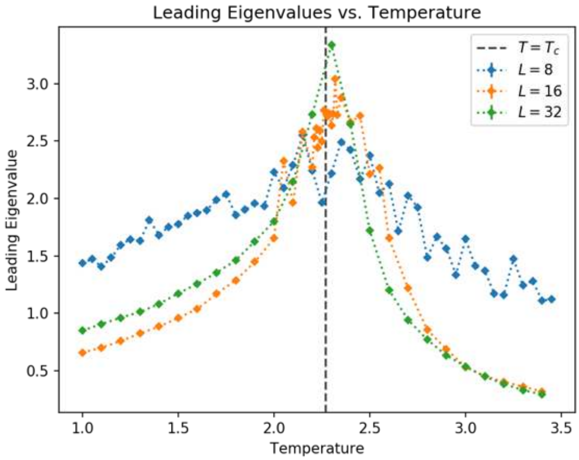
- We implement a ‘coarse-graining‘ renormalization approach, where the lattice is divided into blocks of 2×2 squares.
- Each 2×2 square is then ‘blocked‘ into a single site, where the new external bonds in a given direction are determined by the number of active bonds exiting a given square.
- If a given block has one external bond in a given direction, the blocked site retains this bond in the blocked configuration, otherwise it is ignored.
- Results for the leading PCA eigenvalue and specific heat for blocked configurations are qualitatively similar to the unblocked results. These results and their RG interpretation are still preliminary and will not be presented here.
- The blocking procedure is approximate. Better approximations can be constructed with the TRG method (YM arXiv:1211.3675, Phys. Rev. B 87, 064422). Worm implementation of the critical theory are in progress.

Specific Heat (Sam Foreman)

The worm algorithm allows statistically exact calculations of the specific heat. In order to observe finite scaling effects, we performed this analysis for lattice sizes $L = 4, 8, 16, 32$.

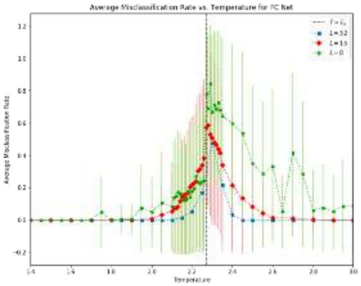


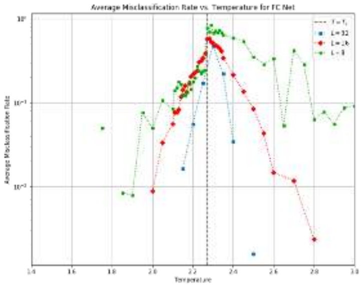
PCA leading eigenvalue (Sam Foreman)



Finding the phase from worm pictures (Sam Foreman)

The one layer perceptron with fully connected W can be used to try to determine the phase from a worm picture. The target values are determined using the known infinite volume transition. Obviously, this information becomes more accurate as the size increases.





Output functions for fully connected W (Sam Foreman)

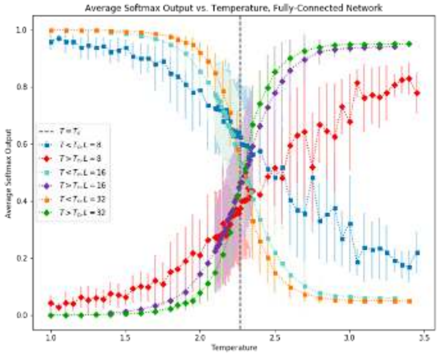


Figure: Softmax outputs vs. temperature. These results can be improved using ConvNet algorithms which have a clear RG flavor (Sam Foreman, work in progress).

Conclusions

- ML has a clear RG flavor: how to extract relevant features from noisy pictures (configurations).
- RG ideas allow to reduce the complexity of the perceptron algorithms for the MNIST data.
- Direct use of Tensor RG methods are being used for the ML treatment of worm configurations of the 2D Ising model near criticality. Quantitative tests in progress.
- Restricted Boltzmann Machines=Ising models
- ML $\rightarrow$ Lattice Field Theory: compress configurations, find features or updates (Lei Wang 1610.02746)
- Private funding for Lattice Field Theory (e. g. to bring students to conferences!)?

Thanks for listening

Acknowledgements:

This research was supported in part by the Dept. of Energy under Award Numbers DOE grants DE-SC0010114, DE-SC0010113, and DE-SC0013496.

Restricted Boltzmann Machines (say for MNIST data)

v (visible) and **h** (hidden) vectors of binary variables (Ising spins).
v are the 784 black and white pixels of a 28×28 MNIST picture.

$$p(\mathbf{v},\mathbf{h}) = (1/Z)\exp(h_k W_{kj} v_j + b_k h_k + a_j v_j)$$

$$p(\mathbf{v}) = \sum_{\{h_k=0,1\}} (1/Z)\exp(h_k W_{kj} v_j + b_k h_k + a_j v_j)$$

$$Z = \sum_{\{h_k=0,1, v_j=0,1\}} \exp(h_k W_{kj} v_j + b_k h_k + a_j v_j)$$

Given an empirical image **v**, you can maximize the model probability (determine the learning parameters W_{kj} etc.) by using gradients:

$$\partial \ln p(\mathbf{v})/\partial W_{kj} = (1/Z(\mathbf{v}))\partial Z(\mathbf{v})/\partial W_{kj} - (1/Z)\partial Z/\partial W_{kj}$$

$$Z(\mathbf{v}) = \sum_{\{h_k=0,1\}} \exp(h_k W_{kj} v_j + b_k h_k + a_j v_j)$$

For $Z(\mathbf{v})$, **v** correspond to the black and white pixels of a given image, the summation over the binary hidden variables factorizes and can performed analytically: $(1 + \exp(b_k + W_{kj} v_j))$. On the other hand, Z requires the sum over the 2^{784} visible binary variables ($v_j = (1 + \sigma_j)/2$ are essentially Ising spins).