Pipelining and Regularization

Tushar B. Kute,
http://tusharkute.com

Regularization

- In order to create less complex (parsimonious) model when you have a large number of features in your dataset, some of the Regularization techniques used to address over-fitting and feature selection are:
  - L1 Regularization
  - L2 Regularization

L1 and L2

- L1 and L2 regularisation owes its name to L1 and L2 norm of a vector w respectively. Here's a primer on norms:
- L1 Norm

$$||\mathbf{w}||_1 = |w_1| + |w_2| + ... + |w_N|$$

- L2 Norm

$$\|\mathbf{w}\|_2 = \left(|w_1|^2 + |w_2|^2 + \ldots + |w_N|^2\right)^{\frac{1}{2}}$$

Vector Norm

- Calculating the length or magnitude of vectors is often required either directly as a regularization method in machine learning, or as part of broader vector or matrix operations.
- The length of the vector is referred to as the vector norm or the vector's magnitude.
- The length of a vector is a nonnegative number that describes the extent of the vector in space, and is sometimes referred to as the vector's magnitude or the norm.

Vector Norm

- The length of the vector is always a positive number, except for a vector of all zero values.
- It is calculated using some measure that summarizes the distance of the vector from the origin of the vector space.
- For example, the origin of a vector space for a vector with 3 elements is (0, 0, 0).
- Notations are used to represent the vector norm in broader calculations and the type of vector norm calculation almost always has its own unique notation.

Vector L^1 Norm

- The length of a vector can be calculated using the L^1 norm, where the 1 is a superscript of the L.
- The notation for the L 1 norm of a vector is ||v|| 1 , where 1 is a subscript.
- As such, this length is sometimes called the taxicab norm or the Manhattan norm.

$L^1(v) = ||v||_1$

Vector L^1 Norm

- The L^1 norm is calculated as the sum of the absolute vector values, where the absolute value of a scalar uses the notation $|a_1|$.
- In effect, the norm is a calculation of the Manhattan distance from the origin of the vector space.

$$||v||_1 = |a_1| + |a_2| + |a_3|$$

Vector L^2 Norm

- The length of a vector can be calculated using the L^2 norm, where the 2 is a superscript of the L.
- The notation for the L 2 norm of a vector is ||v|| 2 where 2 is a subscript.

  $L^2(v) = ||v||_2$

Vector L^2 Norm

- The L^2 norm calculates the distance of the vector coordinate from the origin of the vector space.
- As such, it is also known as the Euclidean norm as it is calculated as the Euclidean distance from the origin.
- The result is a positive distance value. The L^2 norm is calculated as the square root of the sum of the squared vector values.

$$||v||_2 = \sqrt{a_1^2 + a_2^2 + a_3^2}$$

Errors in Linear Regression

$$Loss = Error(y, \hat{y})$$

Loss function with no regularisation

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i|$$

Loss function with L1 regularisation

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2$$

Loss function with L2 regularisation

Ridge Regression

- A regression model that uses L1 regularization technique is called Lasso Regression and model which uses L2 is called Ridge Regression.
- The key difference between these two is the penalty term.
- Ridge regression adds “squared magnitude” of coefficient as penalty term to the loss function. Here the highlighted part represents L2 regularization element.

Ridge Regression – Cost Function

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

- Here, if lambda is zero then you can imagine we get back OLS.
- However, if lambda is very large then it will add too much weight and it will lead to under-fitting.
- Having said that it's important how lambda is chosen. This technique works very well to avoid over-fitting issue.

Lasso Regression – Cost Function

- Lasso Regression (Least Absolute Shrinkage and Selection Operator) adds “absolute value of magnitude” of coefficient as penalty term to the loss function.

$$\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

- Again, if lambda is zero then we will get back OLS whereas very large value will make coefficients zero hence it will under-fit.

Comparing

- The key difference between these techniques is that Lasso shrinks the less important feature's coefficient to zero thus, removing some feature altogether.
- So, this works well for feature selection in case we have a huge number of features.

Elastic Net

- Elastic net is a popular type of regularized linear regression that combines two popular penalties, specifically the L1 and L2 penalty functions.
- a hyperparameter “alpha” is provided to assign how much weight is given to each of the L1 and L2 penalties.
- Alpha is a value between 0 and 1 and is used to weight the contribution of the L1 penalty and one minus the alpha value is used to weight the L2 penalty.
- elastic_net_penalty = (alpha * l1_penalty) + ((1 – alpha) * l2_penalty)

