Homework 4 Report

Solomon Reinman
CSE 142: Machine Learning
Winter 2021

March 11, 2021

1 Boosting Classifier

1.1 Implementation

We implement a generic *BoostingClassifier* class with fit and predict functions. The fit function takes in an array $\boldsymbol{X}$ of instances and an array $\boldsymbol{y}$ of their respective labels. The output of fit is an instance of the *BoostingClassifier* class instantiated with a list M of classifiers of class $\mathcal{A}$. That is, our classifier can be trained with an arbitrary classification alogrithm. The class implementing algorithm $\mathcal{A}$ must have functions train and predict. In the fit function of *BoostingClassifier* then calls the train function of $\mathcal{A}$ repeatedly and updates weights to produce the desired list of classifiers. The *BoostingClassifier* takes one hyperparameter, T, the number of training epochs. The predict function then returns the weighted prediction of all of the classifiers in M.

As our sole classification algorithm, we implement a *BasicLinearClassifier* class. As described above, it has public functions train and predict which are called in fit and predict respectively and produce an instance of *BasicLinearClassifier* and an array of predictions given an input array of instances respectively.

1.2 Local performance

Our local performance is passable at best. With $T = 5$, that is, 5 boosting iterations, we get accuracies of 93.8%, 88.9%, and 80% on datasets 1, 2, and 3 respectively. However, the accuracy is fairly volatile when randomly splitting the training set into a training and testing set. Below is the output for the training runs on three datasets. Note that in this run, on dataset 3, our boosting algorithm reached an error rate of 0.5 on the third iteration and thus only ran for 3 iterations.

1.2.1 Dataset 1

```
Iteration 1:
Error = 0.09999999999999998
Alpha = 1.0986122886681098
```

```
Factor to increase weights = 5.000000000000001
Factor to decrease weights = 0.5555555555555556
Iteration 2:
Error = 0.2708333333333336
Alpha = 0.49519935201393783
Factor to increase weights = 1.8461538461538445
Factor to decrease weights = 0.6857142857142859
Iteration 3:
Error = 0.21575091575091565
Alpha = 0.6453010524690483
Factor to increase weights = 2.3174872665534814
Factor to decrease weights = 0.6375525455394675
Iteration 4:
Error = 0.34820177487155546
Alpha = 0.31347645938579166
Factor to increase weights = 1.4359490274983229
Factor to decrease weights = 0.7671085632389826
Iteration 5:
Error = 0.4299292735240433
Alpha = 0.1410698598426339
Factor to increase weights = 1.1629819851567704
Factor to decrease weights = 0.8770841525066241

Testing:
False positives: 4
False negatives: 1
Error rate: 0.0625%
```

1.2.2 Dataset 2

```
Iteration 1:
Error = 0.30303030303031
Alpha = 0.4164545614675519
Factor to increase weights = 1.6499999999999997
Factor to decrease weights = 0.7173913043478262
Iteration 2:
Error = 0.24347826086956514
Alpha = 0.5668518042396902
Factor to increase weights = 2.0535714285714293
Factor to decrease weights = 0.660919540229885
Iteration 3:
Error = 0.3440578817733989
Alpha = 0.32263132322612165
Factor to increase weights = 1.4532438478747207
Factor to decrease weights = 0.7622623797230695
```

```
Iteration 4:
Error = 0.2793553938825002
Alpha = 0.47383065667306373
Factor to increase weights = 1.789834780173621
Factor to decrease weights = 0.6938232739904472
Iteration 5:
Error = 0.3773067737733391
Alpha = 0.25049770182969533
Factor to increase weights = 1.3251816154786737
Factor to decrease weights = 0.802963608629331

Testing:
False positives: 0
False negatives: 1
Error rate: 0.11111111111111116%
```

1.2.3 Dataset 3

```
Iteration 1:
Error = 0.15789473684210525
Alpha = 0.836988216785836
Factor to increase weights = 3.166666666666668
Factor to decrease weights = 0.59375
Iteration 2:
Error = 0.21354166666666663
Alpha = 0.6518538850553084
Factor to increase weights = 2.3414634146341466
Factor to decrease weights = 0.6357615894039734
Iteration 3:
Error = 0.5

Testing:
False positives: 2
False negatives: 0
Error rate: 0.19999999999999996%
```

1.3 Comparison with basic linear classifier

Our boosting algorithm performs markedly better than our basic linear classifier. On datasets 1, 2, and 3, we achieve accuracies of 93.8%, 66.7%, and 70%, respectively, using our basic linear classifier. On dataset 1, we actually see the same performance, perhaps suggesting a high degree of linear separability in this dataset.

1.4 Discussion

Further experimentation with classification algorithms other than a basic linear classifier is certainly warranted. It is likely that algorithms that are more sensitive to outliers could actually perform well with a boosting classifier since we actually want our model to be changed when we change the weight of previously misclassified instances.

1.5 Notable collaboration

I posted a lot of my reasoning and understanding of the mathematical formulae on Piazza while assisting other students, but did not share any code with anyone.