Python Machine Learning: The 4th Book Circle

Methodology of Data Preprocessing: Build the Good Training Data

Jianan Liu

Gothenburg

Dec, 14, 2016

Disclaimer

All opinions and statements in this presentation are mine and do not in any way represent the company
Any comment or correction of error is welcome, please contact me
chisyliu@hotmail.com

Acknowledgement

Acknowledgement

Here I would like to thank for my colleague in our group of Baseband Software Division, Caj Zell, for his comments and explanation of L1 regularization

The Forth Book Circle of Python Machine Learning

In this presentation belongs to **practical methodology** part of the book

- If you have time, please read the book first. This slide could be used as complementary resource for the book
- We will try to go through every methodology in the chapter 4 of book Python Machine Learning, discuss how to offer preprocessing of data
- All of us need to debug the python code, in order to get practice of implementing machine learning algorithm

Overview

Data Preprocessing

From this slide we go through the methodology in chapter 4

Data set ($\mathbf{y}$, $\mathbf{X}$)

Suppose we have L samples, each sample has D features/dimensions(So the input $\mathbf{X}$ is L by D matrix, label $\mathbf{y}$ is L by 1 vector). If we only consider the I^{th} data sample pair, $\mathbf{x}_I$ is 1 by D vector which represents D features/dimension for I^{th} training data sample, $\mathbf{w}$ is D by 1 vector which represents weight and y_I represents the label of I^{th} data sample

Scikit Learn Data Preprocessing

The best guide regarding how to preprocess data in scikitlearn

Link

Outline for Section 1

Create Data Frame Which Has Missing Element by Pandas

Sometimes there are some data of samples or features missing. ('Data' mentioned in this circle stands for the concrete feature element of training data or output label). Firstly we create Data Frame in **pandas** with missing element by using 'pandas.read_csv()'

```
import pandas as pd
from io import StringIO
csv_data = '''A,B,C,D
1.0,2.0,3.0,4.0
5.0,6.0,,8.0
10.0,11.0,12.0,'''
# If you are using Python 2.7, you need
# to convert the string to unicode:
# csv_data = unicode(csv_data)
df = pd.read_csv(StringIO(csv_data))
```

Create Data Frame Which Has Missing Element by Pandas

It creates data frame like:

	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0
2	10.0	11.0	12.0	NaN

- So df object has 3x4 array with two NaN

Checking if Any Missing Element by Pandas

- How to check if a df has NaN or NOT? By using 'DataFrame.isnull()'
- ```
  # check how many NaN elements in df object and
  # return the total number
  df.isnull().sum()
  ```
- We can use the isnull method to return a data frame with Boolean values that indicate whether data is missing (True). Using the sum method, we can then return the number of missing values per column as follows:

```
A    0
B    0
C    1
D    1
dtype: int64
```
```

Eliminating Samples or Features with Missing Element by Pandas

- Can we eliminate the missing element or fill in the missing one with particular number?
- Yes, we can eliminate the rows(samples)/columns(features) which has at least a missing element by using 'DataFrame.dropna()'/'DataFrame.dropna(axis = 1)'

```
df.dropna()
```

	A	B	C	D
0	1.0	2.0	3.0	4.0

Eliminating Samples or Features with Missing Element by Pandas

```
df.dropna(axis = 1)
```

	A	B
0	1.0	2.0
1	5.0	6.0
2	10.0	11.0

- Can we fill in number? Yes, use 'DataFrame.fillna(value)'

```
df.fillna(26)
```

	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	26.0	8.0
2	10.0	11.0	12.0	26.0

Connection Between Pandas and NumPy for Scikit Learn

- Scikit Learn package is designed for working with NumPy
- We noticed we previously processed data by using Pandas, so the problem is how to connect Pandas with NumPy?
- By using 'DataFrame.values', we could access the underlying NumPy array under DataFrame
- ```
  # Data Frame object has 'values' method to
  # return value in array format in NumPy
  df.values
  ```
- It returns

```
array([[  1.,   2.,   3.,   4.],
       [  5.,   6.,  nan,   8.],
       [ 10.,  11.,  12.,  nan]])
```
```

Imputing Missing Element by Sklearn

We previously learned ways of checking, eliminating and filling in missing element by operating Pandas. Does Scikit Learn package offer any method to do the similar things, e.g. interpolate/estimate the possible missing data then fill in it?

- Yes, by using 'sklearn.preprocessing.Imputer()' in the following:

```
from sklearn.preprocessing import Imputer

imr = Imputer(missing_values='NaN', strategy='mean', axis=0)
imr = imr.fit(df)
imputed_data = imr.transform(df.values)
imputed_data
```

```
array([[  1. ,   2. ,   3. ,   4. ],
       [  5. ,   6. ,   7.5,   8. ],
       [ 10. ,  11. ,  12. ,   6. ]])
```

Imputing Missing Element by Sklearn

- By filling in the rules for imputing the null element by 'mean' for that 'row(for each sample)', we setup the object 'imr' of 'Imputer' class
- Then we call the method 'Imputer.fit(DataFrame)' to let the data we want to impute for by the rules we set in the previous step to learn the parameters and rules
- Finally, we apply the the actual rules for the NumPy object by 'Imputer.transform(DataFrame.values)'

Outline for Section 2

Map Categorical Data into Numerical Element by Sklearn

Data could be given in the format of categories, however we probably would like to have some numerical number rather than 'string' element.

