Tutorial One: Ising Model

Learn the Hamiltonian and Classify the Phases

What would you say to someone who gives up on their diet and goes back to eat a lot of South Asian clarified butter?

1 Learn the Hamiltonian

In this part of the tutorial, we will use supervised learning (specifically linear regression) to learn the Hamiltonian of a one-dimensional Ising model from randomly-generated data. The data set includes M samples of N spins and the Hamiltonian computed using the Ising Hamiltonian

$$H_{data} = -\sum_{j=1}^{N} s_j s_{j+1}. \tag{1}$$

where $s_j = 1$ or -1 are randomly chosen. Let us suppose our supervised learning algorithm assumes a very general Hamiltonian function that accounts for the pairwise interactions from all possible pairs of spins:

$$H_{model} = -\sum_{j=1}^{N} \sum_{k>j} J_{jk} s_j s_k. \tag{2}$$

The goal is to learn the coupling parameters J_{jk}.

Recall that the input of a linear regression is a data set $\mathcal{D} = \{(\vec{x}^{(i)}, y^{(i)})\}_{i=1}^{M}$ where M is the number of samples and each sample includes a vector $\vec{x} \in R^d$ and a label y. Linear regression then fits a linear function $f(\vec{x})$ to each of the labels where

$$f(\vec{x}) = \sum_{\alpha=1}^{d} w_\alpha x_\alpha \tag{3}$$

and w_j are the fitting parameters. Applying the linear function to each data, the results can be neatly summarized in matrix form.

$$\vec{y} = X\vec{w} \tag{4}$$

where $\vec{y} = (y^{(1)}, \ldots, y^{(M)})^T$ is the vector of the label with dimension M, $X_{i\alpha} = x_\alpha^{(i)}$ and $\vec{w}$ is the vector of the fitting parameters with dimension d. Note that X is likely not a symmetric matrix. Note that there is an analytical solution to this problem

$$\vec{w}_{exact} = (X^T X)^{-1} X^T \vec{y} \tag{5}$$

Or we can use gradient descent to numerically solve the problem.

a) Theoretically, what should be the prediction of J_{jk}'s?

b) At first glance, our proposed model is not of the same form as the linear function. We need to rewrite our model as

$$H_{model,new} = \sum_p^d w_p x_p \tag{6}$$

What is p? What is x_p in terms of spins? What is w_p in terms of the coupling parameters J_{jk}? What is the dimension d in terms of the number of spins N?

Now open the notebook for linear regression and run the block to generate the data set.

c) In the next block, write a one-line code using the `np.einsum` function so that x is a matrix $M \times d$ that stores the value $s_j s_k$.

Then the upper triangle of this matrix is extracted. (You do not need to write code for this.)

Now let us examine the section "Find the parameters J_{jk} using the exact solution". There are two ways to do this: either by implementing the exact solution formula or by using the `linear_model.LinearRegression` from the package `sklearn`.

d) Fix the number of spins N at 20 (this number is chosen so that the evaluation of the exact solution does not take too long) and vary the number of samples between $M = 1000$ and 10. What behavior do you observe as the number

of samples decreases? When does the transition happen? Do you have an explanation for it? *Hint: For spin N, we need at least __ samples for linear regression to work.*

Now, let us use gradient descent to solve the same problem.

e) For gradient descent, we need to initialize our parameters w's. Write a one-line code to generate random initial parameters. Pay attention to the size/shape. What kind of random numbers (uniform, or Gaussian, or between -1 and 1 or between 0 and 5 actually does not matter).

f) First run the model with `penalty, learning_rate` and `eta0` arguments, at $M = 1000$ and $N = 20$ and learning rate fixed at 0.001. Change the learning rate up or down. What is the change? Can you explain why? Change M and N (you can generously increase the number of spins up to 100) and fix the learning rate to 0.001, does it still work? What do you conclude?

g) Now run some removing `the penalty, learning_rate` , and `eta0` arguments. Comment on what you find. When do we get the predicted value J_{jk} that agrees with your prediction in the first part?

h) Can you compare the efficiency of the exact method and the gradient descent?

