FYS4480 - Quantum Mechanics for Many-Particle Systems

Project 1

Even M. Nordhagen

October 15, 2018

- Github repository containing programs and results:

  `https://github.com/evenmn/FYS4480`

Abstract

The aim of this project is to study the performance of linear regression in order to fit a two dimensional polynomial to terrain data. Both Ordinary Least Square (OLS), Ridge and Lasso regression methods were implemented, and for minimizing Lasso's cost function Gradient Descent (GD) was used. A fourth method was to minimize the cost function of Ridge using GD. The fitted polynomial was visualized and compared with the data, the Mean Square Error (MSE) and R^2-score were analyzed, and finally the polynomial coefficients were studied applying visualization tools and Confidence Intervals (CI). To benchmark the results, we used Scikit Learn.

We found the self-implemented OLS and Ridge regression functions to reproduce the benchmarks, and Lasso was close to reproducing the benchmark as well. However, the difference between results produced by standard Ridge regression and when minimizing its cost function is large. The OLS regression method is considered as the most successful due to its small MSE and high R^2-score.

Contents

1 Introduction

The linear regression methods were first introduced for more than two centuries ago, and have been used in a large number of fields throughout the years [1][2]. In this project we will investigate whether the methods are sufficient for fitting polynomials to real terrain data, or we need more complicated methods. To challenge the methods, we chose terrain data from the volcanic island of Lombok, Indonesia, where the contour lines are quite dense.

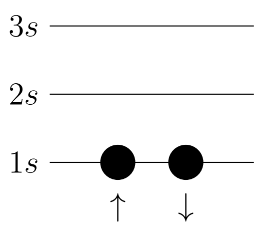
We developed our own software for ordinary least square (OLS), Ridge and Lasso linear regression, where the latter was based on minimization using gradient descent (GD). To verify the implementation, we tested it on data from the Franke function where we knew what the result should be. Further, the error was analyzed in order to decide which method that gave the best result, and all data was resampled using the K-fold validation method to estimate the actual error.

For the results, see section *Results* (5), which again is discussed in section *Discussion* (6). The background theory can be found in section *Theory* (2), and all methods and techniques are presented in the section *Methods* (3). For code structure and implementation, see section *Code* (4), and finally, the conclusion is found in section (7) with the same name.

