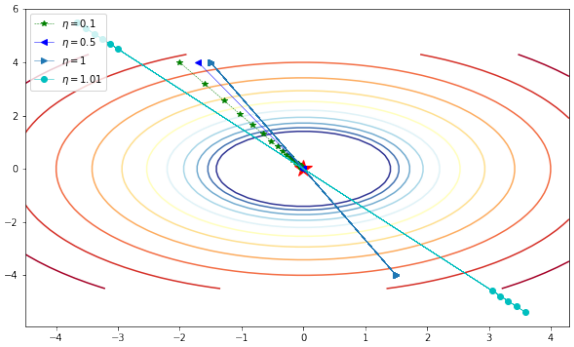
1 Gradient Descent

Gradient descent is an iterative algorithm that aims to minimize a function by taking increments or steps proportional to the negative of the derivative -more generally the gradient- of that function with respect to certain parameters and in the scope of Machine learning this function is the lost function or simply, the Error E(θ) between the predicted points and the labels. The main difference between gradient descent and usual minimizing techniques is the update step, as the main parameter of the predicting model θ is updated in the following manner:

$$v_t = \eta_t \nabla_\theta E(\theta)$$

$$\theta_{t+1} = \theta_t - v_t(1)$$

in the previous equation η_t is called the learning rate, which is the parameters that monitors how would the gradient affect the increments taken as the smaller the η the smaller the effect of the gradient value and more steps would be taken to approach the minima. And vice vers, if the η is too large the algorithm might have large steps which could cause to skip the minima and might even lead to divergence of the method. the effect of η shows in the following figure generated by python notebooks of the review of by Mehta et al.



To be clearer, practically We will consider three simple surfaces: a quadratic minimum of the form z=$ax^2 + by^2$ a saddle-point of the form $z = ax^2 - by^2$, and [Beale's Function], a convex function often used to test optimization problems of the form:

z(x,y) = $(1.5\text{-x+xy})^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$(2)

A final note on this point is that the learning rate η has an upper bound of the optimal learning rate $\eta_o pt$ which is the rate that leads the algorithm to converge directly to the minima, if the η is a value greater than the optimal value but less than twice it it causes the gradient descent to oscillate above and below the minima till it converges but if the η skips the limit of 2 $\eta_o pt$ i.e. becomes larger than twice that value, the Gradient descent diverges. (LeCun et al., 1998b).

1.1 Variations of Gradient descent

The previously mentioned method is the simplest version of gradient descent commonly known as the batch gradient descent because of the usual incremental manner, another version of gradient descent is:

1.2 stochastic gradient descent

which is applied to the type of data that have a random pattern (random probability distribution). One huge drawback of batch (usual) gradient is that under very large number of data points (m) it's very computationally expensive as usually the cost function is a sum from 1 to m which can go to very high orders meaning that for each iteration there's a sum of very large number of terms then for the next step the whole process is conducted again. The main difference between the usual gradient descent and the stochastic gradient descent is the way the cost function is defined differently, in GD (gradient descent) it's only a function of the model's parameter θ while in SGD (stochastic gradient descent the cost function is defined to measure how well the model predicts with respect to a single example (x_i, y_i). So what SGD basically does is that it goes through each training example and find the minimum parameter of the cost function which is much faster and less computationally heavy. A final difference is that the data is already random and being randomly shuffled further more helps diverging to the minima in an even faster manner. A third type of gradient descent that has even a better convergence method is:

1.3 Mini-Batch gradient descent

It's conducted through batching the data into small batches on which the algorithm is done but also with going through each pair of data (x_i, y_i), thus the minimizing parameters are reached in an easy quick manner and also the highly computational procedures are avoided.

