EPJ Web of Conferences **175**, 11025 (2018) https://doi.org/10.1051/epjconf/201817511025
*Lattice 2017*

# RG-inspired machine learning for lattice field theory

*Sam* Foreman[1], *Joel* Giedt[2], *Yannick* Meurice[1,⋆], and *Judah* Unmuth-Yockey[1,3]

[1] *Department of Physics and Astronomy, The University of Iowa, Iowa City, IA 52242, USA*
[2] *Department of Physics, Applied Physics and Astronomy, Rensselaer Polytechnic Institute, Troy, NY 12180*
[3] *Department of Physics, Syracuse University, Syracuse, New York 13244, United States*

**Abstract.**
Machine learning has been a fast growing field of research in several areas dealing with large datasets. We report recent attempts to use renormalization group (RG) ideas in the context of machine learning. We examine coarse graining procedures for perceptron models designed to identify the digits of the MNIST data. We discuss the correspondence between principal components analysis (PCA) and RG flows across the transition for worm configurations of the 2D Ising model. Preliminary results regarding the logarithmic divergence of the leading PCA eigenvalue were presented at the conference. More generally, we discuss the relationship between PCA and observables in Monte Carlo simulations and the possibility of reducing the number of learning parameters in supervised learning based on RG inspired hierarchical ansatzes.

## 1 Introduction

Machine learning has been a fast growing field of research in several areas dealing with large datasets and should be useful in the context of lattice field theory [1]. In these proceedings, we briefly introduce the concept of machine learning. We then report attempts to use renormalization group (RG) ideas for the identification of handwritten digits [2]. We review the multiple layer perceptron [3] as a simple method to identify digits with a high rate of success (typically 98 percent). We discuss the method of principal components analysis (PCA) as a way to identify *relevant* features. We consider the effects of PCA projections and coarse graining procedures on the success rate of the perceptron. The identification of the MNIST digits is not really a case where some critical behavior can be reached. In contrast, the two-dimensional (2D) Ising model near the critical temperature $T_c$ offers the chance to sample the high temperature contours (closed "worms" [4]) at various temperatures near $T_c$. At the conference, we gave preliminary evidence that the leading PCA eigenvalue of the worm pictures has a logarithmic singularity related in a precise way to the singularity of the specific heat. We also discussed work in progress relating the coarse graining of the worm images to an approximate procedure in the tensor renormalization group (TRG) treatment of the Ising model [5–7]. After the conference, much progress has been made in regard to this question. We briefly mention an upcoming preprint about this question [8].

⋆Speaker, e-mail: yannick-meurice@uiowa.edu

## 2 What is machine learning?

If you open the web page of a machine learning (ML) course, you are likely to find the following definition: “machine learning is the science of getting computers to act without being explicitly programmed" (see e.g. Andrew Ng’s course at Stanford [1]). You are also likely to find the statement that in the past decade, machine learning has been crucial for self-driving cars, speech recognition, effective web search, and the understanding of the human genome. We hope it will also be the case for lattice field theory. From a pragmatic point of view, ML amounts to constructing functions that provide features (outputs) using data (inputs). These functions involve “trainable parameters" which can be determined using a “learning set" in the case of supervised learning. The input-output relation can be written in the generic form $\mathbf{y}(\mathbf{v}, \mathbf{W})$ with $\mathbf{v}$ the inputs, $\mathbf{y}$ the outputs and $\mathbf{W}$ the trainable parameters. This is illustrated in Fig. 1 as a schematic representation of the so-called perceptron [3] where the function’s outputs have the form $y_l = \sigma(\sum_j W_{lj} v_j)$ with $v_j$ the pixels, $W_{lj}$ the tunable parameters and $\sigma(x)$ the sigmoid function defined below. This simple parametrization allows you to recognize correctly 91 percent of the digits of the testing set of the MNIST data (see section 3).

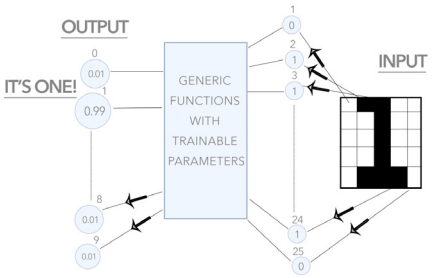


**Figure 1.** Schematic representation of Frank Rosenblatt’s perceptron [3].

## 3 The MNIST data

A classic problem in ML is the identification of handwritten digits. There exists a standard learning set called the MNIST data [2]. It consists of 60,000 digits where the correct answer is known case by case. Each image is a square with $28 \times 28$ grayscale pixels. There is a UV cutoff (pixels are uniform), an IR cutoff (the linear size is 28 lattice spacings) and a typical size (the width of lines is typically 4 or 5). Unless you are attempting to get a success rate better than 98 percent, you may use a black and white approximation and consider the images as Ising configurations: if the pixel has a value larger than some gray cutoff (0.5 on Fig. 3), the pixel is black, and white otherwise. Another simplification is to use a blocked image, namely replacing groups of four pixels forming a 2 by 2 square by their average grayscale. The blocking process can only be repeated 5 times, after that, we obtain a uniform grayscale that makes the identification of the digit difficult.

