THE UNIVERSITY OF HONGKONG
DEPARTMENT OF PHYSICS

Assignment 3

Course: *Machine Learning in Physics (PHYS3151)* – Professor: *Dr. Ziyang Meng*
Due date: *Mar. 14th, 2023*

This assignment is also a project, you will need to deliver a presentation on how you solve the questions as well as your results.

Square lattice Fermi surface away from half-filling

In this assignment, we are going to investigate the affect of symmetry and how it can be used to reduce over-fitting. Use the functions in notebook *SVM_Fermi_Surface.ipynb* with chemical potential μ set to be -2.

In all questions, you may need to try a few times with different random samples of points $\{(k_x, k_y)\}$ to see the effect of randomness.

1. In the sample code, we perform logistic regression and support vector machine with $N = 200$ sample points, and the resulting decision boundary is very close to the actual Fermi surface. Try to repeat the process with $N_{training} = 30$ for both methods, see if the result is as good.
The features used in this part should be $X_1 = [k_x, k_y, k_x^2, k_x^2, k_x * k_y]$, which is the same as what we use in the sample code.
Comment on how the lack of samples affects on these two method. Will they suffer by the same extent or does one fits better than the other in this case?

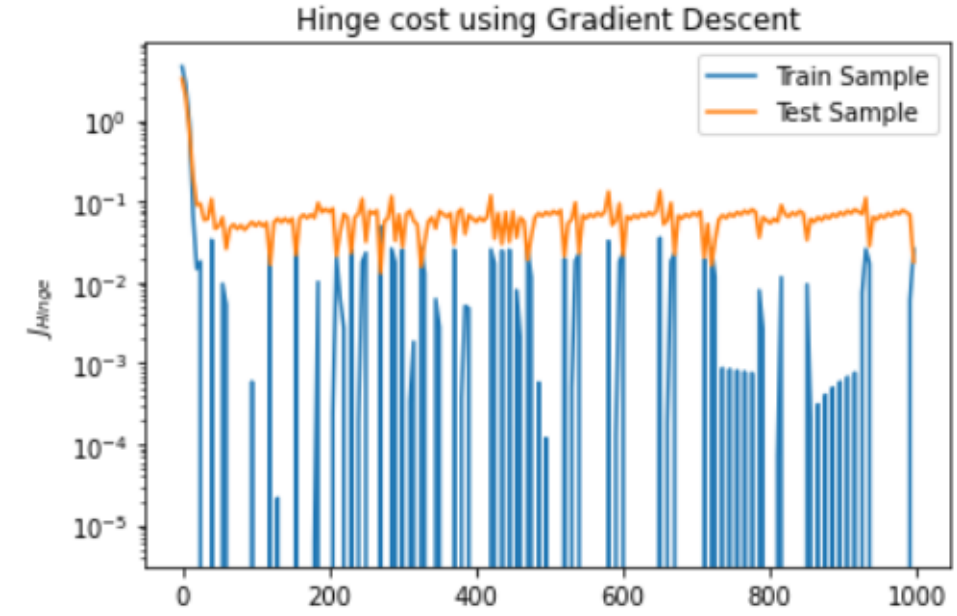
Finish the following parts using SVM only.

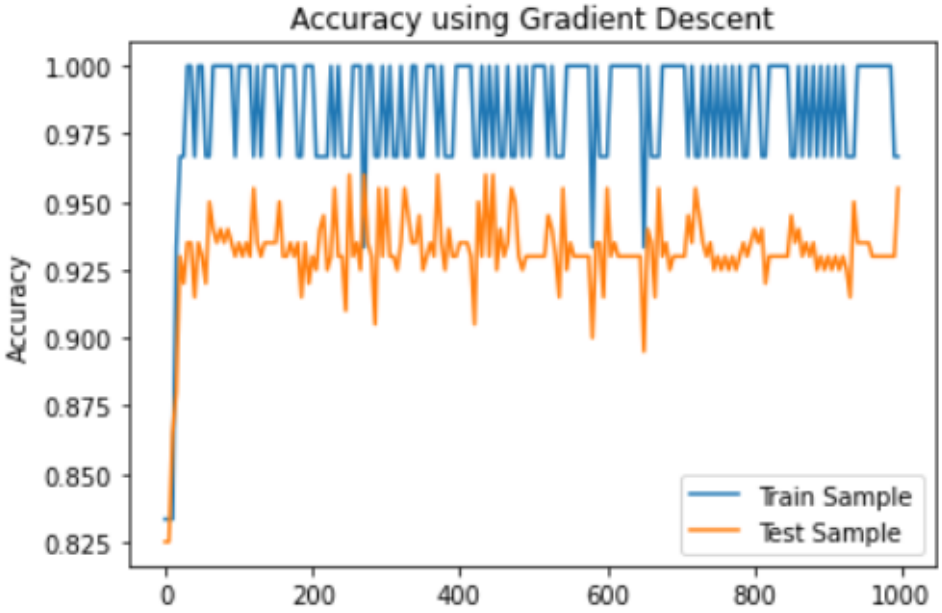
2. Due to the fact that x and y directions are equivalent in this model, the Fermi surface should be symmetric along $y = x$. In other words, the coefficients of Θ in k_x and k_y terms should equal, the same goes for k_x^2 and k_y^2.
We can manually enforce this condition by introducing an symmetric features for the fitting. Consider the following three features $X_2 = [k_x + k_y, k_x^2 + k_x^2, k_x * k_y]$. Perform SVM and compare its performance against that with X_1, with $N = 30$ sample points.
(Hint: You can compare the final costs and resulting decision boundary in the two case.)

3. We proceed to test the **forecastability** for the two fitting with X_1 and X_2.
Besides the 30 sample points used to train the model, construct $N_{test} = 200$ more sample points and calculate their corresponding energy and occupancy. This 200 points should be independent of the 30 samples before. The test samples should **not** be used in training the model, it should be used **only** to evaluate the performance of the models.
During the iteration, besides calculating the overall cost function with the training sample, compute also (i) the Hinge cost, that is the part of cost function without the effect of $\|\omega\|$, and (ii) accuracy, that is the proportion of samples that the model succeed in classifying (You may need to use the *check* function defined in the sample code). These two values need to be calculate for train sample and test sample separately. In the end you will have 4 more *cost_history* lists for each case.
Compare the performance with two sets of features X_1 and X_2 by comparing these values. One typical output for X_1 is shown in Figures below

4. Besides forecastability, are there other advantages of using X_2 comparing to X_1?

```
[[ 7.26272335]
 [-0.0562331 ]
 [-0.71993329]
 [-2.2202115 ]
 [-2.05323987]
 [-0.86417304]]
Final Cost is : 0.1735565705391023
Final Hinge Cost is : 0.0
Final Hinge Cost for test sample is : 0.05728009619032316
Final Accuracy is : 1.0
Final Accuracy for test sample is : 0.94
```


Hinge cost using Gradient Descent
Train Sample
Test Sample
J_{Hinge}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0
200
400
600
800
1000


Accuracy using Gradient Descent
Accuracy
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
Train Sample
Test Sample
0
200
400
600
800
1000