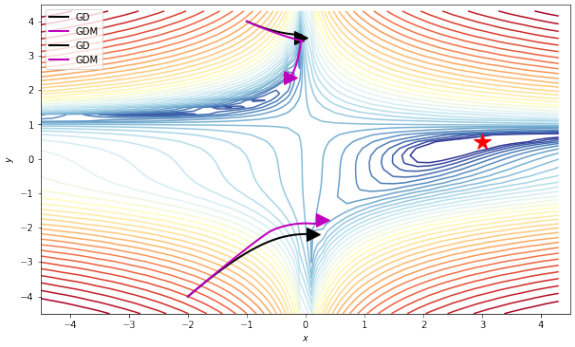
1.4 optimization methods

Gradient Descent has various optimization methods that could direct the algorithm towards the correct direction of minimization to avoid having the curves to be more steep in a certain direction than the others. A method that will be briefly adopted here is the momentum adding method, it adds a term that helps moving the direction of SGD to be quickly reaching the minimum in the following manner:

$$v_t = \gamma v_{t-1} + \eta_t \nabla_\theta E(\theta)$$

$$\theta_{t+1} = \theta_t - v_t (3)$$

This γ term is usually set between zero and 1 and usually called the momentum parameter. Momentum gradient descent can be visulaized in the following figure generated by python notebooks of the review of by Mehta et al.



Other optimization techniques include variations in terms of replacement of the momentum term, for example in the NAG (Nesterov accelerated gradient) the cost function is a function of the momentum term, and the update of the parameters is calculated as follows:

$$v_t = \gamma v_{t-1} + \eta_t \nabla_\theta E(\theta - \gamma v_{t-1})$$

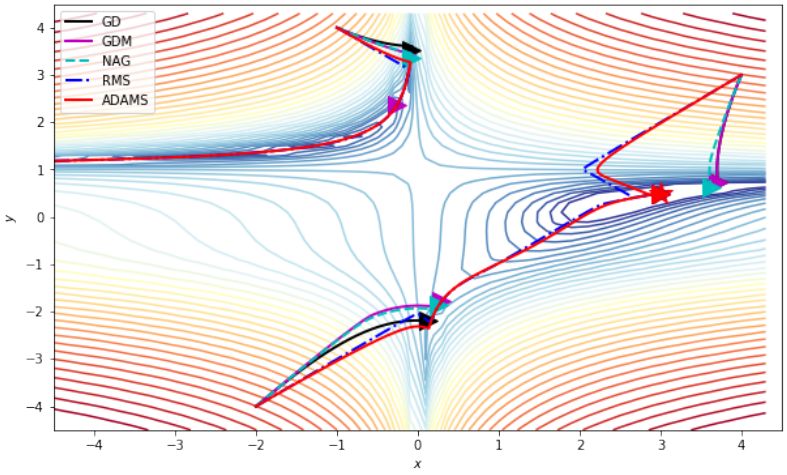
$$\theta_{t+1} = \theta_t - v_t (4)$$

And finally, the Adagrad method that update the θ parameters with smaller updates performing the following calculations:

$$g_{t,i} = \nabla_\theta E(\theta)$$

$$\theta_{t+1,i} = \theta_{t,i} - \eta g_{t,i} (5)$$

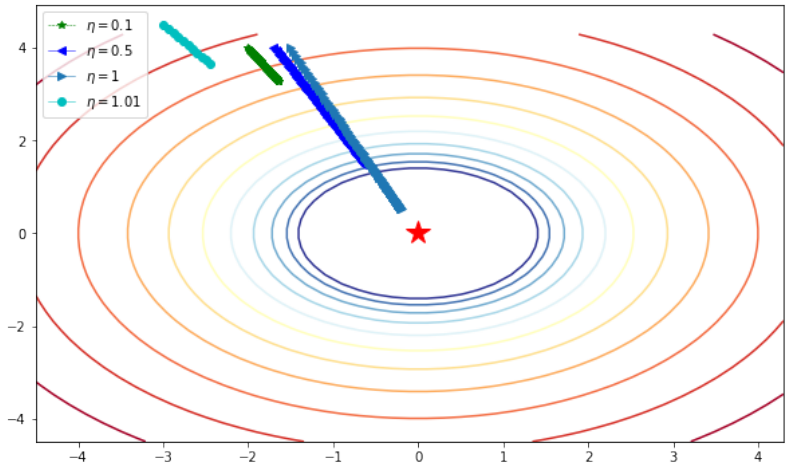
this Graph representing is all methods from Mehta et al review.



