Home Work Assignment 2: AdaBoost Classifier Algorithm

Fatemeh Lotfi
University of Colorado, Colorado Springs
Colorado Springs, CO 80918
flotfi@uccs.edu

Introduction

To improve the precision of classifiers, it integrates several classifiers. AdaBoost is a tool for creating iterative ensembles. The AdaBoost classifier produces a powerful classifier by merging many low-performing classifiers, resulting in a high-accuracy classifier. In this paper, we learn how to develop and evaluate the AdaBoost classifier algorithm in Python.

1. Decision stump implementation

A stump is a single-node tree that connects to its leaves. A decision stump makes a prediction based on the value of a single input feature. In this section, we implement a decision stump in Python. To do this, we consider entropy as a metric to split tree nodes. Entropy shows how much dissimilarity or impurity exists in a dataset. The code consists of one class and four functions including a function to fit the tree and build the model, a function to find the best split feature, a function to find the best split values in each feature, and a function to measure the entropy metric for decision making. To define a decision tree classifier, we implement a class and initialize the object's state with maximum depth for the tree.

Find the best splitter value

In this part, our goal is to find the best value in each feature column to split the data into two classes (true/false). To do this, we move the value parameter for whole data in a column and measure the dataset entropy. The best value is the one with the lowest entropy.

Find the best splitter feature

To find the best feature as a root node in the tree, we consider entropy a decision metric. Using the function of the previous part, we measure features' entropy in the dataset separately. Finally, we find the feature which results in minimal entropy.

2. AdaBoost algorithm Explanation

Adaboost is a kind of ensemble method that combines stumps as weak learners to make a decision. It is only necessary for each weak classifier's results to be better than random guessing (rather than 1/2). Adaboost is a binary classifier that maps each example $(\vec{x}^{(i)}, y^{(i)})$ of the dataset to negative or positive classification. Then, The classification decision is made by a linear weighted combination of K classifiers.

$$\mathcal{H}^{(k-1)}(\vec{x}) = \alpha_1 h^{(1)}\vec{x} + \cdots + \alpha_{k-1} h^{(k-1)}\vec{x}. \quad (1)$$

In each iteration of the algorithm, it is extended to a better ensemble classifier by adding another weak classifier.

$$\mathcal{H}^{(k)}(\vec{x}) = \mathcal{H}^{(k-1)}(\vec{x}) + \alpha_k h^{(k)}\vec{x}. \quad (2)$$

The AdaBoost uses the exponential error function $e^{-y^{(i)}h^{(k)}(x^{(i)})}$ for each example $(x^{(i)}, y^{(i)})$ of the dataset and classifier $h^{(k)}$. So, for all N example in the dataset we have

$$L(\mathcal{H}^{(k)}) = \sum_{i=1}^{N} e^{-y^{(i)}\left[\mathcal{H}^{(k-1)}(\vec{x}^{(i)}) + \alpha_k h^{(k)}(\vec{x}^{(i)})\right]}. \quad (3)$$

Assuming $w^{(i)(k)} = e^{-y^{(i)}\mathcal{H}^{(k-1)}(\vec{x}^{(i)})}$ and $w^{(i)(k)} = \frac{1}{N}$ we have

$$L(\mathcal{H}^{(k)}) = \sum_{i=1}^{N} w^{(i)(k)} e^{-y^{(i)}\alpha_k h^{(k)}(\vec{x}^{(i)})}. \quad (4)$$

Then, the summation can split to two parts, misclassified (MC) data examples which $y^{(i)}h^{(k)}(x^{(i)}) = -1$ and data examples with correctly classified (CC) which $y^{(i)}h^{(k)}(x^{(i)}) = 1$. By considering these two part, the loss function in (4) can be rewrite such as:

$$\begin{aligned} L(\mathcal{H}^{(k)}) &= e^{-\alpha_k} \sum_{\vec{x}^{(i)} \in CC} w^{(i)(k)} + e^{\alpha_k} \sum_{\vec{x}^{(i)} \in MC} w^{(i)(k)}, \\ &= e^{-\alpha_k} \sum_{i=1}^{N} w^{(i)(k)} + \left(e^{\alpha_k} - e^{-\alpha_k}\right) \sum_{\vec{x}^{(i)} \in MC} w^{(i)(k)}. \end{aligned} \quad (5)$$

To determine the weight α_k for the kth classifier, we differentiate $L(\mathcal{H}^{(k)})$ with respect to α_k and solve it.

$$\alpha_k = \frac{1}{2}\ln\Big(\frac{\sum_{i=1}^{N} w^{(i)(k)} - \sum_{\vec{x}^{(i)}\in MC} w^{(i)(k)}}{\sum_{\vec{x}^{(i)}\in MC} w^{(i)(k)}}\Big),$$
$$= \frac{1}{2}\ln\Big(\frac{1-e^{(k)}}{e^{(k)}}\Big). \quad (6)$$

Thus, we have the AdaBoost algorithm that expresses like Algorithm 1.

