Linnaeus University
Introduction to Machine learning, 2DV516
Jonas Lundberg, Jonas Nordqvist
`jonas.lundberg@lnu.se, jonas.nordqvist@lnu.se`

Assignment 2: Linear and logistic regression

Introduction

In this Assignment you will use Python to handle a number of exercises related to gradient descent, linear regression, and logistic regression. The datasets used in the assignment can be downloaded in Moodle. The exercises are further divided into lectures. That is, for example, the first set of exercises related to regression are suitable to handle after Lecture 4.

Submission instructions

All exercises are individual. We expect you to submit at least one py- file (or Jupyter Notebook) for each exercise and your submission should include all the datasets and files we need to run your programs. Execises A and B are just to get you started and should not be submitted. When grading your assignments we will in addition to functionality also take into account code quality. We expect well structured and efficient solutions. Finally, keep all your files in a single folder named as `username_A2` (e.g. cb223pg_A2) and submit a zipped version of this folder.

Certain quantitive questions such as: *What is the expected benchmark result for a certain graphics card?* in Exercise 1, can simply be handled as a print statement in the program. More qualitative questions such as: *Motivate your choice of model.*, should be handled as a comment in the notebook or in a separate text file. (All such answers can be grouped into a single text-file.) The non-mandatory VG-exercise will require a separate report.

Linear and polynomial regression (Lecture 4)

Exercise A: Not to be submitted!!!

In order to get started we suggest that you recompute the results related to the `girls_height.csv` dataset presented in the Lecture 4 slides. Column 1 is the girl height and columns 2 and 3 are the mom and dad heights. Recomputing the results will allow you to verify that your implementation of the Normal Equation, Cost function, Feature normalization, and Gradient descent works. Adapting this code to handle Exercises 1 and 2 should be rather straight forward if your implementation is properly vectorized. In short:

1. Plot the dataset
2. Compute the extended matrix $X_e = [\mathbf{1}, X_1, X_2]$
3. Implement the Normal Equation $\beta = (X_e^T X_e)^{-1} X_e^T y$. A girl with parent heights (65,70) should have a predicted height $X_e\beta$ of 65.42 inches.
4. Apply Feature Normalization $X_n = (X - \mu)/\sigma$ and plot the dataset once again to verify that the mom and dad heights are centered around 0 with a standard deviation of 1.

5. Compute the extended matrix X_e and apply the Normal equation on the normalized version of (65.70). The prediction should still be 65.42 inches.

6. Implement the cost function $J(\beta) = \frac{1}{n}(X_e\beta - y)^T(X_e\beta - y)$ as a function of parameters X_e, y, β. The cost for β from the Normal equation should be 4.068.

7. Gradient descent $\beta^{j+1} = \beta^j - \alpha X_e^T(X_e\beta^j - y)$.

   (a) Implement a vectorized version of gradient descent

   (b) Find (and print) suitable hyperparameters α, N. Remember to start with a small α (say 0.001) and N (say 10) to make sure that the cost is decreasing. A plot J vs Iterations is an excellent way to see that J is decrasing as expected. Then gradually decrease α, and increase N, to find a suitable pair of (α, N) that rapidly decreases/stabilizes the cost at its minimum (4.068).

   (c) Verify that the predicted height for a girl with parents (65.70) is still 65.42 inches.

Exercise 1: Multivariate regression

In this exercise you will use the dataset `GPUBenchmark.csv`. The first six columns (X) are various data related to graphic cards. For example, number of cores and clock speed. See the excel file `NvidiaGPU_Speed.xlsx` for more details. The 7th column is the response y. In this case the result of testing the graphic card in a certain benchmarking program. The dataset contains 18 observations. The main objective is to find the hypothesis $f(X) = \beta_0 + \beta_1 X_1 + \ldots + \beta_6 X_6$, which estimates the linear relation between the graphic card properties and the benchmark result.

1. Start by normalizing X using $X_n = (X - \mu)/\sigma$.

2. Multivariate datasets are hard to visualize. However, to get a basic understanding it might be a good idea to produce a plot X_i vs y for each one of the features. Use `subplot(2,3,i)` to fit all six plots into a single figure. Make sure that each nomalized X_i is centralized around zero.

