INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

DEPARTMENT OF ELECTRICAL ENGINEERING

AN ASSIGNMENT
ON

”THE EFFECTIVENESS OF LINEAR REGRESSION”

Submitted by

Sunaina Saxena	**213070001**
Nihar Mahesh Gupte	**213070002**
Harsh Diwakar	**213070018**
Mohit Kumar Meena	**213070021**

Under the guidance of

Preethi Jyothi
Assistant Professor
Department Of Computer Science And Engineering
IIT Bombay

***Abstract*—This assignment focuses on developing a Machine Learning model to predict a real valued FRP (Fire Radiative Power) of recently raging forest fires in Turkey, based on a dataset with fourteen different attributes describing fires in Australia. In this model designing, first we will extract the attributes which are mainly affecting the FRP and then we will train the Linear Regression Model to predict the FRP for ongoing or upcoming forest fires in Turkey or in any other part of the world.**

I. Introduction

Linear regression is the model that assumes a linear relationship between the input variables (x) and the single output variable (y). In this assignment we learned about various linear regression techniques through the given problem of prediction of FRP. The predictions that we made are discussed in this report along with the results and approaches applied on given data-set. We have also implemented the basis functions and the results are drastically improved by introducing it. Irrelevant features such as version, instruments, $acqdate$ and $acqtime$ are dropped since these features does not contribute in predicting the unknown values. Early stopping algorithm is also implemented that is used stop the gradient descent from further hyperparameter tuning when it finds that the validation loss starts increasing.

II. Solving Linear Regression

Linear Regression is simple but very powerful model that provides interpretations of how inputs affect the final predicted outputs.
$\boldsymbol{F(x,w) = w^T x + w_o} \implies LinearRegression$
where, $x \implies features$, $w \implies parameters$

A. Closed form/Analytical solution

When we have 'N' number of rows and 'M' number of columns in our feature matrix 'X' and target 'Y' is $(N \times 1)$ matrix, then our regression parameters can be estimated by the solving the following equation given below.

$$\boldsymbol{w = (x^T x)^{-1} x^T y}$$

where,
W is $(N \times 1)$ matrix known as regression parameters.
This approach is known as closed form or analytical solution. However this approach of obtaining parameters becomes computationally very difficult when number of features increases because number of operations to invert the matrix scales cubically with the increase in the number of features.

B. Regression Model with basis functions

The basic idea behind basis functions is that in addition to the original inputs, we add inputs that are calculated as deterministic functions of the existing inputs and treat them as additional inputs mainly to improve our model's accuracy. for 1D data,

$$\boldsymbol{y_i^{'} = w_o + w_1 x_i + (w_2 x_i^2 + w_3 x_i^3 + \cdots + w_m x_i^m)}$$

$$\boldsymbol{= \sum_{j=1}^{m} w_j \phi_j(x_i)}$$

where, $\phi_j(x_i)$ is known as basis function.
Common basis functions for 1D data,

- Polynomial Basis Function:
  $(1, x_1, x_2, x_3, x_1.x_2, x_2.x_3, x_1.x_3, x_1^2, x_2^2, x_3^2)$
  In short, $\phi_j(x) = x^j$
- Radial Basis Function:

$$\phi_j(x) = e^{-}(\frac{x-\mu_j)^2}{2\sigma^2}$$

Closed form solution for basis function is:

$$\boldsymbol{w^* = (\phi^T \phi)^{-1} \phi^T . y}$$

When we use high degree polynomial basis functions and some other complex sets to fit the curve exactly to the data points for increasing accuracy of model then the phenomenon of **Over-fitting**(Variance) occurs.So we can say that increasing system complexity leads to over-fitting and its opposite phenomenon is **Under-fitting**(Bias) where the model is too simple to describe the characteristics and they both causes test error.

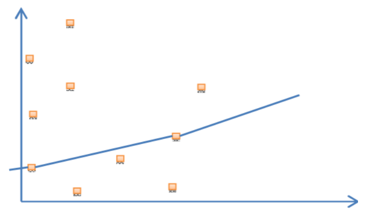


Fig. 1. Under-fitting [1]

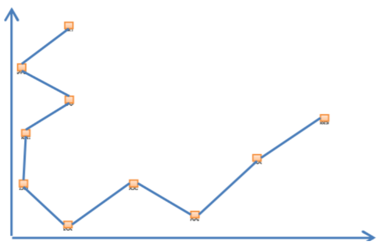


Fig. 2. Over-fitting [1]

