INF264 - Exercise 2

August 2019

1 Instructions

- Deadline: 6/09/2019 , 23:59
- Submission place: https://mitt.uib.no/courses/19532/assignments
- Format: Your answers are to be returned in a single pdf report. You can also return scanned pages for your calculations. For results, your answers must include any values and plots that are requested in the Notebook.

2 Univariate Linear Regression

2.1 Linear regression in closed form

First load the dataset "unilinear.csv".

In the first part of the exercise, you are required to implement the closed form of linear regression which was taught in the lecture, the solution to which is as below:

$$\hat{w} = (X^T X)^{-1} X^T y. \tag{1}$$

In the equation (1), X is the matrix of input vectors and y is the outputs. Once you solve the normal equation, the vector $\bar{w}$ will contain the parameter estimates (w_0 and w_1) of the line.

Your task is to implement the equation (1) and answer the following questions:

- what values does the vector $\bar{w}$ contains?
- Plot the estimated line function. How does this function fit to the data points?

2.2 Gradient descent

In this section, you will learn to implement gradient descent. In many of the real world scenarios, the X matrix maybe too large therefore it is not possible for normal computation powers to run the inversion procedure. Therefore, in

these scenarios the gradient descent method is a well-suited solution. Answer the following questions:

- What is the answer to the following partial derivatives? Calculate them by hand and write your method step by step in the pdf file.

$$\frac{\partial MSE}{\partial w_0} = \frac{\partial}{\partial w_0}\left(\frac{1}{n}\sum_{i=1}^{n}[y_i - (w_1 x_i + w_0)]^2\right), \tag{2}$$

$$\frac{\partial MSE}{\partial w_1} = \frac{\partial}{\partial w_1}\left(\frac{1}{n}\sum_{i=1}^{n}[y_i - (w_1 x_i + w_0)]^2\right). \tag{3}$$

- After finding the derivatives, implement the equations in the python code section provided in the Notebook. Then update the (w_0, w_1) according to the following: (η is the learning rate.)

$$(w_0, w_1)_{t+1} = (w_0, w_1)_t - \eta\left(\frac{\partial MSE}{\partial w_0}, \frac{\partial MSE}{\partial w_1}\right). \tag{4}$$

  Repeat this procedure for 40 iterations with $\eta = 0.1$. Write the resulting (w_0, w_1).

- Repeat the last part only this time with $\eta = 1$. Explain what is the results and why?

- Compare your implementation with sklearn version of the linear regression on a plot for each previous question. Did you get the same answer? How was your implementation different in practice? Plot both lines.

3 Multivariate Regression

3.1 Multiple variables

In many of the real world machine learning problems the input of model has a set of independent variables in the feature vector instead of a single variable. In this scenario, the dependent variable is modeled as a linear combination of multiple independent variables.

First, you need to load the Psychology Grades dataset and then fit a multivariate linear regression using sklearn to this dataset.

- What are the intercept and coefficients learned by the model for each variable?

After fitting the model, write a code that answers the following question:

- What grade would a student get if they get 50, 60, and 70 on the three exams, respectively?

3.2 Basis functions

Sometimes the function that you are trying to estimate is not in linear format e.g., a polynomial function of degree 2. The data points corresponding to an none linear function are given in "nonlinear.csv" and "nonlinear_val.csv".

These datapoints are split into two parts of training and validation sets. Fit a linear regression model on the training datapionts.

- What is the Mean Square Error on the training and the validation sets? Plot the resulting function on training and validation sets.

Next, implement the RBF formula according to

$$RBF(x) = e^{\frac{-(x-c)^2}{\epsilon}} \tag{5}$$

Then, transform your datapoints to a new 11-dimesional space according to multiple RBFs with centers at (-5,-4,...,4,5) and $\epsilon = 0.1$.

- What is the MSE of fitting a Linear regression to the RBF transformed training datapoints, and the validation set? Plot the resulting function on training and validation sets.

3.3 Regularization

Repeat the last part but this time put your centers at (-5,-4.9,-4.8,...,4.8,4.9).

- Now calculate the MSE on both sets and plot the function same as before. What is different about the model?

This is refred to as "overfitting". Model overfitting often happens when you have lots of features, and too little data per feature. In order to solve this problem, regularization is used. In regularization, we add a penalty term to the loss function. This penalty term prevents the model from overfitting. Use Ridge penalty in order to stop the model from overfitting.

- plot the results and compare it with the overfitted function. Also compare the MSE and plots with previous section on both training and validation sets.

4 Logistic Regression

Logistic Regression (also called Logit Regression) is a generalized linear model which is commonly used to estimate the probability that an instance belongs to a particular class (e.g., what is the probability that this email is spam?). If the estimated probability is greater than 0.5, then the model predicts that the instance belongs to that class (called the positive class, labeled "1"), or else it predicts that it does not (i.e., it belongs to the negative class, labeled "0"). This makes it a binary classifier.

Let's use the iris dataset to illustrate Logistic Regression. This is a famous dataset that contains the sepal and petal length and width of 150 iris flowers of three different species: Iris-Setosa, Iris-Versicolor, and Iris-Virginica.

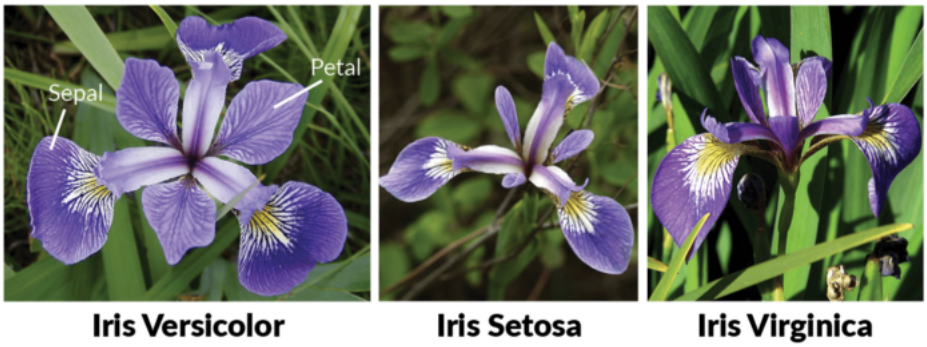
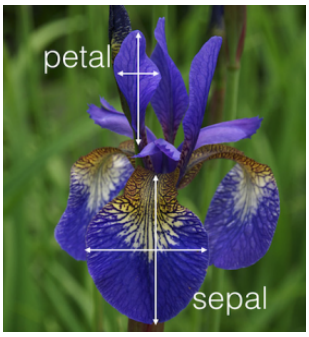


Figure 1: Caption



4.1 Logistic regression on two classes

In order to try logistic regression on two classes you need to divide the Iris dataset into two classes of Iris-Virginica and not Iris-Virginica. then, fit a linear regression model to the petal width of the Iris-Virginica (Read data DESC for more information). Then, fit a logistic regression to the same feature.

- compare the two models on plots.

4.2 Multi class logistic regression

Your task now is build a multi-class classifier that can distinguish between Iris-Virginca, Iris-Setosa, and Iris-Versicolor. Furthermore, instead of using one feature, now you have to use two features — petal length and petal width — to train your model.

- Plot the boundaries for predicting the labels of the dataset. How is it different than the one that was in the previous week's assignment for KNN?

Read more about the Logistic Regression parameters in the online document. Understanding C, solver, and multi_class is important for this assignment.