Exercise 4
Supervised Learning and Bayesian Inference

Paolo Frazzetto, Enrico Lorenzetti

Università degli Studi di Padova
Statistical Mechanics of Complex Systems
Prof. S. Suweis, Prof. A. Maritan

14 June, 2019

1 Introduction

In this work, we aim to discuss a study of a Deep Learning Neural Network (DNN), built to acquire the ability of discerning the two phases of a $2D$ *Ising Model* - ordered and disordered. Nevertheless, since it is very hard to evaluate the state of the system when it is close to the critical point, we add a third phase which we denote as the *critical phase.*
The Hamiltonian for the classical Ising model is

$$H = -\sum_{<i,j>} J\sigma_i\sigma_j \qquad \sigma \in \{\pm 1\} \tag{1}$$

where $< i, j >$ indicates the sum of all nearest neighbours with periodic boundary condition and J is some arbitrary constant. Onsager proved that this system, in the thermodynamic limit, has a phase transition at $\frac{T_C}{J} \approx 2.26$. Within a finite system, as we are dealing with a block of 40×40 spins, we expect that this theoretic value is an overestimation. We consider three regions of temperature: for $T < 2.0$ as ordered, for $2.0 < T < 2.5$ as near-critical and for $T > 2.5$ as disordered. This work is based on the review A high-bias, low-variance introduction to Machine Learning for physicists by Metha et al. We deal a data set of 10^4 states obtained using Monte-Carlo simulation at fixed temperature. Even so, the critical states are not used for training.

1.1 Structure of DNN

The structure of the network is very essential. We have a single hidden layer containing a number of hidden neurons. Its architecture consists in Re-Lu activated input layer, the hidden layer, and the softmax output layer. Figure 1 shows this graph structure made with TensorBoard. We choose the cross-entropy cost function and we minimise it using the *ADAM optimiser.* We train the DNN for 100 epochs.

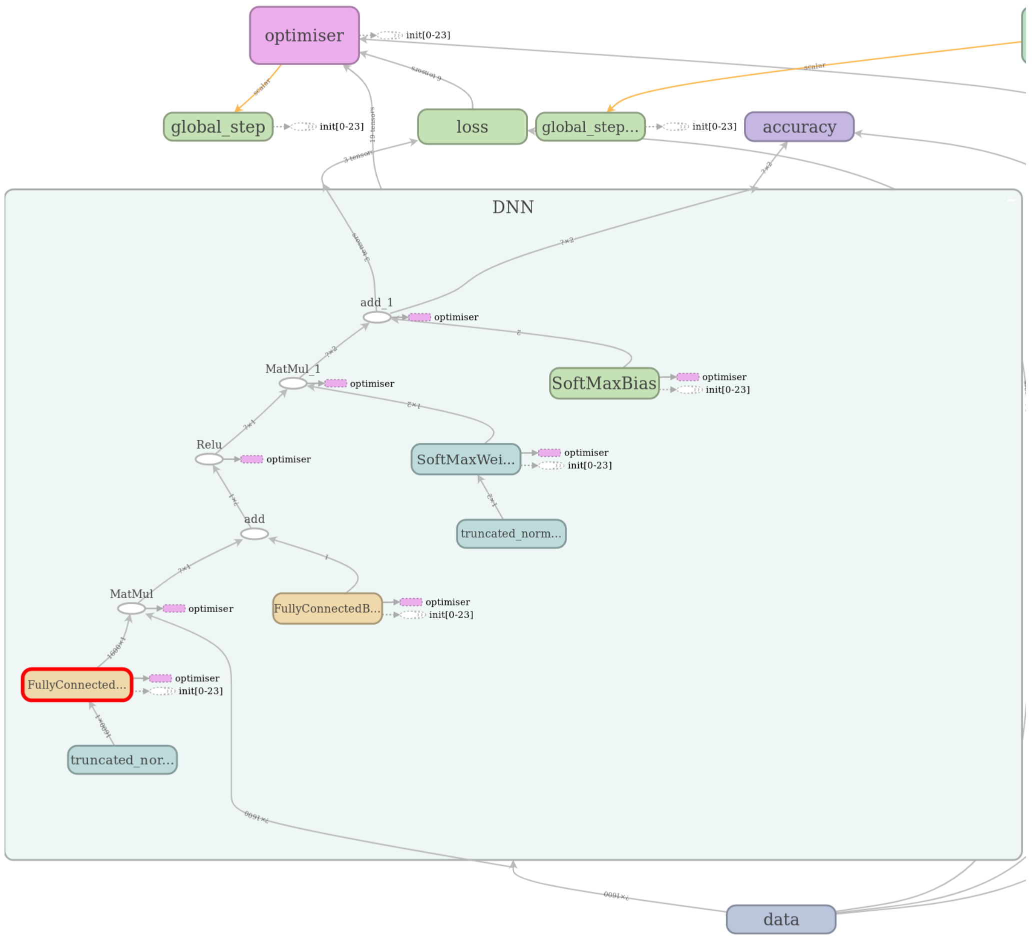


Figure 1: NN TensorBoard illustration.

2 Implementation

Generally speaking, the steps required to implement a NN are:

1. Load and classify data;
2. Set the model and its the architecture;
3. Define a cost function and optimisation procedure;
4. Train the model;
5. Test the results using the validation set;
6. Tune the hyperparameters to optimise the results.