C. Loss Function for Linear Regression

A loss function is a measure of how good a prediction model does in terms of being able to predict the expected outcome. There is no single loss function that works for all kinds of data. It depends on the number of factors like presence of outliers and choice of machine learning

algorithm,etc.
Mean Square Error(MSE) is the most commonly used regression loss function which is represented as:

$$MSE = \frac{1}{n}\sum_{D}(y - y^{'})^2$$

where,
D = $(x_i, y_i)|_{i=0}^{n}$
n= number of training examples
y = output/dependent/'target' variables
$y^{'}$ =predicted output variables

D. Gradient Descent

Gradient Descent is an iterative optimisation algorithm to find the minimum of a function and in linear regression that function is the error function E(D,W) or MSE.
We start by defining the initial parameter's values and from there gradient descent uses calculus to iteratively adjust the values so they minimize the given error function(also known as cost function).

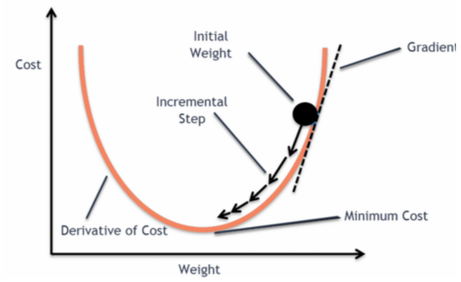


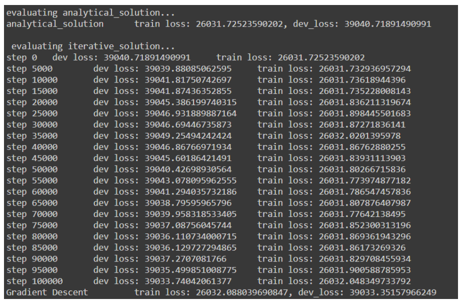
Fig. 3. Figure shows how gradient descent algorithm reaches minima [2]

for 1D data,
Hypothesis: $h(x, w) = w_o + w_1 x$
Parameters: w_o, w_1
Cost function: $J(w_o, w_1) = \frac{1}{2n}\sum_{i=0}^{n}(h(x^i, w) - y^i)^2$
Goal: $\underset{w_o, w_1}{minimize} J(w_o, w_1)$
Gradient descent algorithm: Repeat until convergence,

$$w_j \leftarrow w_j - \alpha\frac{\partial J(w_o, w_1)}{\partial w_j}$$

where,
α is the learning rate

The MSE loss as reported in fig 5 is obtained without any feature engineering on given dataset. This result is improved as shown in subsequent sections after introducing Basis function, regularization, early stopping,etc.



```
evaluating analytical_solution...
analytical_solution      train loss: 26031.72523590202, dev_loss: 39040.71891490991

 evaluating iterative_solution...
step 0   dev loss: 39040.71891490991     train loss: 26031.72523590202
step 5000       dev loss: 39039.88085062595     train loss: 26031.732936957294
step 10000      dev loss: 39041.81750742697     train loss: 26031.73618944396
step 15000      dev loss: 39041.87436352855     train loss: 26031.735228008143
step 20000      dev loss: 39045.386199740315    train loss: 26031.836211319674
step 25000      dev loss: 39046.931889887164    train loss: 26031.898445501683
step 30000      dev loss: 39046.69446735873     train loss: 26031.87271836141
step 35000      dev loss: 39049.25494242424     train loss: 26032.0201395978
step 40000      dev loss: 39046.86766971934     train loss: 26031.86762880255
step 45000      dev loss: 39045.60186421491     train loss: 26031.83931113903
step 50000      dev loss: 39040.42698930564     train loss: 26031.80266715836
step 55000      dev loss: 39043.078095962555    train loss: 26031.773974877182
step 60000      dev loss: 39041.294035732186    train loss: 26031.786547457836
step 65000      dev loss: 39038.79595965796     train loss: 26031.807876407987
step 70000      dev loss: 39039.958318533405    train loss: 26031.77642138495
step 75000      dev loss: 39037.08756045744     train loss: 26031.852300313196
step 80000      dev loss: 39036.110734000715    train loss: 26031.869361943296
step 85000      dev loss: 39036.129727294865    train loss: 26031.86173269326
step 90000      dev loss: 39037.2707081766      train loss: 26031.829708455934
step 95000      dev loss: 39035.499851008775    train loss: 26031.900588785953
step 100000     dev loss: 39033.74042061377     train loss: 26032.048349733792
Gradient Descent         train loss: 26032.088039690847, dev_loss: 39033.35157966249
```

