slp

December 3, 2021

0.0.1 MT21MCS013 Jay - MLP

[50]:
```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

[4]:
```
dataset = pd.read_csv('cell_samples.csv')
dataset
```

[4]:
```
          ID  Clump  UnifSize  UnifShape  MargAdh  SingEpiSize BareNuc  \
0    1000025      5         1          1        1            2       1
1    1002945      5         4          4        5            7      10
2    1015425      3         1          1        1            2       2
3    1016277      6         8          8        1            3       4
4    1017023      4         1          1        3            2       1
..       …    …       …         …        …       …          …     …
694   776715      3         1          1        1            3       2
695   841769      2         1          1        1            2       1
696   888820      5        10         10        3            7       3
697   897471      4         8          6        4            3       4
698   897471      4         8          8        5            4       5

     BlandChrom  NormNucl  Mit  Class
0             3         1    1      2
1             3         2    1      2
2             3         1    1      2
3             3         7    1      2
4             3         1    1      2
..          …       …  …    …
694           1         1    1      2
695           1         1    1      2
696           8        10    2      4
697          10         6    1      4
698          10         4    1      4

[699 rows x 11 columns]
```

[5]:
```
dataset = dataset[pd.to_numeric(dataset['BareNuc'],errors='coerce').notnull()]
dataset['BareNuc'] = dataset['BareNuc'].astype('int')
dataset.dtypes
```

```
/tmp/ipykernel_6345/2488212115.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  dataset['BareNuc'] = dataset['BareNuc'].astype('int')
```

[5]:
```
ID             int64
Clump          int64
UnifSize       int64
UnifShape      int64
MargAdh        int64
SingEpiSize    int64
BareNuc        int64
BlandChrom     int64
NormNucl       int64
Mit            int64
Class          int64
dtype: object
```

[6]:
```
features = dataset[['Clump', 'UnifSize', 'UnifShape', 'MargAdh', 'SingEpiSize',
       'BareNuc', 'BlandChrom', 'NormNucl', 'Mit']]
X = np.asarray(features)
y = np.asarray(dataset['Class'])
```

[7]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.
↪2,random_state=4)
```

1 Rules

1.0.1 Wnew = Wold + n(Yi-Yi^) * Xi

1.0.2 BiasNew = BiasOld + n(Yi-Yi^)

[56]:
```
class SLP :
    lr = 0.1
    epochs = 100
    weights = None
    bias = None
    def __init__(self,learning_rate = 0.1,n_iteration=1000) :
        self.lr = learning_rate
        self.epochs = n_iteration
        self.weights = None
```

```
        self.bias = None

    def activation_function(self,activation) :
        if activation >= 0 :
            return 4
        else :
            return 2

    def fit(self,X,Y) :
        self.weights = np.zeros(X.shape[1])
        self.bias = 0
        for epoch in range(self.epochs) :
            for i in range(X.shape[0]) :
                y_pred = self.activation_function(np.dot(self.weights,X[i]) +␣
↪self.bias)
                self.weights = self.weights + self.lr * (Y[i] - y_pred) * X[i]
                self.bias = self.bias + self.lr * (Y[i] - y_pred)
        print("training complete")
        print(self.weights,self.bias)

    def predict(self,X) :
        y_predi = []
        for i in range(X.shape[0]) :
            y_predi.append(self.activation_function(np.dot(self.weights,X[i]) +␣
↪self.bias))
        return np.array(y_predi)
```

[57]:
```
clf = SLP()
clf.fit(X_train,y_train)
```

```
training complete
[6.4 2.4 5.2 3.4 1.6 4.6 0.2 1.2 1.2] -78.80000000000058
```

[58]:
```
y_predict = clf.predict(X_test)
print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

           2       1.00      0.93      0.97        90
           4       0.89      1.00      0.94        47

    accuracy                           0.96       137
   macro avg       0.94      0.97      0.95       137
weighted avg       0.96      0.96      0.96       137
```

[ ]:
```

```