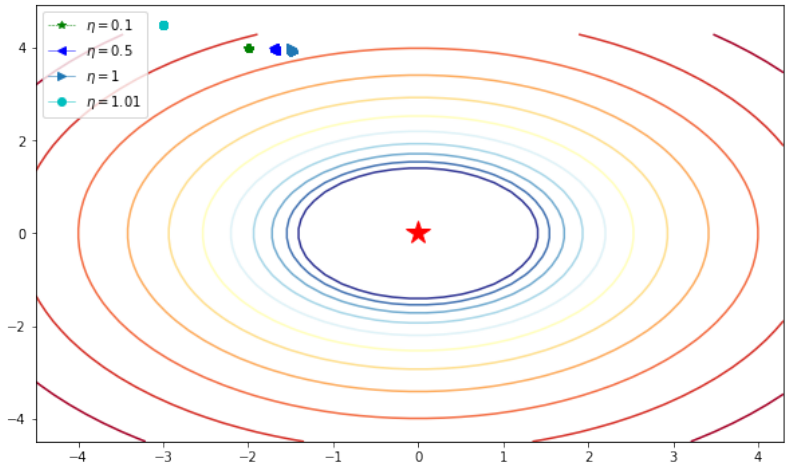
2 Change of parameters

I previously mentioned the change of solutions convergence according to the change of learning rate η but what happens when we change the parameters of the surface that we apply the gradient methods over, previously defined as a and b. The most obvious change appears when a and be have values of 0.01

causes the model to almost converge but weirdly enough not with $\eta = 0.1$ like earlier but with a larger $\eta = 1$



Another bizarre result of parameters change is that when decresing the parameters another bit to be 0.001 the model does not seem to converge with any value of η



3 Linear Regression (Ising Model)

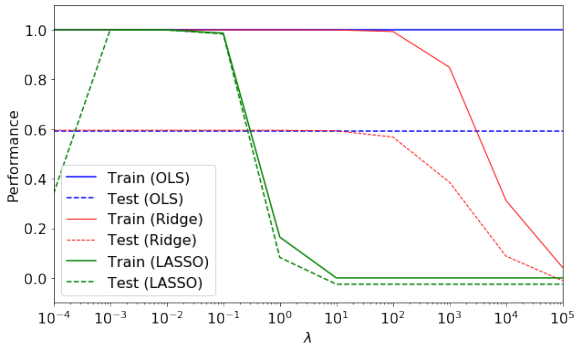
Trying to learn a model representing random spin configurations, we can say that the model of the hamiltonian to go as the nearest neighbour interactions as follows:

$$H = -J \sum_{i=1}^{L} S_i S_{i+1} Where S_i \in (+1, -1) \tag{6}$$

Under the assumption that the data was generated with J=1 the data is represented with the following distribution: $D = (s_{i\,i=1}^{L}, E_i)$
The main idea is to predict E_i without prior knowledge. Trying to put the model in a more familiar form, the Hamiltonian can take the following form:

$$H(S_i) = X^i \cdot J \tag{7}$$

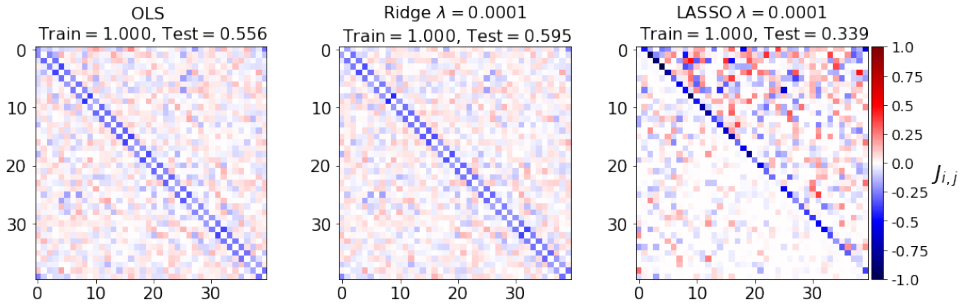
Where X^i can be considered to be the nearest neighbour interaction between spins.
The R^2 parameter is a good way to see how the model is performing and it shows as a function of the regularization parameter λ The following plot from the Methe wide review notebooks show the performance of the three regression types, ordinary least square(OLS), Ridge and LASSO Regression, it shows how the performance change as λ change on the learned data (Training) and the test data.



$$
\begin{aligned}
R^2 &= \left(1 - \frac{u}{v}\right), \\
u &= (y_{pred} - y_{true})^2 \\
v &= (y_{true} - \langle y_{true} \rangle)^2
\end{aligned}
\tag{8}
$$

