Instituto Superior Técnico

Departamento de Engenharia Electrotécnica e de Computadores

Machine Learning

1st Lab Assignment

Shift: Thursday Group Number: 21

Number 80966 Name Francisco Azevedo

Number 81356 Name Duarte Dias

Linear Regression

Linear Regression is a simple technique for predicting a real output *y* given an input $\mathbf{x}=(x_1, x_2,\ldots, x_P,)$ via the linear model

$$f(x) = \beta_0 + \sum_{k=1}^{P} \beta_k x_k$$

Typically there is a set of training data $T=\{(\mathbf{x}^i, y^i), i=1,.., N\}$ from which to estimate the coefficients $\boldsymbol{\beta}=[\beta_0, \beta_1, \ldots,\beta_P]^T$. The Least Squares (LS) approach finds these coefficients by minimizing the sum of squares error

$$SSE = \sum_{i=1}^{N} (y_i - f(x^i))^2$$

The linear model is limited because the output is a linear function of the input variables x^k. However, it can easily be extended to more complex models by considering linear combinations of nonlinear functions, $\phi_k(x)$, of the input variables

$$f(x) = \beta_0 + \sum_{k=1}^{K} \beta_k \phi_k(x)$$

In this case the model is still linear in the parameters although it is nonlinear in **x**. Examples of nonlinear function include polynomial functions and Radial basis functions.

This assignment aims at illustrating Linear Regression. In the first part, we'll experiment linear and polynomial models. In the second part, we'll illustrate regularized Least Squares Regression. The second part of this assignment requires MatLab's Statistics Toolbox.

1. Least Squares Fitting

1. Write the matrix expressions for the LS estimate of the coefficients of a polynomial fit of degree P and of the corresponding sum of squares error, from training data $T=\{(x_i, y_i), i= 1,.., N\}$.

LS estimate of the coefficients

$$\beta = (X^T X)^{-1} X^T y$$

Corresponding SSE:

$$SSE(\beta) = [(\beta^T X - y)^2]$$

2. Write Matlab code to fit a polynomial of degree P to 1D data variables *x* and *y*. Write your own code, do not use any Matlab ready made function for LS estimation or for polynomial fitting. You should submit your code along with your report.

3. Load the data in file 'data1.mat' and use your code to fit a straight line to variables *y* and *x*.

    a. Plot the fit on the same graph as the training data. Comment.

The obtained graphs correspond to what was expected: the straight line fits to the data in order to minimize the sum of squares error

NOTE: all images are attached in the ZIP folder sent together with this report. We labelled them with the number of the exercise (e.g.: for this exercise the label is "Part1_3a_fit"). Also code for Part1 is in "code1.m", code for Part2 is in "regularization.m".

    b. Indicate the coefficients and the error you obtained.

$$\beta_0 = 0.9312$$
$$\beta_1 = 1.734$$
$$\beta_2 = 0.0088$$
$$Error\_SSE = 36.05$$

4. Load the data in file 'data2.mat', which contains noisy observations of a cosine function $y = \cos(2x) + s$, with $x \in [-1,1]$, in which s is Gaussian noise with a standard deviation of 0.15. Use your code to fit a second-degree polynomial to these data.
    a. Plot the training data and the fit. Comment.

The obtained graph ("Part1_4a_fit") shows that the second-degree polynomial forms a curve similar to a cosine. Despite the Gaussian noise, the model was able to fit the curve to the data and show the underlying signal. We conclude that due to Gaussian Noise having zero mean the model used to fit the data is able to filter that and show the original signal.

    b. Indicate the coefficients and the error you obtained. Comment.

$$\beta_0 = 0.9757$$
$$\beta_1 = -0.0257$$
$$\beta_2 = -1.532$$
$$Error\_SSE = 19.57$$

5. Repeat item 4 using as input the data from file ‘data2a.mat’. This file contains the same data used in the previous exercise except for the presence of an outlier point.

    a. Plot the training data and the fit. Comment.

The graph obtained shows a subtle difference as the outlier “pulled” the graph towards that point (which can be seen by the increase of the maximum value of Y highlighted in the images “Part1_4a_fit” and “Part1_5a_fit”). This shows the influence a single outlier point has on fitting a model to a limited number of data.

The error obtained also increased meaning that the new fit to the data with the outlier is a worse fit to the data in general in order to decrease the error to the outlier (which remains big). The curve still resembles a cosine but the shape changed slightly.

    b. Indicate the coefficients and the error you obtained. Comment.

$$\beta_0 = 1.052$$
$$\beta_1 = -0.0716$$
$$\beta_2 = -1.631$$
$$Error\_SSE = 26.86$$

2. Regularization

The goal of this second part is to illustrate linear regression with regularization, we’ll experiment with Ridge Regression and Lasso.

1. (T) Write the expression for the cost function used in Ridge Regression and Lasso and explain how Lasso can be used for feature selection.

$$Ridge = \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$
$$Lasso = \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Lasso can be used for feature selection by discarding the feature that for a certain lambda grants the respective coefficient to zero. Ridge uses a quadratic penalty function while Lasso uses a modular penalty, this means that Lasso penalizes irrelevant data to the point of discarding them (making coefficients exactly zero).

2. Load the data in file 'data3.mat' which contains 3-dimensional features in variable **x** and a single output *y*. One of the features in **x** is irrelevant. Use function `lasso` with default parameters (type help for more information on this function) and obtain regression parameters for different values of the regularization parameter λ (the values for lambda are returned in `FitInfo.Lambda`). Use function `lassoPlot` to plot the coefficients against λ. For comparison plot the LS coefficients in the same figure (λ = 0).

```
[B,FitInfo] = lasso(x,y);
lassoPlot(B,FitInfo,'PlotType','Lambda','XScale','log');
```

3. Comment on what you observe in the plot. Identify the irrelevant feature.

In the plot we observe the coefficients fit plotted against lambda resulting from the LassoPlot. By analysing the plot we can conclude that for λ > 0.069478 the second feature is irrelevant given that the respective coefficient is zero (B2 = 0 in the image "Part2_3_coefficients_fit").

4. Choose an adequate value for λ. Plot *y* and the fit obtained for that value of λ. Compare with the LS fit. Compute the error in both cases. Comment.

In this case we choose the value of λ = 0.069478.

$$Error\_LASSO = 15.719$$
$$Error\_LS = 14.982$$

We conclude that by using LASSO our error increases to prevent overfitting on the data. This is done by eliminating a feature from the initial data.

5. Repeat the previous items but using ridge regression (function `ridge)` instead of Lasso. Use the same λ values as in Lasso.

We observe that the coefficients decrease with an increase of lambda using ridge regression as highlighted in "Part2_5_coefficients_fit_ridge"). We also calculated the error for the same value of lambda:

$$Error\_ridge = 16.122$$

We conclude that ridge regression results in a worse fit than LASSO (higher error) but is an alternative to it due to not selecting features, only shrinking them.