Fig. 4. MSE loss after applying analytical solution and gradient descent on raw data

III. Effect of Regularization

Regularization is the technique to modify the error function so that model complexity is also explicitly penalized.

$$Loss_{reg}(w) = Loss_D(w) + \lambda Reg(w)$$

where, λ is the scaling factor.
We have different types of regression techniques which uses regularization to overcome the problem of overfitting.
1: Ridge Regression
A popular penalty function $Reg(w) = ||w||_2^2$ also known as L2 regularization, when applied to linear regression then it is known as ridge regression. Adding this penalty to our loss function reduces the effect of over-fitting due to increased complexity of model.

$$w_{Ridge} = \arg\min_{w} ||\phi w - y||_2^2 + \lambda ||w||_2^2$$

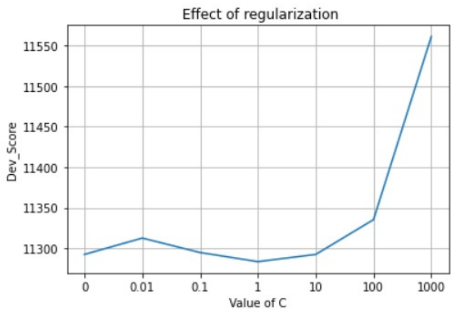


Fig. 5. Plot to demonstrate the effect of regularization

The plot in figure 5 is obtained for the C=1e8 for analytical solution and C=1e-8 which is because of the inverse relationship between them.
2: Lasso Regression
It is known as least absolute shrinkage and selection operator regression.Instead of penalizing the L2 norm of the parameter vector, here L1 norm is penalized.

$$w_{Lasso} = \arg\min_{w} ||\phi w - y||_2^2 + \lambda ||w||_1$$

Regularization can be achieved using different types of priors on the parameters. We get an L2 regularized solution for the linear regression problem using a Gaussian prior on the weights and Lasso(L1-regularized solution) is the MAP estimate of linear regression subject to a Laplace prior.

IV. Gradient Descent Stopping criteria

Early stopping is a form of regularization used to avoid over-fitting when training a learner with an iterative method, such as gradient descent. Such methods update the learner so as to make it better fit the training data with each iteration. Up to a point, this improves the learner's performance on data outside of the training set. Past that point, however, improving the learner's fit to the training data comes at the expense of increased generalization error. Early stopping rules provide guidance as to how many iterations can be run before the learner begins to over-fit. Early stopping rules have been employed in many different machine learning methods, with varying amounts of theoretical foundation [3]. The early stopping rules proposed for these problems are based on analysis of upper bounds on the generalization error as a function of the iteration number. They yield prescriptions for the number of iterations to run that can be computed prior to starting the solution process.

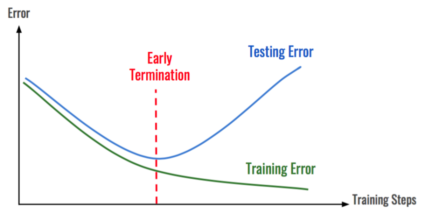


Fig. 6. Figure to illustrate early stopping criteria [3]

Unfortunately we were unable to implement the early stopping criteria. So we used an alternate technique where, after all iterations were completed for the development set, the minimum weights were stored as an array during the iterations. After all iterations are completed, the final errors are calculated on the basis of these gradient descent weights. Also a lot of trial and error was involved, where constantly learning rate and maximum iterations were adjusted so that the gradient descent could converge to the obtained analytical solution. The final parameters obtained for gradient descent which seemed to work well were : LR = 0.01, iterations = 2,00,000 , evalsteps = 20000, c = 1e-8 and batchsize = 512.