2 Theory

2.1 The Helium atom

2.1.1 Ground state



2.1.2 Excited states

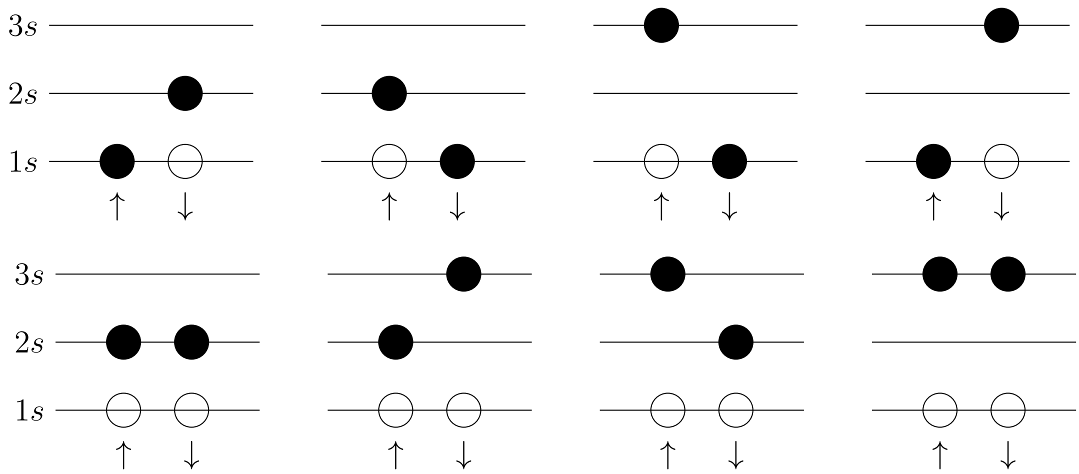


Figure 1

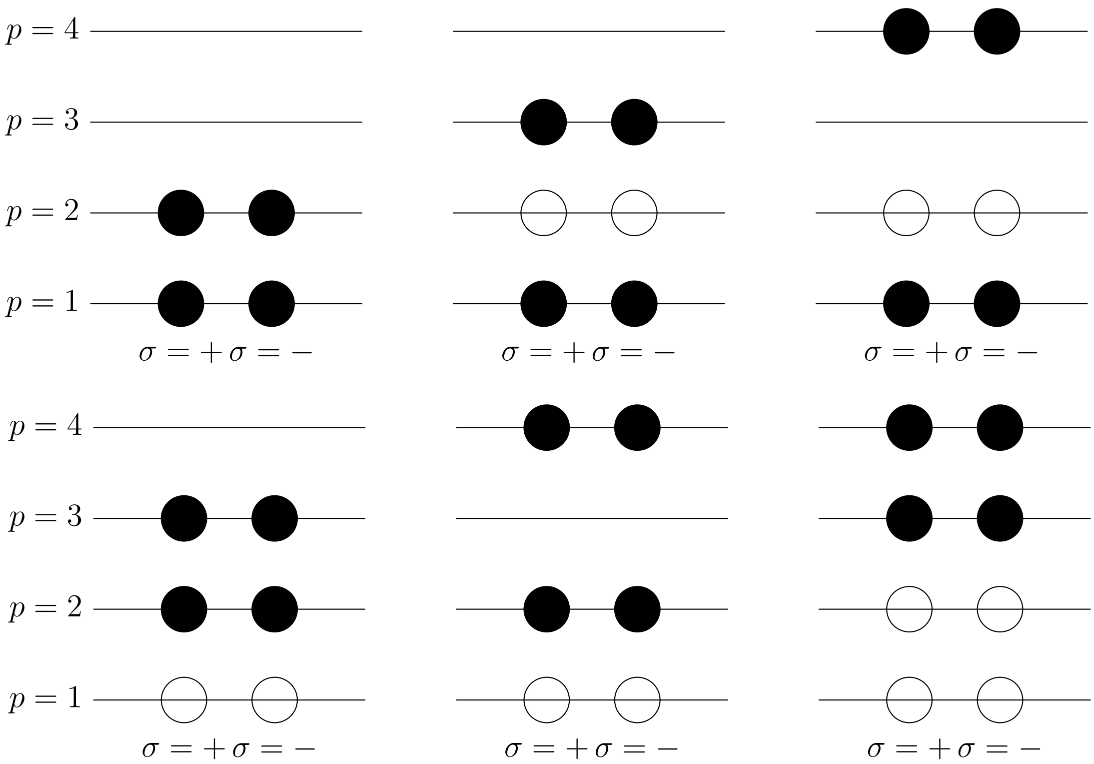


Figure 2: Above all basis states of our pair interaction system are presented schematically with $P = 2$ as the number of pairs and $S_z = 0$ are the total spin. The solid dots indicate occupied states, while the empty dots indicate unoccupied states (holes). The reference state $|\Phi\rangle$ is represented in the upper left corner. This is all possible states since the excusion principle does not allow two particles with same spin to stay at the same level. For further description, see the text.

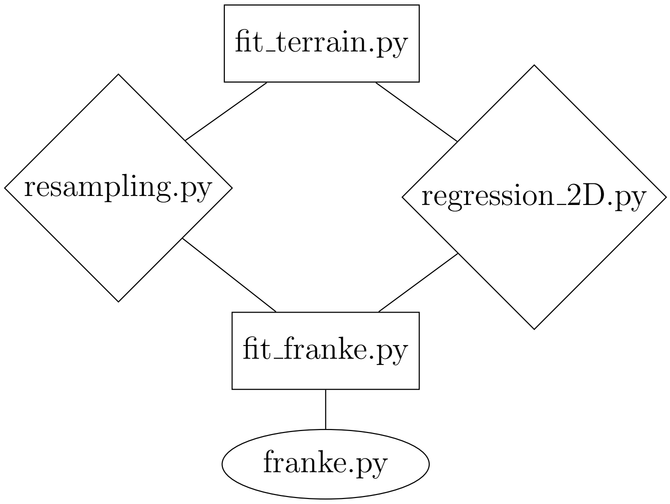


Figure 3: Code structure

2.2 The Beryllium atom

3 Methods

The implementation could look something like this

```
def bootstrap(data, K=1000):
  dataVec = np.zeros(K)
  for k in range(K):
    dataVec[k] = np.average(np.random.choice(data, len(data)))
  Avg = np.average(dataVec)
  Var = np.var(dataVec)
  Std = np.std(dataVec)

  return Avg, Var, Std
```

4 Code

4.1 Code structure

5 Results

Table 1: Mean Square Error and R^2-score presented for OLS, Ridge, Lasso and Ridge + gradient descent (RidgeGD), where noise was added to the data. The parameters used were $\lambda = 1e-5$ (penalty), $\eta = 1e-4$ (learning rate), niter $= 1e5$ (number of iterations) and $\mathcal{N}(0, \sigma^2 = 0.1)$ (noise). See text for more information.

	MSE			**R2**		
	Self	K-fold	Scikit	Self	K-fold	Scikit
OLS	0.008494	0.009119	0.008494	0.9048	0.8956	0.9048
Ridge	0.009128	0.009651	0.009128	0.8977	0.8895	0.8977
Lasso	0.01439	0.01489	0.01555	0.8387	0.8296	0.8257
RidgeGD	0.01451	0.01504	0.009128	0.8373	0.8280	0.8977

6 Discussion

7 Conclusion

8 References

[1] A.M.Legendre. Nouvelles méthodes pour la détermination des orbites des comètes. Libraure pour les Mathématiques. (1805).

[2] C.F. Gauss. Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientum. Hamburg: Frid. Perthes and I.H. Besser. (1809).

[3] T. Hastie, R. Tibshirani, J. Friedman. The Elements of Statistical Learning. Springer-Verlag, New York. (2009).

[4] United States Geological Survey (USGS). https://earthexplorer.usgs.gov/

[5] Lecture notes in Statistical Physics: Statistical Physics - a second course. F. Ravndal, E. G. Flekkœy. (2014).

[6] Simulation stimulation. L. Bastick. https://www.sumproduct.com/thought/simulation-stimulation

[7] Machine Learning Crash Course: Part 4 - The Bias-Variance Dilemma. D. Geng, S. Shih. https://ml.berkeley.edu/blog/2017/07/13/tutorial-4/ (2017).

[8] Regression Analysis: How Do I Interpret R-squared and Assess the Goodness-of-Fit? http://blog.minitab.com/blog/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit (2013).

[9] Bootstrap Methods: Another Look at the Jackknife. B. Efron. Ann. Statist., Volume 7, Number 1, 1-26. (1979).

[10] Lecture notes FYS-STK4155: Regression Methods. https://compphysics.github.io/MachineLearning/doc/pub/Regression/pdf/Regression-minted.pdf M. Hjorth-Jensen. (2018).

[11] Scikit-learn: Machine Learning in Python. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thiron, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay. Journal of Machine Learning Research, Volume 12. (2011).