M 124 Coursework 2

Y. Panagakis and **K. Panousis**

Due Date: April 23, 2021, 23:59

1 Introduction

For the second coursework, we focus on both Classification and Regression.

In the third lecture, we covered the basic approaches to Classification. We started with Logistic Regression which we implemented in the respective tutorial. Then, we introduced Linear and Quadratic Discriminant Analysis and Naive Bayes. Despite its *naive* assumption of feature independence, Naive Bayes constitutes a widely used approach, providing remarkable results. Thus, in the first part of this coursework you are asked to implement Naive Bayes in python and evaluate its performance using a real-life dataset from the UCI Machine Learning Repository.

In the sixth Lecture, we focused on augmenting the simple Linear Regression approach with more sophisticated procedures in order to increase model performance and interpretability. Hence, in the second part, you'll delve deeper into the Regularization approach, focusing on the derivation and implementation of Ridge Regression in Python.

2 Naive Bayes **(60 Points)**

Recall that in Naive Bayes (as was the case with LDA and QDA), we follow an indirect route to classification. Specifically, considering a K-class classification problem, given the feature values for each datapoint we compute:

$$p(y = k|\boldsymbol{X}_n) \propto f_k(\boldsymbol{X}_n)\pi_k \tag{1}$$

where $f_k(x)$ is the density function of x for an observation from the k^{th} class, π_k is the prior probability of class k and $\propto$ is the *proportional to* symbol.

Implementation

Dataset

For this implementation, you'll use a real life dataset from the UCI repository and specifically the *Adult* dataset. You can find the train and test data files here. For this dataset, given some features, we want to predict if a person makes over $50K$ a year. Thus there are two classes: (i) if a person makes $\leq 50K$ and (ii) if a person makes $> 50K$. The dataset contains 48842 instances with 14 features each; a full description can be found in the provided link. The features contain a combination of both continuous as well as discrete components. For the continuous components, use Gaussian distributions as described in the Lecture, while for discrete features use *binning*.

[**Hint**: For each discrete feature i that takes M potential values and each class k, we learn different parameters $\alpha_{i,k,1}, \dots \alpha_{i,k,M}$, while for the continuous components we learn estimates $\mu_{i,k}$ and $\sigma_{i.k}$.]

Sometimes, the datasets in consideration are not ready to be used in any algorithm. For example the dataset may contain missing/corrupted values; if the considered model does not take such peculiarities into consideration, the fitting process will break down. In this case, the dataset contains data points with missing values. Thus, before applying your Naive Bayes implementation, make sure that you remove these points. To this end, simply discard the rows of the data matrix that has any missing values (however this is not always the best practice).

Specifics

Your code must implement the following functionality:

1. **Prior Class Probabilities**: Implement a function to compute the estimated prior class probabilities from the given data.
2. **Parameter Estimation**: A function that implements the Maximum Likelihood Estimation (MLE) for the parameters of the distributions for each feature and class given the training data.
3. **Log-Posterior**: A method to compute the log-posterior $\log f_k(x)\pi_k$ for each different k given some datapoint after training the model.
4. **Prediction**: A method used to make predictions using the Naive Bayes classifier after training the model.

Evaluation

To evaluate the performance of your Naive Bayes implementation train your classifier with the given training data and report the following:

- Parameters estimation results (written to a text file with name nb_parameters.txt):
  - The prior probabilities for each class k, $k = 1, \dots K$.
  - The parameters of the densities $f_k(x)$.
  - For printing to file, report each estimated parameter to a new line.
- Prediction (written to a text file with name predictions.txt):
  - Print in the corresponding file, the predicted label for all points in the test data. For the label coding use 0 if the person makes $\leq 50K$ and 1 otherwise and print one label per line.
- Classification Error (writen to a text file with name classification.txt):
  - Report the classification error for both the training as well as the test data.
- In many ML approaches, the data are usually preprocessed in such a way as to have zero mean and standard deviation of 1. Is this a good idea in general? Explain your thought. Would this preprocessing help the Naive Bayes Classifier? If so, how?

3 Shrinkage Methods: Regularization (40 points)

In the sixth Lecture (Linear Model Selection), we introduced various methods for feature selection. The motivation behind these approaches was that even though linear regression is a very well-known model and easy to implement, it suffers from *simplicity*. We can increase the prediction accuracy and interpretability of a considered model by replacing the conventional Linear Regression approach with more sophisticated methods. To this end, we introduced three different approaches: (i) Best Subset Selection, (ii) Shrinkage/Regularization and (iii) Dimensionality Reduction. In this part of the coursework, we focus on Regularization that constitutes one of the fundamental approaches in Machine Learning.

In this context, we augment the least squares cost function in order to constrain or *regularize* the produced coefficient estimates. This process translates to shrinking the coefficient estimates toward zero and that is why these approaches are also called *Shrinkage* methods. Thus, in this part, you are asked to derive, implement and examine one of the most well known approaches, namely *Ridge Regression*.