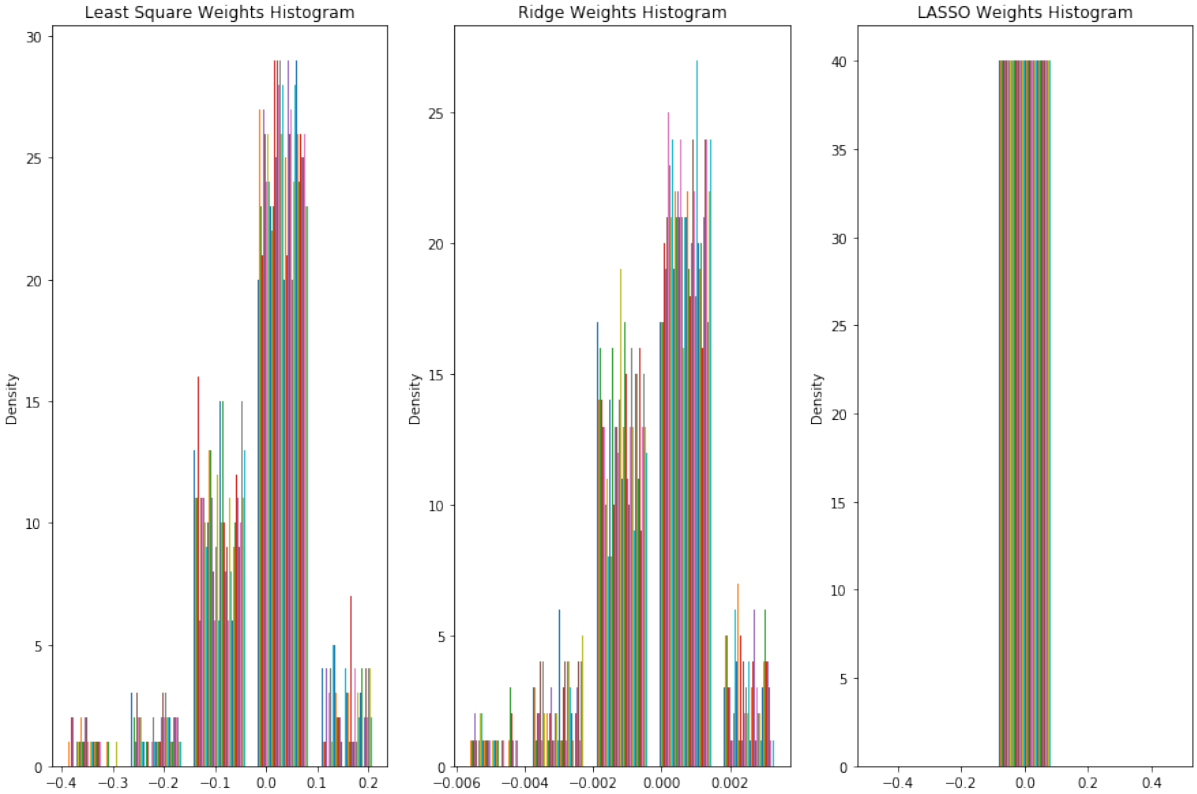
3.1 Notes on Results of the notebook

- Regularization parameter affects Ridge and LASSO regression at scales separated by few orders of magnitude, thus it is not a number to be sought, it's better to describe a model performance as a function of λ.
- Choosing the type of regression is kind of similar to choosing a degree of freedom to be assigned to the configurations of the system.
- Both Ordinary and Ridge regressions learn the data resulting in symmetric weights $J \approx -0.5$ While LASSO regression Tends to break that symmetry, this can be showed in the following graph.
- the OLS and Ridge regression curves are monotonic, while the LASSO curve is different – suggesting an optimal LASSO regularization parameter is $\lambda \approx 10^2$. At this point, the Ising interaction weights J contains only nearest-neighbor terms (which agrees with the previous assumption when the data was generated).