We now consider a simple model, called the perceptron, which generates 10 output variables, one associated with each digit, using functions of the pixels (the visible variables) with one intermediate set of variables called the hidden variables. The visible variables $v_i$ are the $28 \times 28 = 784$ pixel’s

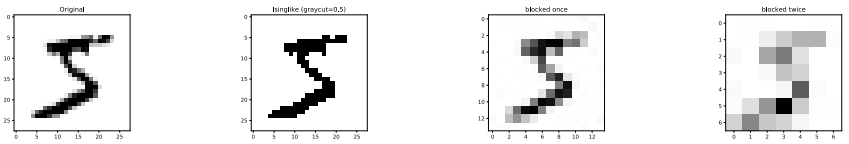


**Figure 2.** An example of MNIST image, a black and white approximation with a graycut at 0.5, the image after one and two blockings (left to right). On the rightmost picture, the blocks should appear sharply otherwise you need to use another viewer, Acrobat is showing it properly.

grayscale values between 0 and 1. We decided to take 196=784/4 hidden variables $h_k$. Later (see hierarchical approximations in section 5), we will "attach" them rigidly to $2 \times 2$ blocks of pixels. The hidden variables are defined by a linear mapping followed by an activation function $\sigma$.

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

We choose the sigmoid function $\sigma(x) = 1/(1 + \exp(-x))$, a popular choice of activation function which is 0 at large negative input, 1 at large positive input. We have $\sigma(x)' = \sigma(x) - \sigma(x)^2$ which allows simple algebraic manipulations when computing gradients. The output variables are defined in a similar way as functions of the hidden variables

$$y_l = \sigma(\sum_k W_{lk}^{(2)} h_k). \tag{2}$$

with $l = 0, 1, \ldots 9$ which we want to associate with the MNIST characters by having target values $y_l \simeq 1$ for $l$ =digit while $y_l \simeq 0$ for the 9 others. The trainable parameters $W_{kj}^{(1)}$ and $W_{kj}^{(2)}$ are determined by gradient search. Given the MNIST 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:

$$\mathcal{E}(W^{(1)}, W^{(2)}) = (1/2) \sum_{n=1}^{N} \sum_{l=0}^{9} (y_l^{(n)} - t_l^{(n)})^2. \tag{3}$$

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:

$$\mathcal{G}_{lk}^{(2)} \equiv \frac{\partial \mathcal{E}}{W_{lk}^{(2)}} = (y_l - t_l)(y_l - y_l^2) h_k \tag{4}$$

$$\mathcal{G}_{kj}^{(1)} \equiv \frac{\partial \mathcal{E}}{W_{kj}^{(1)}} = \sum_l (y_l - t_l)(y_l - y_l^2) W_{lk}^{(2)} (h_k - h_k^2) v_j. \tag{5}$$

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 on an independent testing set of 10,000 digits). After 10 learning cycles, the performance saturates near 0.98 (with 196 hidden variables and learning parameters, which control the gradient changes, 0.1 and 0.5). It is straightforward to introduce more hidden layers: $y_l^{(n)} \equiv \sigma(W^{(m)} \ldots \sigma(W_{lk}^{(2)} \sigma(W_{kj}^{(1)} v_j^{(n)})) \ldots)$. With two hidden layers, the

performance improves (but only very slightly). On the other hand, if we remove the hidden layer, the performance goes down to about 0.91 as mentioned above. It is instructive to look at the outputs for the 2 percent of cases where the algorithm fails to identify the correct digits. There are often cases where humans would have hesitations. In the following, we will focus more on getting comparable performance with less learning parameters rather than attempting to reduce the number of failures.

It has been suggested [9, 10] that the hidden variables can be related to the RG "block variables" and that a hierarchical organization inspired by physics modeling could drastically reduce the number of learning parameters. This may be called "cheap" learning [11]. 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 a precise way in the RG context to describe unstable directions of the RG transformation linearized *near a fixed point*, can be used generically in ML context. The MNIST data does not seem to have "critical" features and we will reconsider this question for the more tunable Ising model near the critical temperature (see section 6).