3. Compute β using the normal equation $\beta = (X_e^T X_e)^{-1} X_e^T y$ where X_e is the extended normalized matrix $[1, X_1, \ldots, X_6]$. What is the predicted benchmark result for a graphic card with the following (non-normalized) feature values?

   $$2432,\ 1607,\ 1683,\ 8,\ 8,\ 256$$

   The actual benchmark result is 114.

4. What is the cost $J(\beta)$ when using the β computed by the normal equation above?

5. Gradient descent

   (a) Find (and print) hyperparameters (α, N) such that you get within 1% of the final cost for the normal equation.

   (b) What is the predicted benchmark result for the example graphic card presented above?

Exercise 2: Polynomial regression

The data for this exercise is found in `housing_price_index.csv`. The data consists of housing price index for houses in Småland from the year 1975 to 2017. The objective is to predict the pricing index using only the year.

1. Plot the data in the matrix `housing_price_index`.

2. As you probably notice the relationship among the variables doesn't seem to be linear. Try to fit (and plot using `subplot(2,2,i)`) all polynomial models $f(X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \ldots + \beta_d X^d$ for degrees $d \in [1, 4]$. Which polynomial degree do you think gives the best fit? Motivate your answer!

3. Jonas Nordqvist bought in 2015 a house in Växjö for 2.3 million SEK. What can he expect to get, using your "best fit model", for his house when he (after completing his PhD) sells his house in 2022 (to start his new career as a data scientist in Stockholm)? Is your answer realistic?

Logistic regression (Lecture 5)

Exercise B: Not to be submitted!!

We once again suggest that you to get started by recomputing the results for the `admission.csv` dataset presented in the lecture slides since it will give you a chance to verify that your logistic regression classifier implementation works before you start to work with other datasets.

1. Start by normalizing the features ($X_n = (X - \mu)/\sigma$) and then plot the 2D data using different markers for for the two labels (Admitted, not Admitted) to verify that both features are centered around 0 with a standard deviation of 1.

2. Implement the sigmoid function. The function should take any matrix as input and output a matrix of the same size where you apply the sigmoid function on each element. Test your function using the 2×2 matrix `[[0,1],[2,3]]` you should get the following output: `[[0.5000,0.7311],[0.8808,0.9526]]`

3. Extend X to make it suitable for a linear assumption $y = X\beta = \beta_0 + \beta_1 X_1 + \beta_2 X_2$.

4. Implement a vectorized version of the logistic cost function
$$J(\beta) = -\frac{1}{n}(y^T \log(g(X\beta)) + (1-y)^T \log(1 - g(X\beta))).$$
The cost for $\beta = [0, 0, 0]$ is 0.6931.

5. Implement a vectorized version of gradient descent
$$\beta^{j+1} = \beta^j - \frac{\alpha}{n} X^T (g(X\beta) - y).$$
Starting with $\beta^0 = [0, 0, 0]$, using $\alpha = 0.5$, then after one iteration we have $\beta^1 = [0.05, 0.141, 0.125]$ and a reduced cost $J = 0.6217$.

6. Increasing the number of iterations N the cost will eventually stabilize at $J = 0.2035$ resulting in $\beta = [1.686, 3.923, 3.657]$. Plot also the linear decision boundary

7. The admission probability for a student with scores 45, 85 is 0.77, and the number of training errors is 11.

Exercise 3: Multivariate Logistic Regression

You will now try to classify woman breast cancer tumours. This dataset `breast_cancer.csv` contains 683 observations and has 9 features and (in column 10) binary labels of either benign (2) or malignant (4). A full description of the dataset is available is this note[1] (note that the ID has been removed from the original data).

[1]https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.names

1. Read data and shuffle the rows in the raw data matrix:

```
data = ... load breast_cancer.csv
np.random.shuffle(data)                    % Shuffle rows in Python
```

2. Replace the responses 2 and 4 with 0 and 1 and divide the dataset into a training set and a test set. How many observations did you allocated for testing, and why this number?

3. Normalize the training data and train a *linear logistic regression model* using gradient descent. Print the hyperparameters α and N_{iter} and plot the cost function $J(\beta)$ as a function over iterations.

4. What is the training error (number of non-correct classifications in the training data) and the training accuracy (percentage of correct classifications) for your model?

5. What is the number of test error and the test accuracy for your model?

6. Repeated runs will (due to the shuffling) give different results. Are they qualitatively the same? Do they depend on how many observations you put aside for testing? Is the difference between training and testing expected?

