Expt. No.

Name : Piyush Pandurang Bumte Class : TYCS Roll No.: 523

Title of Experiment : Adaboost Ensemble learning

Date

Sub titles : Assignment/ Problem Solution, Flow chart/Algorithm, Problem Listing, Input Screen, Output Screen, Comments (If any)

Aim :- Adaboost Ensemble Learning.
- Implement the Adaboost algorithm to create an ensemble of weak classifier.
- Train the ensemble model on a given dataset and evaluate its performance.
- Compare the results with individual weak classifier.

Theory :-

★ Adaboost Algorithm

Adaboost, short for adaptive boosting is an ensemble machine learning algorithm that can be used in a wide variety of classification and regression tasks.

It is supervised learning algorithm that is used to classify data by combining multiple weak or bare learners into a strong learner.

★ Key Takeaways

1. Adaboost is an ensemble learning technique used to improve the predictive accuracy of any given model by combine multiple "weak" learners.

Remark

Signature

2. Adaboost works by weighting incorrectly classified instances more ~~wei~~ heavily so that the subsequent weak learners focus mainly on the difficult cases.

3. It is adaptive in the sense that subsequent weak learners are tweaked in favour of those instances missclassified by previous classifiers.

4. Adaboost is fast simple to implement and versatile.

5. Adaboost is not suitable for noisy data and is sensitive to outliers.

AdaBoost

Code:

```
class perceptron:
  def Adaboost(self,examples,K):
     w=[]
     N=len(examples[1])
     y=examples[1]
     for i in range(0,N):
       w.append(1/N)
     print("Original w: ", w)
     h=[]
     for k in range(0,K):
       print("K= ",k+1)
       h.append([])
       h=self.L(examples,w)
       error=0
       for j in range(0,N):
         if(h[j] !=y[j]):
           error=error+w[j]
       print("error : ",error)
       for j in range(0,N):
         if(h[j] == y[j]):
           w[j]=w[j]*error/(1-error)
       self.normalize(w)
       print(w)
       print("result using final w: ")
       h=self.L(examples,w)
       print(h)
  def normalize(self,w):
    for t in range(0,len(w)):
      normalizer =1/float(sum(w))
      w=[x*normalizer for x in w]

  def L(self,ex,w):
    hresult=[]
    for i in range(0,len(ex[1])):
      hresult.append(0)
      hresult[i]=hresult[i]+(w[i]*ex[0][i])
    return hresult
ex=[ [1,2,3,4,5,6],[15,20,30,40,45,60] ]
k=30
p=perceptron()
p.Adaboost(ex,k)
```

Output:

```
Original w:  [0.16666666666666666, 0.16666666666666666, 0.16666666666666666,
0.16666666666666666, 0.16666666666666666, 0.16666666666666666]
K=  1
error :  0.9999999999999999
[0.16666666666666666, 0.16666666666666666, 0.16666666666666666,
0.16666666666666666, 0.16666666666666666, 0.16666666666666666]
result using final w:
[0.16666666666666666, 0.3333333333333333, 0.5, 0.6666666666666666,
0.8333333333333333, 1.0]
........
K=  30
error :  0.9999999999999999
[0.16666666666666666, 0.16666666666666666, 0.16666666666666666,
0.16666666666666666, 0.16666666666666666, 0.16666666666666666]
result using final w:
[0.16666666666666666, 0.3333333333333333, 0.5, 0.6666666666666666,
0.8333333333333333, 1.0]
>>>
```