## 4 Principal component analysis (PCA)

The PCA method has been used successfully for more than a century. It consists of identifying directions with the largest variance (most relevant directions). It may allow a drastic reduction of the information necessary to calculate observables. We call $v_i^{(n)}$ the grayscale value of the $i$-th pixel in the $n$-th MNIST sample. We first define $\bar{v}_i$ as the average grayscale value of the $i$-th pixel over the learning set. This average is shown in Fig. 3. We can now define the covariance matrix:

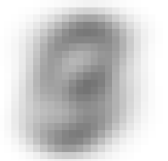

**Figure 3.** Average of each of the 784 pixels over the MNIST learning set.

$$C_{ij} = (1/N)\sum_{n=1}^{N}(v_i^{(n)} - \bar{v}_i)(v_j^{(n)} - \bar{v}_j). \tag{6}$$

We can project the original data onto a small dimensional subspace corresponding to the largest eigenvalues of $C_{ij}$. The first nine eigenvectors are displayed in Fig. 4. The projections in subspaces of dimension 10, 20, ... 80 are shown in Fig. 5.

## 5 RG-inspired approximations

We have reconstructed PCA-projected training and learning sets either as images (we keep 784 pixels after the projection, as shown in Fig. 5, and proceed with the projected images following the usual procedure), or as abstract vectors (the coordinates of the image in the truncated eigenvector basis without using the eigenvectors, a much smaller dataset). The success rate of these PCA projections are shown on Fig. 6.

We have used the one hidden layer perceptron with blocked images. First we replaced squares of four ($2 \times 2$) pixels by a single pixel carrying the average value of the four blocked pixels. Using the $14 \times 14$ blocked pictures with 49 hidden variables, the success rate goes down slightly (97 percent). Repeating once, we obtain $7 \times 7$ images. With 25 hidden variables, we get a success rate of 92 percent.

As mentioned before, we can also 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.

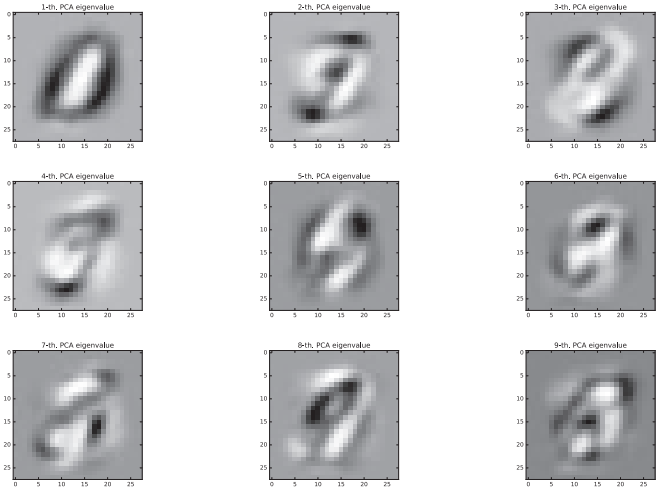


**Figure 4.** First 9 PCA eigenvectors.

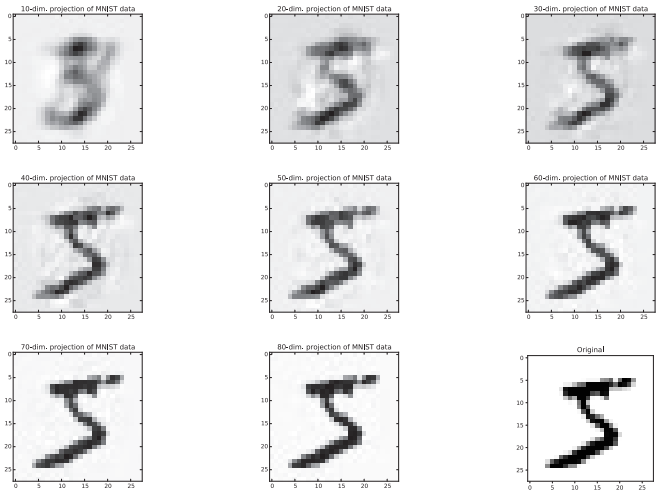


**Figure 5.** PCA projections in subspaces of dimensions 10, 20, ... 80 compared to the original 784-dimensional data in the lower right corner.