V. Feature Importance

In the given assignment we have been provided with 13 different attributes describing fires in Australia. Our target is to find the appropriate Fire Radiative Power(FRP) for raging forest fires in Turkey. A heat map is a two-dimensional representation of information with the help of colors. Heat maps can help the user visualize simple or complex information. Here we used heatmap to visualize any NAN value(represented by different colour in heatmap) if present in our data set. Luckily in our data there is no NAN value present so we get uniform colour throughout the heatmap. Firstly, we found the analytical solution using all the features including date and time by converting them to a numbering system where latest time corresponded to larger value, using the 'mktime' function of 'time' library in colab, and estimated the loss. Then we went on dropping each redundant feature, by elimination method. After estimating loss due to all the features, after dropping each feature one by one, we checked if there was any significant drop in MSE loss. From this method, we dropped the "acqdate", "acqtime feature", as dropping those did not have any significant effect on MSE loss.Also we dropped the redundant feature like "version","instrument" and "frp"(Since it is the output). At the same time, we found out that increasing power of "brightness" feature was able to reduce the MSE loss significantly, similar effect was observed for "latitude", "longitude", "scan", "track" as well as "confidence" and "brightt31". Also, we converted "satellite" and "daynight" feature into categorical data, using the "getdummies" function of pandas. "brightness" is the most important feature, because eliminating it from the feature list resulted into loss going very high.without "brightness": (without basis functions and excluding "acqdate","acqtime","frp","instrument","version").

```
analytical_solution.......
train loss: 39368.78482043279
dev_loss: 52180.322438570554
```

with "brightness" : (without basis functions and excluding "acqdate","acqtime","frp","instrument","version")

```
analytical_solution......
train loss: 26031.72523590202
dev_loss: 39040.71891490991
```

While least important features is track :
with "track" : (without basis functions and excluding "acqdate","acqtime","frp","instrument","version")

```
analytical_solution......
train loss: 26031.72523590202
dev_loss: 39040.7189149099
```

without "track" : (without basis functions and excluding "acqdate","acqtime","frp","instrument","version")

```
analytical_solution......
train loss: 26071.934948930146
dev_loss: 39082.068864640445
```

The 13 attributes given for the prediction of FRP which are as follows:
Latitude, Longitude, Brightnes, Scan, Track, Acq_date, Acq_time, Satellite, Instrument, Confidence, Version, Brightt31, Day-night.

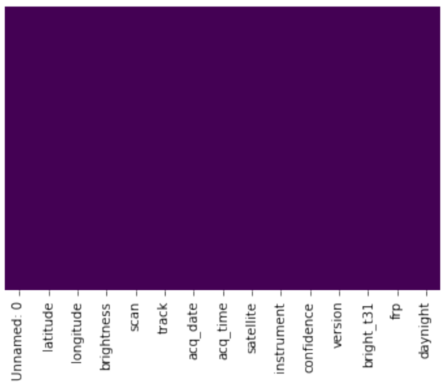


Fig. 7. Heatmap of attributes and output

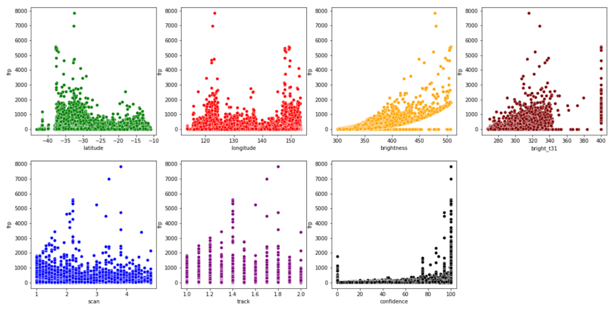


Fig. 8. Plot of FRP against all the important attributes

Observations

- During the training of model, all outliners above the FRP value of 3000 were discarded for smooth model training. This can be seen (Fig: 8) from data-points lying at abnormally high values of FRP.

- Important features are ”Brightness” and ”Bright_t31” as they have a direct relationship with FRP as evident from the scatter plot as shown in (Fig: 8), it can be visualized in terms of exponential relationship or polynomial basis. ”Latitude” and ”Longitude” also have impact on FRP.

- Unimportant features are ”Acq_date”, ”Acq_time” as they did not affect Predictions during model training. Other unimportant features are ”Version” and ”Instrument” because they have same value throughout all the rows.These featues were dropped before model training.

- Then we took polynomial basis functions for important features and then we took their product of two at a time. With this, two new features were generated and then we again repeated this process. We took power 8 polynomial for brightness, confidence, bright_31 and latitude.

- Categorical features such as ”Satellite” and ”Daynight” were assigned with dummy variable. Firstly they were assigned dummies according to the number of categories in all the columns(Terra=1, Aqua=0 for satellite and Daynight_D=0, Daynight_N=1 for Daynight). In the end to get rid of dummy variable trap, we dropped the redundant columns.

- For standardization, we subtracted each features from their mean value and divided by the standard deviation to finally get the feature matrix.