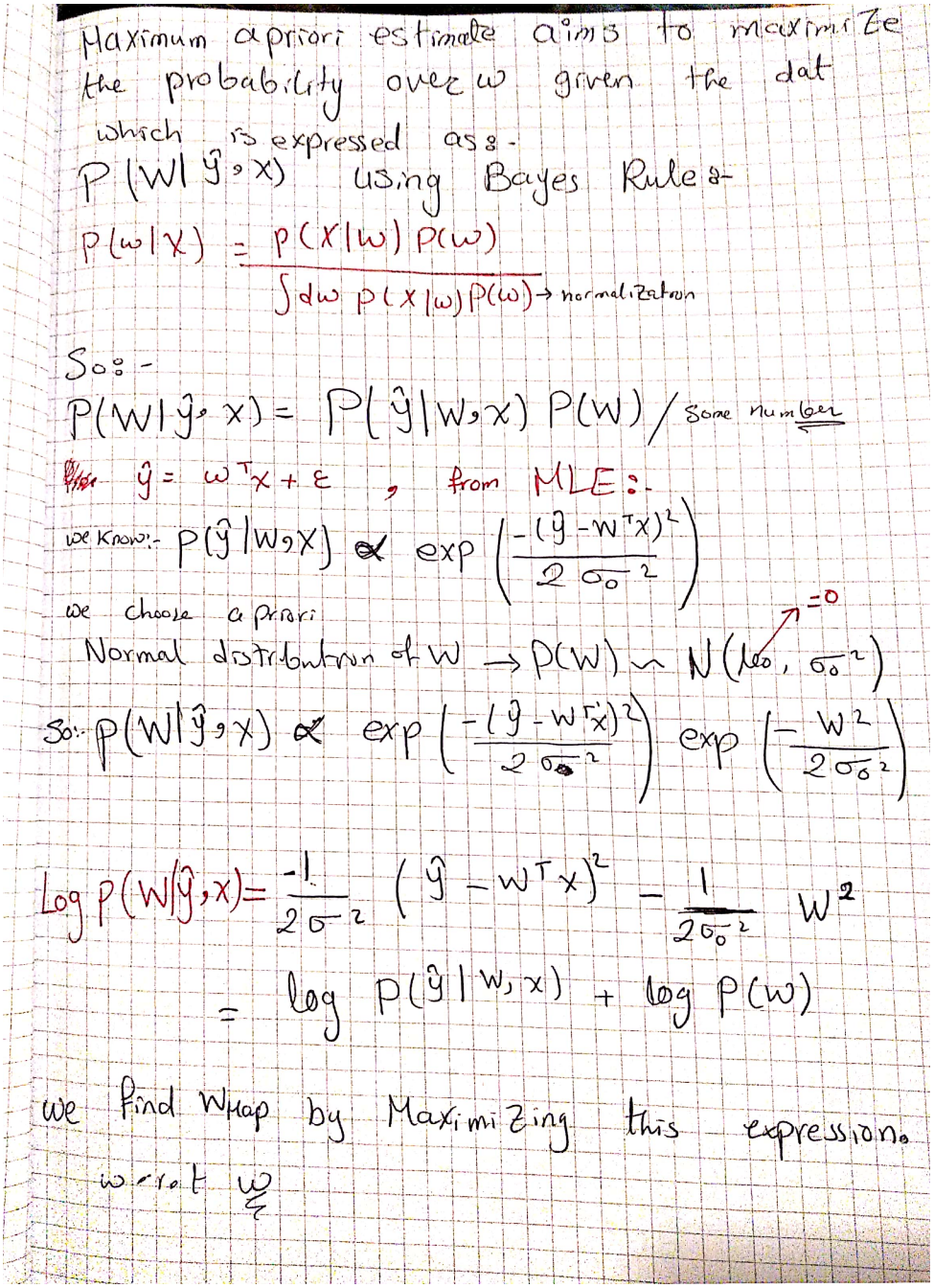


3.1.1 Histogram

The following is a histogram plot of the three resulting "coupling" weights, it shows that for both ordinary and ridge regressions the weights vary with different densities while for LASSO regression there exists a small range of values with the same density. another note on the histogram plot is that these weights are discrete, in other words not forming a continuous spectrum at least in the ridge and OLS regression, but seems a little more continuous in the lasso regression plot.



3.1.2 MAP Estimation



A final coment on the final expression of of the Map estimation, for $\frac{1}{\sigma^2} = \lambda$ and $\sigma_0^2 = 1$ it can be easily recognized to be the for of Ridge regression which is a way of calculation parameters that aims to describe the data without bias or overfitting which a similar aim to the maximum apriori estimation that calculate the parameters to aligne with the haigest pobability of agreement with the data.