Boosting

September 25, 2018

Name: Shushil Kumar Ravishankar
reg : 16BCE1259
topic : Boosting
Boosting : Boosting is a sequential process, where each subsequent model attempts to correct the errors of the previous model.The succeeding models are dependent on the previous model.
steps:
1)A subset is created from the original dataset.
2)Initially, all data points are given equal weights.
3)A base model is created on this subset.
4)This model is used to make predictions on the whole dataset.
5)Errors are calculated using the actual values and predicted values.
6)The observations which are incorrectly predicted, are given higher weights.
7)Another model is created and predictions are made on the dataset.(This model tries to correct the errors from the previous model)
8)Similarly, multiple models are created, each correcting the errors of the previous model.
9)The final model (strong learner) is the weighted mean of all the models (weak learners).
10)Thus, the boosting algorithm combines a number of weak learners to form a strong learner. The individual models would not perform well on the entire dataset, but they work well for some part of the dataset. Thus, each model actually boosts the performance of the ensemble

Adaboost:
Adaptive boosting or AdaBoost is one of the simplest boosting algorithms. Usually, decision trees are used for modelling. Multiple sequential models are created, each correcting the errors from the last model. AdaBoost assigns weights to the observations which are incorrectly predicted and the subsequent model works to predict these values correctly.

XG-boost:
XGBoost (extreme Gradient Boosting) is an advanced implementation of the gradient boosting algorithm. XGBoost has proved to be a highly effective ML algorithm, extensively used in machine learning competitions and hackathons. XGBoost has high predictive power and is almost 10 times faster than the other gradient boosting techniques. It also includes a variety of regularization which reduces overfitting and improves overall performance. Hence it is also known as 'regularized boosting' technique. it works for both regression ans classification problems

```
In [1]: #adaboost classification
        import pandas as pd
        from sklearn import model_selection
        from sklearn.preprocessing import LabelEncoder

In [2]: from sklearn.ensemble import AdaBoostClassifier
```

In [3]:
```
data=pd.read_csv('clean_bmart.csv',sep=',')
data.head()
```

Out[3]:

	Unnamed: 0	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	\
0	0	FDA15	9.30	Low Fat	0.016047	
1	1	DRC01	5.92	Regular	0.019278	
2	2	FDN15	17.50	Low Fat	0.016760	
3	3	FDX07	19.20	Regular	0.000000	
4	4	NCD19	8.93	Low Fat	0.000000	

	Item_Type	Item_MRP	Outlet_Identifier	\
0	Dairy	249.8092	OUT049	
1	Soft Drinks	48.2692	OUT018	
2	Meat	141.6180	OUT049	
3	Fruits and Vegetables	182.0950	OUT010	
4	Household	53.8614	OUT013	

	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	\
0	1999	Medium	Tier 1	
1	2009	Medium	Tier 3	
2	1999	Medium	Tier 1	
3	1998	Medium	Tier 3	
4	1987	High	Tier 3	

	Outlet_Type	Item_Outlet_Sales
0	Supermarket Type1	3735.1380
1	Supermarket Type2	443.4228
2	Supermarket Type1	2097.2700
3	Grocery Store	732.3800
4	Supermarket Type1	994.7052

In [4]:
```
X=data.loc[(data['Outlet_Location_Type']=='Tier 1')|(data['Outlet_Location_Type']=='Ti
x=X.values[:,:]
y=X.values[:,10]
ley=LabelEncoder()
ley.fit(y)
y=ley.transform(y)
for i in [1,3,5,7,9,11]:
    en=LabelEncoder()
    en.fit(X.values[:,i])
    x[:,i]=en.transform(x[:,i])

x=x[:,[1,2,3,4,5,6,7,8,9,11,12]]
print (x)
print(y)
```

```
[[156 9.3 0 ... 0 1 3735.138]
 [659 17.5 0 ... 0 1 2097.27]
```

```
 [438 16.2 1 ... 0 1 1076.5986]
 ...
 [890 8.38 1 ... 0 1 549.285]
 [1348 10.6 0 ... 1 1 1193.1136]
 [50 14.8 0 ... 1 1 765.67]]
[0 0 1 ... 1 1 0]
```

```
In [5]: seed = 7
        num_trees = 30
```

```
In [6]: kfold = model_selection.KFold(n_splits=10, random_state=seed)
        model = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
        results = model_selection.cross_val_score(model, x, y, cv=kfold)
        print(results.mean())
```

```
1.0
```

mean estimate of classification accuracy = 1.0 this example demonstrates the constuction of 30 decision trees in sequence using the adaboost algorithm

```
In [7]: import pandas
        from sklearn import model_selection
        from sklearn.ensemble import GradientBoostingClassifier
        from sklearn.metrics import accuracy_score
```

```
In [8]: seed = 7
        num_trees = 100
```

```
In [9]: kfold = model_selection.KFold(n_splits=10, random_state=seed)
        model = GradientBoostingClassifier(n_estimators=num_trees, random_state=seed)
        results = model_selection.cross_val_score(model, x, y, cv=kfold)
        print(results.mean())
```

```
1.0
```

mean estimate of classification accuracy = 1.0 this example demonstrates the constuction of 100 decision trees in sequence using the gradient boost algorithm

```
In [10]: from xgboost import XGBClassifier
```

```
In [11]: model=XGBClassifier()
         model.fit(x,y)
         y_pred=model.predict(x)
```

```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/sklearn/preproces
  if diff:
```

```
In [12]: print(accuracy_score(y_pred,y))
```

```
1.0
```

mean estimate of classification accuracy = 1.0 using the gradient boost algorithm