Firstly, data is divided in three groups, depending on the phase. As previously mentioned, we use the ordered and disordered data to create a training and a test data set for the problem. Classifying the states in the critical region is expected to be harder and we only use this data eventually to test the performance of our model. We followed the provided *notebook* and used the TensorFlow library to build our NN. A grid search has been carried out to find the best number of neurons and learning rate, as shown in Figures 2 for each of the training, test and critical data sets. We notice that 10 neurons are enough at a learning rate of 0.1 to get to a very high accuracy on the test set (99.9%). However, if we aim at capturing the physics of the DNN for the Ising classification problem as a function of the learning rate and the number of hidden neurons close to criticality, clearly more neurons are required to reliably learn the more complex correlations in the Ising critical configurations.

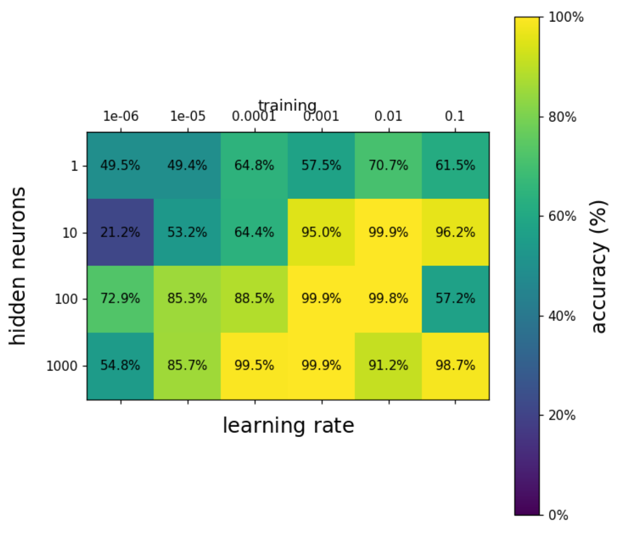
3 A Bayesian approach

The estimated weights obtained in the previous work are fixed once you trained the net. However, there could be another approach to this problem which can rather determine a probabilistic distribution of a given configuration of weights $\theta = (\theta_1, .., \theta_M)$. To do so, if we call x_i the input vector and y_i the output one in our DNN and $\mathcal{D} = (x, y)$ all our data set, we can obtain the posterior, i.e. the distribution probability of the parameters given the N samples, exploiting Bayes' rule

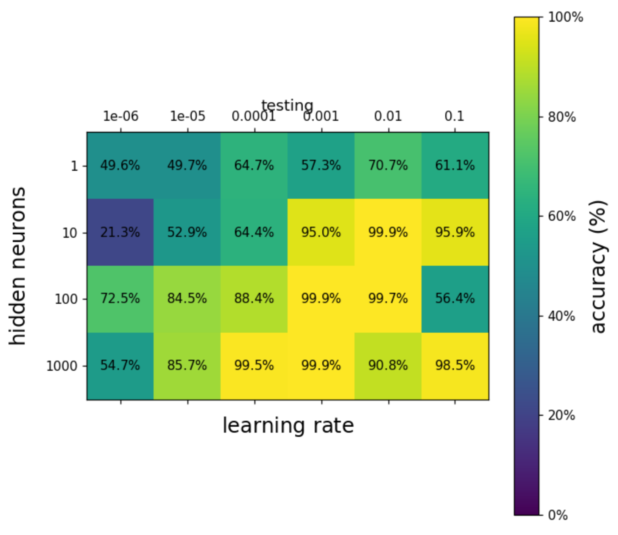
$$P(\theta|\mathcal{D}) = \frac{\overbrace{P(\mathcal{D}|\theta, x)}^{\text{likelihood}}\overbrace{P(\theta)}^{\text{prior}}}{\underbrace{P(\mathcal{D})}_{\text{marginal likelihood}}} \tag{2}$$

where $P(\mathcal{D}|\theta) = \prod_{i=1}^{N} P(y_i|x_i, \theta)$ is the probability to get the observed results given a certain set of parameters; $P(\theta)$ is the probability of expecting the parameters without having seen data; and $P(\mathcal{D})$ is just a normalisation factor. The probability distribution of a new testing point $(x^\star, y^\star)$ yields

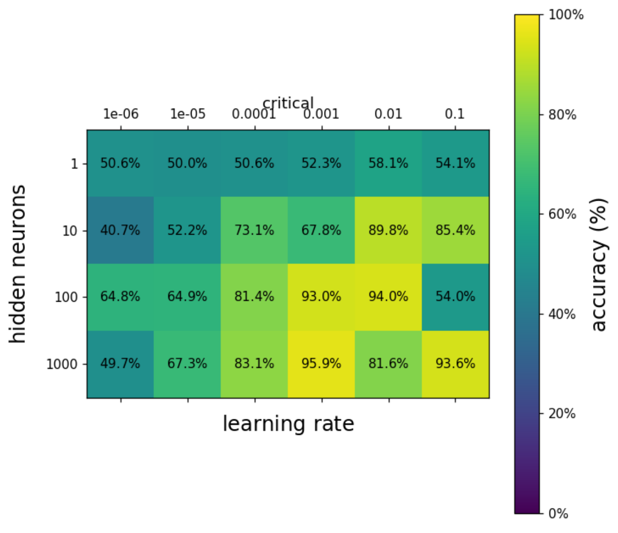
$$P(y^\star|x^\star, \mathcal{D}) = \int P(y^\star|x^\star, \theta)P(\theta|\mathcal{D})d\theta \tag{3}$$



(a) Training



(b) Test



(c) Critical

Figure 2: Accuracy scores for different data sets.