- Category element:

```
import pandas as pd

df = pd.DataFrame([['green', 'M', 10.1, 'class1'],
                   ['red', 'L', 13.5, 'class2'],
                   ['blue', 'XL', 15.3, 'class1']])

df.columns = ['color', 'size', 'price', 'classlabel']
df
```

	color	size	price	classlabel
0	green	M	10.1	class1
1	red	L	13.5	class2
2	blue	XL	15.3	class1

Map Categorical Data into Numerical Element by Sklearn

- How to map element from 'category' to 'numerical element'?
- We have to define a mapping rule(dictionary way) as a input parameter of method 'DataFrame[selected_feature].map(mapping_rule)' to replace the original category

```
size_mapping = {'XL': 3,
                'L': 2,
                'M': 1}

df['size'] = df['size'].map(size_mapping)
df
```

	color	size	price	classlabel
0	green	1	10.1	class1
1	red	2	13.5	class2
2	blue	3	15.3	class1

Map Categorical Data into Numerical Element by Sklearn

- The other example shows how to map with same method, however we use for loop to define the mapping rule array

```
import numpy as np

class_mapping = {label: idx for idx, label in enumerate(np.unique(df['classlabel']))}
class_mapping
```

```
{'class1': 0, 'class2': 1}
```

```
df['classlabel'] = df['classlabel'].map(class_mapping)
df
```

	color	size	price	classlabel
0	green	1	10.1	0
1	red	2	13.5	1
2	blue	3	15.3	0

Encoding Categorical Features by OneHotEncoder from Sklearn

Outline for Section 3

Standardization of Data for being Suitable for Sklearn

Standardization is really important before applying machine learning algorithm package in sklearn on your data!

- Why standardization? Simply reason is—Scikit Learn needs!
- What is standardization? According to Scikit Learn website: Standardization of datasets is a common requirement for many machine learning estimators implemented in scikit-learn; they might behave badly if the individual features do not more or less look like standard normally distributed data: Gaussian with zero mean and unit variance
  In practice we often ignore the shape of the distribution and just transform the data to center it by removing the mean value of each feature, then scale it by dividing non-constant features by their standard deviation

Standardization

Standardization of Data for being Suitable for Sklearn

How standardization used to make data suitable for Sklearn?

- According to Scikit Learn website: For instance, many elements used in the objective function of a learning algorithm (such as the RBF kernel of Support Vector Machines or the L1 and L2 regularizers of linear models) assume that all features are centered around zero and have variance in the same order. If a feature has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected.

Standardization by Scale in Sklearn

How to do standardization in sklearn?

- By using 'sklearn.preprocessing.scale()'

```
>>> X = [[ 1., -1.,  2.],
...      [ 2.,  0.,  0.],
...      [ 0.,  1., -1.]]
>>> X_normalized = preprocessing.normalize(X, norm='l2')

>>> X_normalized
array([[ 0.40..., -0.40...,  0.81...],
       [ 1.  ...,  0.  ...,  0.  ...],
       [ 0.  ...,  0.70..., -0.70...]])
```

Normalization by Sklearn

Normalization is the process of scaling individual samples to have unit norm(give uniformed weight for every sample to avoid unbalance effect between sample).

- We could select L1 or L2 normalization by using 'sklearn.preprocessing.normalize()'
- L2 normalization means we use element divided by L2 norm over all the feature for each sample $\frac{X_{l,d}}{\|\mathbf{x}_l\|^2}$ to replace all the element $X_{l,d}$ for each row(each sample)l in the data matrix

```
>>> X = [[ 1., -1.,  2.],
...      [ 2.,  0.,  0.],
...      [ 0.,  1., -1.]]
>>> X_normalized = preprocessing.normalize(X, norm='l2')

>>> X_normalized
array([[ 0.40..., -0.40...,  0.81...],
       [ 1.  ...,  0.  ...,  0.  ...],
       [ 0.  ...,  0.70..., -0.70...]])
```

Normalization by Sklearn

- L1 normalization does almost same but we use element divided by L1 norm over all the feature for each sample $\frac{x_{l,d}}{\|\mathbf{x}_l\|}$ to replace all the element $x_{l,d}$ for each row(each sample)l in the original data matrix
- For detail of L_p factor, please go to the section 5

Outline for Section 4

Partitioning Dataset in Training and Test Sets

Partitioning Dataset in Training and Test Sets

Outline for Section 5

L_p normalization/regularization

In general, what is L_p regularization?

- $\|\mathbf{x}\|_p = \left(\sum_d |x_d|^p\right)^{\frac{1}{p}}$ where $\mathbf{x}$ is 1 x D vector

L_1 Normalization/regularization

- L_1 regularization: $\|\mathbf{x}\|_1 = (\sum_d |x_d|^1)^{\frac{1}{1}} = (\sum_d |x_d|)$
- By adding $\|\mathbf{x}\|_1^1$ to the original objective function, the sparse matrix without solution could become higher rank than original one. It makes the matrix form of equations solvable

L_1 Normalization/regularization

- L_1 also has ability to select features

Outline for Section 6

Conclusion

- All the codes for showing how to use the methods/commands in scikit learn to do preprocessing in chapter 4 is in GitHub:
  Link
-
-
-

Conclusion

References

 http://scikit-learn.org/stable/modules/preprocessing ▸ Link

 Hsuan-Tien Lin, Machine Learning Technique Course, National Taiwan University

 Hsuan-Tien Lin, Machine Learning Foundation Course, National Taiwan University

 Bastian Leibe, Machine Learning Course, RWTH Aachen

 Sebastian Raschka, Python Machine Learning

 Zhihua Zhou, Machine Learning(Chinese Version)

Question?