2 Classify the Phases

Now we will move on to a 2D Ising example. The 10^4 states were prepared using Monte-Carlo sampling at each fixed temperature T from a pre-defined set $T \in [0.25, \ldots, 4.0]$ (with a step of 0.25) (which totals to 16×10^4 states). The critical temperature can be computed analytically in the thermodynamic limit

$$T_c/J = \frac{2}{\log(1+\sqrt{2})} \sim 2.26 \tag{7}$$

Using this critical temperature, we assign a label to each state according to its phase: 0 for disordered states and 1 for ordered states. The goal of this part of the tutorial is to predict the phase of a sample given the spin configuration using logistic regression. (If this works, we could use the same technique to identify phase transition in more complicated situations where analytical solutions are not available, which is more often the case).

First, let us load the data (this may take a couple of minutes).

a) Take a look at the label distribution. Write a one-line code to determine if the data is balanced or imbalanced.

b) Let us visualize the spin configuration on a 40 by 40 grid. Select some samples between 0 and 60,000, 60,000 and 90,000 and above 90,000 to view the spin configuration. Can you tell which are ordered and which are disordered?

c) Let us randomly select a fraction of the data (10% in the code but the result or timing does not seem to depend on the fraction much) to shuffle the data. Look at the scatter plot of the label to make sure that the data is shuffled.

   Now it is time to apply a logistic regression model to the randomly selected data.

d) Run the model. This will take a couple minutes.

e) The default results use accuracy as the score metric. How does our model perform from the look of the results?

f) Group activity: Run a grid search to optimize the regularization parameter C between 10^{-3} and 10^{3}. With a group of three, each only needs to run 2 different C's. You may want to write a code to access the `'test_score'` and the `'train_score'` to compute the mean and standard deviation to compare different results.

   - What value of C should we pick?
   - How does this model perform?

   For a logistic regression model, it will be good to access the odds as well as the predicted label to see what the model is doing.

g) Looking at the probabilities, is our model very confident?

   The next block plots a few examples to see how our classifier is doing. Some of them are misidentified. But our model is not very confident, so they could be cases with $p \sim 0.5$ and if we change the threshold maybe it will work.

h) Now take a look at the corresponding probabilities together with the corresponding true labels shown in the figure. Are there any concerns? Are there

any cases in which the model is confident but wrong? Could changing the threshold help?

i) Physically, why is it tricky to use logistic regression to classify the phases of the Ising model? *Hint: think about how our data is presented to the model and what information is lost.*

3 Homework One (Numerical): Classify the Phases

In this assignment, we continue the problem of classifying the phases of the Ising Model. You need to submit the written answers to the questions, the plot you make, and the codes you wrote to generate the plots and other codes that you implemented to perform your analysis. The codes need to be in a file that can directly run on symmetry and other files need to be in pdf format.

a) In the tutorial, we used accuracy as the scoring method. In this assignment, you should also use precision, recall, and F1-score to evaluate.

b) We know the temperature of the data. Make a plot of the average accuracy as a function of the temperature. Where does our model perform well? Where does our model perform badly? Could you use your knowledge of the Ising model to explain this? What kind of feature do you think we can add to improve the performance (you don't need to implement this).

c) We have also learned how to use the support vector machine to classify. Use tutorial two's codes as template, train a support vector machine code (what kernel should we use? If unsure, use both. Complicated kernels will take a few minutes to run.) to classify the data. Also make a plot of the average accuracy as a function of the temperature. Compare the two methods: Logistic regression v.s. support vector machine. Which is better? Why?

A Reference

A high-bias, low-variance introduction to Machine Learning for physicists section VI.D and section VII.C.

Lo, ghee stick regression.