Dataset: Boston Housing

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	"crim"	"zn"	"indus"	"chas"	"nox"	"rm"	"age"	"dis"	"rad"	"tax"	"ptratio"	"b"	"lstat"	"medv"
2	0.00632	18	2.31	"0"	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24
3	0.02731	0	7.07	"0"	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
4	0.02729	0	7.07	"0"	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
5	0.03237	0	2.18	"0"	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
6	0.06905	0	2.18	"0"	0.458	7.147	54.2	6.0622	3	222	18.7	396.9	5.33	36.2
7	0.02985	0	2.18	"0"	0.458	6.43	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
8	0.08829	12.5	7.87	"0"	0.524	6.012	66.6	5.5605	5	311	15.2	395.6	12.43	22.9
9	0.14455	12.5	7.87	"0"	0.524	6.172	96.1	5.9505	5	311	15.2	396.9	19.15	27.1
10	0.21124	12.5	7.87	"0"	0.524	5.631	100	6.0821	5	311	15.2	386.63	29.93	16.5
11	0.17004	12.5	7.87	"0"	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.1	18.9
12	0.22489	12.5	7.87	"0"	0.524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15
13	0.11747	12.5	7.87	"0"	0.524	6.009	82.9	6.2267	5	311	15.2	396.9	13.27	18.9
14	0.09378	12.5	7.87	"0"	0.524	5.889	39	5.4509	5	311	15.2	390.5	15.71	21.7
15	0.62976	0	8.14	"0"	0.538	5.949	61.8	4.7075	4	307	21	396.9	8.26	20.4
16	0.63796	0	8.14	"0"	0.538	6.096	84.5	4.4619	4	307	21	380.02	10.26	18.2
17	0.62739	0	8.14	"0"	0.538	5.834	56.5	4.4986	4	307	21	395.62	8.47	19.9
18	1.05393	0	8.14	"0"	0.538	5.935	29.3	4.4986	4	307	21	386.85	6.58	23.1
19	0.7842	0	8.14	"0"	0.538	5.99	81.7	4.2579	4	307	21	386.75	14.67	17.5
20	0.80271	0	8.14	"0"	0.538	5.456	36.6	3.7965	4	307	21	288.99	11.69	20.2

Read libraries and dataset

```
import numpy as np
import pandas as pd
import math
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
```

```
#import training dataset
train_df = pd.read_csv('BostonHousing.csv')
```

```
X = train_df.drop('medv', axis=1) #input
y = train_df['medv']    #output

X_train, X_test, y_train, y_test = train_test_split(
    X, y, random_state=0, test_size=0.3)
```

Basic Linear Model

```
lr = LinearRegression()
lr.fit(X_train, y_train)

print('Training Acc:',lr.score(X_train, y_train)*100)
print('Test Acc:',lr.score(X_test, y_test)*100)

y_pred = lr.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = math.sqrt(mse)

print('RMSE:', rmse)
```

```
Training Acc: 76.4545102694255
Test Acc: 67.33825506400194
RMSE: 5.214975145375405
```

Polynomial Model

```
steps = [
    ('scalar', StandardScaler()),
    ('poly', PolynomialFeatures(degree=2)),
    ('model', LinearRegression())
]

pipeline = Pipeline(steps)

pipeline.fit(X_train, y_train)

print('Training Acc:',pipeline.score(X_train, y_train)*100)
print('Test Acc:',pipeline.score(X_test, y_test)*100)
```

```
Training Acc: 95.16134299280634
Test Acc: 64.62815294725782
```

Ridge Regression

```
steps = [
    ('scalar', StandardScaler()),
    ('poly', PolynomialFeatures(degree=2)),
    ('model', Ridge(alpha=10, fit_intercept=True))
]

ridge = Pipeline(steps)
ridge.fit(X_train, y_train)

print('Training Acc:',ridge.score(X_train, y_train)*100)
print('Test Acc:',ridge.score(X_test, y_test)*100)
```

```
Training Acc: 93.7200975636591
Test Acc: 79.09830005263463
```

Lasso Regression

```
steps = [
    ('scalar', StandardScaler()),
    ('poly', PolynomialFeatures(degree=2)),
    ('model', Lasso(alpha=0.3, fit_intercept=True))
]

lasso = Pipeline(steps)

lasso.fit(X_train, y_train)

print('Training Acc:',lasso.score(X_train, y_train)*100)
print('Test Acc:',lasso.score(X_test, y_test)*100)
```

```
Training Acc: 87.19391604372422
Test Acc: 76.41692818439569
```

ElasticNet Regression

```
steps = [
    ('scalar', StandardScaler()),
    ('poly', PolynomialFeatures(degree=2)),
    ('model', ElasticNet(alpha=0.1, fit_intercept=True))
]

eln = Pipeline(steps)

eln.fit(X_train, y_train)

print('Training Acc:',eln.score(X_train, y_train)*100)
print('Test Acc:',eln.score(X_test, y_test)*100)
```

```
Training Acc: 91.89478837576667
Test Acc: 81.01310659764083
```

Useful resources

- www.geeksforthegeeks.org
- www.scikit-learn.org
- www.towardsdatascience.com
- www.medium.com
- www.analyticsvidhya.com
- www.kaggle.com
- www.stephacking.com
- www.github.com

Thank you

This presentation is created using LibreOffice Impress 5.1.6.2, can be used freely as per GNU General Public License

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com