VI. Basis Functions

The two basis functions that we used in the implementation are mentioned in this section. One is the polynomial basis function, and other is feature ensemble technique. For polynomial basis function, we took a particular feature, and passed it to a function called ”createBasisFunctions()”. This function takes in three arguments, feature matrix X, series on which basis function is to be applied, and degree of polynomial ”n”. This leads to addition of (n-1) terms, from linear power to (nth) power. When this basis function is applied to ”brightness”; Here gradient descent solution is obtained for first 50,000 iterations. LR = 0.01, iterations = 2,00,000 evalsteps = 20000. and c = 1e-8 without basis functions on ”brightness” feature:

```
gradient_descent_soln
train loss: 26130.62745287628
dev_loss: 38917.99276874161
```

with basis function on ”brightness” feature (degree 8):

```
gradient_descent_soln
train loss: 22694.182732045476
dev_loss: 35234.84130966437
```

Similarly, we obtained the best result on applying power 8 to latitude and power 8 to brightt31 and power 8 to scan:

```
gradient_descent_soln
train loss: 22293.6821431273
dev_loss: 34674.89490864072
```

The second basis function that we have used is feature ensemble. In feature ensemble, we take product of each feature with the other feature and create new features. The MSE obtained using feature ensemble was amazingly low.

with only feature ensemble : (LR = 0.01, iterations = 2,00,000 evalsteps = 20000).

```
gradient_descent_soln
train loss: 9704.477625955145
dev_loss: 17057.595737893997
analytical_solution
train loss: 9006.079659416866,
dev_loss: 17038.36391509057
```

Another upgraded version we implemented was using the feature ensemble twice, means now, instead of 2, a combination of 4 columns were multiplied at an instant. We have implemented the same thing in kaggle.

The main point to note here is the presence of outliers. We can easily bring the MSE down by removing the outliers, the same has been implemented in kaggle, where we train the model after removing the outliers from the train set, which gives better result.

VII. Results and Experiments

After dropping the irrelevant features and using simple linear regression the MSE is not very satisfying as shown in fig 4. Then we remove the outliers that can be visualized in the fig: 8. Removing the outliers from the training data improves the MSE loss on train as well as validation data by almost halving the previous MSE as discussed in section V, then we tried several basis functions namely exponential, power, inverse-exponential, logarithmic,etc. And after several iterations of trying different basis functions power and product of different feature vectors had proved to be the most important basis functions and they drastically improved the model performance. The final MSE reported after applying all the approaches is 2034.266 on training data and 2153.57 on the validation set that is updated on Kaggle platform too. The plot of MSE loss on development set versus the training size is decreasing exponentially which matches the theoretical aspects as shown in figure 9.

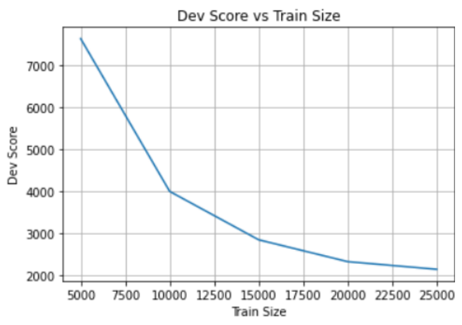


Fig. 9. Plot of development loss vs training size

VIII. Conclusion

An estimate of fire radiation power using this model with a reasonable accuracy as evident from the position on kaggle competition, included learning of various methods like feature ensemble, feature scaling, data visualisation and implementation of linear regression through analytical solution and gradient descent. The model takes longer time for execution due to limited computation power hence certain basis functions were not included in LR.py, thereby resulting in a relatively higher Mean squared error. The importance of a particular feature and it's effect on the output was also a major challenge, which included timely visualisation of error after dropping/adding a feature to the model. The other challenges included hyper parameter tuning, as well as convergence of gradient descent.This model also has an improved performance compared to sklearn's linear regression on this particular dataset.

Acknowledgment

We would like to thank Prof. Preethi Jyothi, Dept. of Computer Science and Engineering, IIT Bombay for his keen efforts during teaching that helped throughout the assignment and also students of Foundations of Machine Learning (CS 725) whose discussion helped to avoid and correct several mistakes while implementing the assignment.

References

[1] Jabbar, H., and Rafiqul Zaman Khan. ”Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study).” Computer Science, Communication and Instrumentation Devices (2015): 163-172.

[2] Rekha M, ”The Ascent of Gradient Descent” https://blog.clairvoyantsoft.com/the-ascent-of-gradient-descent-23356390836f

[3] https://en.wikipedia.org/wiki/Earlystopping, accessed on 9 Sep,2021.