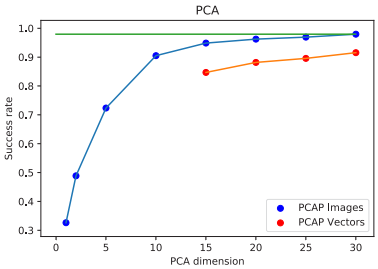


**Figure 6.** Success rate for PCA projections of the $28 \times 28$ images (blue, using the PCA eigenvectors=images) or using the PCA coordinates only (red, no images) as a function of the dimension of the subspace. The green line represents the asymptotic value (98 percent) for the original version (dimension 784).

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

$$W^{(1)}_{lj} v_j \to W^{(1)}_{l,\alpha} v_{l,\alpha} \tag{7}$$

with $\alpha = 1, 2, 3, 4$ are the position in the $2 \times 2$ 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.

## 6 Transition to the 2D Ising model

The MNIST data has a typical size built in the images, namely the width of the lines, and UV details can be erased without drastic effects until that size is reached. One can think that the various digits are separate "phases", but there is nothing like a critical point connected to all the phases. It might be possible to think of the average $\bar{v}_i$ as a fixed point. However, there are no images close to it. In order to get images that can be understood as close to a critical point, we will consider the images of worm configurations for the 2D Ising model at different $\beta = 1/T$, some close to the critical value. The graphs in this section have been made by Sam Foreman.

The worm algorithm [4] allows us to sample the set of contours appearing in the high-temperature expansion, whose statistics are governed by the number of active bonds in a given configuration. An example of an equilibrium configuration is shown in Fig. 7. We implemented a "coarse-graining" procedure where the lattice is divided into blocks of $2 \times 2$ squares, essentially reducing the size of each linear dimension by two. Each $2 \times 2$ 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 exactly one external bond in a given direction, the blocked site retains this bond in the blocked configuration, otherwise it is ignored. The motivation for this procedure and comparison with other blocking procedures tried after will be discussed in Ref. [8]. This is illustrated on the right side of Fig. 7.

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

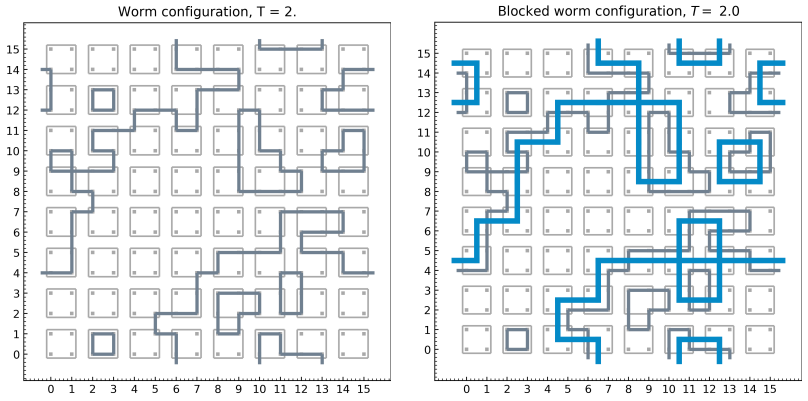


**Figure 7.** Example of legal high temperature contribution also called a worm (left). All the paths close due to periodic boundary conditions. An example of "worm blocking" applied to the same configuration (right).

will be presented elsewhere [8], we conjectured that near criticality, the largest PCA eigenvalue $\lambda_{max}$ is proportional to the specific heat per unit of volume, with a proportionality constant $\frac{2}{3}\left(\ln(1+\sqrt{2})\right)^2 \approx 0.52$. This is in agreement with our results in Fig. 8.

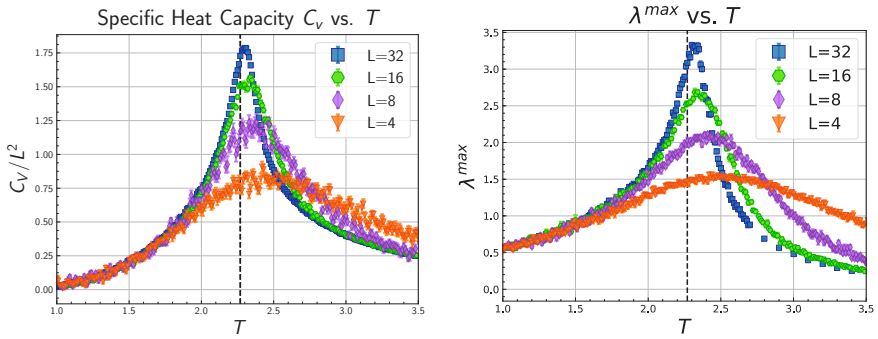


**Figure 8.** Specific heat capacity (left) and largest PCA eigenvalue (right) for various lattice sizes helps to illustrate the conjectured proportionality constant. The dashed black line represents the critical temperature, $T_c$.