Exercise 4: Nonlinear logistic regression

For this exercise you will once again work with the microchip dataset `microchips.csv` that we used in Assignment 1.

1. Plot the data in X and y using different symbols or colors for the two different classes. Notice also that X_1 and X_2 are already normalized. Hence, no need for normalization in this exercise

2. Use gradient descent to find β in the case of a quadratic model.

$$X\beta = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_1 X_2 + \beta_5 X_2^2.$$

   Print the hyper parameters α and N_{iter}, and produce a 1×2 plot with: 1) the cost function $J(\beta)$ as a function over iterations, 2) the corresponding decision boundary (together with the X, y scatter plot), and 3) the number of training errors presented as a part of the decision boundary plot title.

3. In this the final part of this exercise and upcoming exercises we will consider polynomial expressions of the features in logistic regression. Implement a method called `mapFeatures`. That is a function that takes two features X_1, X_2 and a degree d as input and outputs all combinations of polynomial terms of degree less than or equal to d of the variables X_1 and X_2. Suggestions for how to do it are presented in the lecture slides.

4. Use `mapFeatures` to repeat 2) but with a *polynomial of degree five* ($d = 5$) model.

Accuracy, regularization and model selection (Lecture 6)

Exercise 5: Regularized logistic regression

For this exercise we will go back to the microship dataset `microchips.csv` used in Exercise 4. We will revisit logistic regression but now instead use `sklearn`.

1. Use `Logistic regression` and `mapFeatures` from the previous exercise to construct nine different classifiers, one for each of the degrees $d \in [1, 9]$, and produce a figure containing a 3×3 pattern of subplots showing the corresponding decision boundaries. Make sure that you pass the argument `C=10000`.[2]

2. Redo 1) but now use the regularization parameter $C = 1$. What is different than from the step in 1)?

3. Finally, you should use cross-validation (in `sklearn`) to see which of the regularized and unregularized models performs best. The results could for instance be visualized in a graph where you plot the degree d vs. #errors, and differentiate regularized and unregularized by color.

Exercise 6: Forward selection

In this exercise we will go back to the dataset used in Exercise 1, `GPUbenchmark.csv`. On this dataset we will use *forward selection* in order to obtain a model of (eventually) fewer features.

1. Implement the forward selection algorithm as discussed in Lecture 6 (see lecture notes for details). In the loop use the training MSE to find the best model in each iteration. The algorithm should produce $p + 1$ models $\mathcal{M}_0, \ldots, \mathcal{M}_p$, where $\mathcal{M}_i$ is the best model using i features. In terms of output, an alternative could be to let the algorithm produce a p-dimensional vector where its first entry is the feature in $\mathcal{M}_1$, its second entry is the new feature in $\mathcal{M}_2$ etc.

2. Apply your forward selection on the `GPUbenchmark.csv`. Use 3-fold cross-validation to find the best model among all $\mathcal{M}_i$, $i = 1, \ldots, 6$. Which is the best model? Which is the most important feature, *i.e.* selected first?

Exercise 7: Insurances and regularization techniques (VG-exercise)

This exercise is optional for passing the assignment, but required to obtain grades A or B.

In this exercise you will investigate a regression problem. The data is found in `insurance.csv`, and the goal is to predict insurance *charges* given certain traits of the policyholders. Throughout the exercise you may utilize `sklearn`.[3]

The objective is to find a good linear regression model (you should determine on your own what can be considered a relevant way to measure which model is best among your candidates). In your comparison you should at least test the following variants of linear regression: standard linear regression, lasso regression, ridge regression and elastic net regression. Extensions and further work can be for instance optimizing for the regularization hyperparameter λ (this is called `alpha` in `sklearn`), adding transformed features (*e.g.* polynomial features), combining transformed features and regularization. Note that several of the variables in the dataset are categorical variables, and you'll need to figure out a way to treat these variables properly.

Your solutions should be accompanied by a short report (preferably a pdf-document), stating which was the best model, and how you decided which was the best model. The report should be self-contained from the code, and the reader should fully understand what you've done from reading the report. To be granted the higher grade from this exercise you will have investigate several models, and compare them in a systematic and correct way in order to deduce which model is the best. Besides this your code should be well-structured and well-written.

[2] $C = 1/\lambda$ as the regularization parameter from the lecture

[3] Use the online documentation to read about necessary tools for the assignment.