Ridge Regression constitutes a variation of the Least Squares estimation approach, where we augment the cost function that we'll optimize. Thus, the optimization problem of RR reads:

$$\arg\min_{\beta} ||\boldsymbol{y}_n - X_n\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_2^2 \tag{2}$$

where $\lambda > 0$ is a non-negative parameter called the *regularization* parameter, and $\boldsymbol{\beta}$ are the coefficients of the model. The regularizing term penalizes the large coefficients and pushes their value towards zero. We consider pairs $(\boldsymbol{x}_n, \boldsymbol{y}_n),\ n = 1, \ldots, N$, where $\boldsymbol{x}_n \in \mathbb{R}^p$ and $\boldsymbol{y}_n \in \mathbb{R}^M$.

Theoretical (6 points)

We start with a mathematical investigation of the RR estimator.

1. Derive the solution of the Ridge Regression optimization problem by hand. Show the intermediate steps.
2. Explain the differences between Least Squares and Ridge Regression, and why RR is more robust to overfitting.
3. We talked about the bias-variance trade-off in Lecture 5. How does the value of λ affect the bias and variance of the estimator?

Implementation of Ridge Regression(34 points)

In this task, you are asked to implement the Ridge Regression Estimator that you derived in the previous section and compare the results with the standard Linear Regression approach. For both implementations you can't use the built-in functions of sklearn (only for sanity-check if you want but they shouldn't be included in the deliverable).

Dataset

For this implementation you'll use a dataset from Hastie et al. [1], corresponding to a prostate cancer study of Stamey et al. [2]. In this study, there are 8 covariates/predictors and we want to predict a prostate-specific antigen. The given predictors are:

- lcavol: the log cancer volume

- lweight: the log-prostate weight
- age: the age of the patient
- lbhp: log-amount of benign hyperplasias
- svi: seminal vesicle invasion
- lcp: log-capsular penetration
- glean: the Gleason Score
- pgg45: the percent of Gleason scores 4 or 5
- lpsa: the response variable, log-psa

The dataset is already split to training and test sets and you can find the respective files in the corresponding e-class folder.

Implementation

Your code must have the following functions:

- fit($\boldsymbol{X}, \boldsymbol{y}, \lambda$): This function will implement the basic fit procedure for the Ridge Regression Estimator. It takes as arguments the data and the regularization parameter and returns the coefficient estimates.
- predict($\boldsymbol{X}, \boldsymbol{\beta}$): Takes as input the data covariates and provides predictions using the coefficient estimates $\boldsymbol{\beta}$.

Further Investigation

For the given dataset and using your implementation:

1. Fit the model using multiple values for the shrinkage parameter λ. To this end, consider a range of potential values $\lambda \in [0.002, 2]$ with a step size of 0.004 and a large value for lambda, e.g. 10000. Plot the corresponding coefficient estimates (y-axis) to the lambda value (x-axis). Explain the produced plot. What happens when we change the regularization parameter?
2. Plot the Root Mean Squared Error on the test set to the regularization parameter. Investigate the plot and find the best value for the regularization parameter.
3. For the optimal value of λ that you found above, plot the predictions (y-axis) versus the real value of the target variable (x-axis) for some datapoints in the test set. Specifically, select 15 points and plot the values.
4. Fit a simple Linear Regression model. To this end, use your implementation from the previous coursework and compare the obtained coefficient estimates, statistics and predictions with the results using Ridge Regression with the optimal λ. How does Linear Regression fare against Ridge Regression.

For all questions, save the figures from matplotlib and include them in your report along with your observations.

General Notes

- In this coursework, you are asked to implement basic approaches to ML problems. As was the case in the first coursework, I highly recommend to take your time and understand the fundamental techniques and rationale behind the introduced approaches and avoid searching ready-made solutions online.
- Individual Coursework.
- Must be submitted via e-class. E-mailed submissions will not be accepted.
- Programming:
  - For the programming part, you can turn to StackOverflow if you bump into some kind of bug or problem, but again I strongly advise against copying code for these simple tasks.
  - As mentioned in the first tutorial, it is best practice to use Virtual Environments for your projects to avoid breaking the system. You can use either Virtual Environments of native python venv or install an Anaconda/Miniconda platform. Or simply use Google Colab.
  - Implement your approach using a Jupyter Notebook, with sufficient but not redundant comments.
- Typesetting:
  - I would suggest using LaTeXfor reporting the results. LaTeXis a very powerful typesetting system, most commonly employed in research papers and reports. It provides all the necessary tools for easily typesetting equations, arrays and more, while avoiding the hassles of other editors. This CS was written in LaTeX. You can use Overleaf for online editing.
  - A template will be provided.

References

[1] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[2] T. A. Stamey, J. N. Kabalin, M. Ferrari, and N. Yang. Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate. IV. Anti-androgen treated patients. *J Urol*, 141(5):1088–1090, May 1989.