To cross check the accuracy of the worm for some observables, the TRG was used. We calculated the average number of occupied bonds, $\langle N_b \rangle$, for a blocked and unblocked lattice using TRG, and calculated $\langle (N_b - \langle N_b \rangle)^2 \rangle$ for the unblocked case, and good agreement was found between the worm and the TRG. The TRG also provides a clear picture of what bond numbers are associated with which states as opposed to editing pictures by setting pixels to chosen values, such as what happens in the blocked case for a block of four sites. This correspondence makes the picture of RG for the configurations sturdier.

Results for the leading PCA eigenvalue and specific heat for blocked configurations were found qualitatively similar to the unblocked results. The blocking procedure is approximate. Systematically

improvable approximations can be constructed with the TRG method [5–7]. Improvement of the blocking method used here was developed after the conference and will be presented elsewhere [8].

## 7 Conclusions

ML has a clear RG flavor. A general question in this field is how to extract relevant features from noisy pictures ("configurations"). The RG ideas allow us 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 and will be presented elsewhere [8]. Sam Foreman has been developing methods to identify the side of transition. He has improved upon the fully-connected perceptron model by implementing a modified convolutional neural network (ConvNet), designed to respect the periodic boundaries of the training data. ConvNets have wide applications in image recognition problems, and in particular, are known to perform extremely well at classification tasks [12]. See also Refs. [13, 14]. On a large enough lattice, the methods could be extended to identify the temperature of an individual configuration, borrowing from the master field ideas presented at the conference [15]. Judah Unmuth-Yockey has made progress with restricted Boltzmann machines (RBMs) learning on configurations of Ising spins. Here it was found that the RBMs are capable of learning the local two-body interaction of the original model. The Hamiltonian of the learned RBM is then comprised of effective $n$-body interactions. ML could benefit lattice field theory, for instance how to compress configurations, find features or updates [16].

This work was supported by the U.S. Department of Energy (DOE), Office of Science, Office of High Energy Physics, under Award Numbers DE- SC0013496 (JG) and DE-SC0010113 (YM).

## References

[1] Andrew Ng, Machine Learning, Stanford course available from Coursera
[2] See Yann Lecun website, http://yann.lecun.com/exdb/mnist/
[3] F. Rosenblatt, *The perceptron*, Psychological Review, Vol. 65, No. 6 (1958)
[4] N. Prokof'ev, B. Svistunov, Phys. Rev. Lett. **87**, 160601 (2001)
[5] Y. Meurice, Phys. Rev. **B87**, 064422 (2013), 1211.3675
[6] E. Efrati, Z. Wang, A. Kolan, L.P. Kadanoff, Rev. Mod. Phys. **86**, 647 (2014)
[7] Y. Liu, Y. Meurice, M.P. Qin, J. Unmuth-Yockey, T. Xiang, Z.Y. Xie, J.F. Yu, H. Zou, Phys. Rev. **D88**, 056005 (2013), 1307.6543
[8] S. Foreman, J. Giedt,Y. Meurice, and J. Unmuth-Yockey, Machine learning inspired analysis of the Ising model transition, preprint in progress
[9] P. Mehta, D.J. Schwab, ArXiv e-prints (2014), 1410.3831
[10] D.J. Schwab, P. Mehta, ArXiv e-prints (2016), 1609.03541
[11] H.W. Lin, M. Tegmark, ArXiv e-prints (2016), 1608.08225
[12] Dan Cireşan, Ueli Meier, Juergen Schmidhuber, ArXiv e-prints (2012), 1202.2745
[13] Carrasquilla, J., Melko, R. G., *Machine learning phases of matter.* Nat. Phys. http://dx.doi.org/10.1038/nphys4035 (2017)
[14] Akinori Tanaka and Akio Tomiya, J. Phys. Soc. of Japan 86 063001 (2017), 1609.09087
[15] M. Luscher, *Stochastic locality and master-field simulations of very large lattices*, in *Proceedings, 35th International Symposium on Lattice Field Theory (Lattice2017): Granada, Spain*, to appear in EPJ Web Conf., 1707.09758, http://inspirehep.net/record/1613675/files/arXiv:1707.09758.pdf
[16] Li Huang and Lei Wang, Phys. Rev. **B95**, 035105 (2017), 1610.02746