Algorithm 1: AdaBoost Algorithm

Result: $\{h^{(1)}\cdots h^{(K)}\}$ and $\{\alpha^{(1)}\cdots\alpha^{(K)}\}$
$w^{(i)(1)} \leftarrow \frac{1}{N}\ \ i \in \{1,\cdots,N\}$;
for $k = 1$ **to** K **do**
  $h(k) \leftarrow$ a weak classifier ;
  $\alpha^{(k)} \leftarrow \frac{1}{2}\ln\Big(\frac{1-e^{(k)}}{e^{(k)}}\Big)$;
  for $i = 1$ **to** N **do**
    if $x^{(i)}$ *is a misclassified example* **then**
      $w^{(i)(k+1)} \leftarrow w^{(i)(k)}\sqrt{\frac{1-e^{(k)}}{e^{(k)}}}$;
    else
      $w^{(i)(k+1)} \leftarrow w^{(i)(k)}\sqrt{\frac{e^{(k)}}{1-e^{(k)}}}$;
    end
  end
end

As we can see, at first the algorithm initializes with $w^{(i)(1)} = \frac{1}{N}$. Then it does the four-step in each iteration. These steps are briefly described below.

- At first build a stump that has minimal weighted misclassification error as a weak classifier.
- Compute and assign a new weight to each elements.
- Weights on misclassified examples are emphasized, while weights on correctly identified examples are reduced.
- Normalize the weights of all elements to 1.

It repeats the loop a significant amount of times, potentially hundreds of times. Finally, to find a classifier for new example $\vec{x}$, the output of program would be $sign\big(\sum_{k=1}^{K}\alpha^{(k)}h^{(k)}(\vec{x})\big)$.

3. AdaBoost Algorithm Implementation

To implement the AdaBoost classifier, we define a *AdaBoost* class and two functions include of *fit* and *predict*. Since, the label variables should be in $\{-1,+1\}$, it is important to validate them before starting classification. So, we consider additional function *check* to check the format of response variables.
Then, in the *fit* function we use algorithm 1 and a weak classifiers (stumps) to build an Adaboost classifier. Finally, in *predict* function we can estimate the new value. In the following, we show the results of training and testing of this classification.

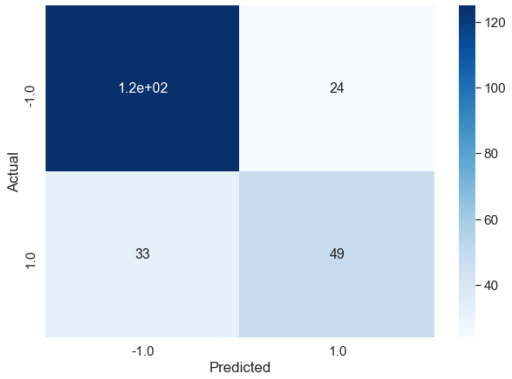

Figure 1: Confusion matrix of dataset 1 AdaBoost classifier with 10 estimators.

4. Analysis of Results

In this section, we show the AdaBoost classification results for two datasets from the UCI Machine Learning Repository.

Dataset 1 : Pima Indian Diabetes

Fig. 1 shows the confusion matrix of AdaBoost classifier with 10 estimators for dataset 1. As we can see, the accuracy of classification is acceptable. To show the effect of the number of estimators in the method perofrmance accuracy, we run the classifier for different numbers of estimators and compare the actual values and predicted values. Fig. 2 shows although for a higher number of estimators we have a better classifier, after a certain point, no matter how much you raise the n estimators, the efficiency of the AdaBoost Classifier will not change.

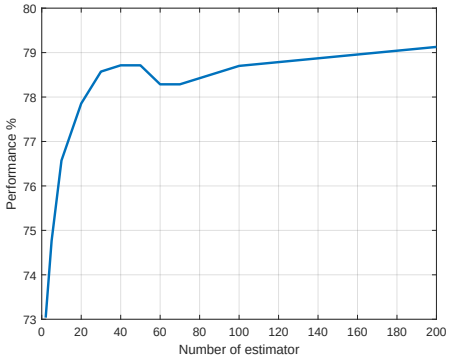

Figure 2: Accuracy performance of AdaBoost classifier vs. the different number of estimators.

Dataset 2: Sonar

Fig. 3 shows the confusion matrix of AdaBoost classifier with 10 estimators for dataset 2. The accuracy performance of classification for this dataset is also acceptable.

5. AdaBoost multi-class classification

In this section, we want to discuss multi-class classification with the AdaBoost algorithm.

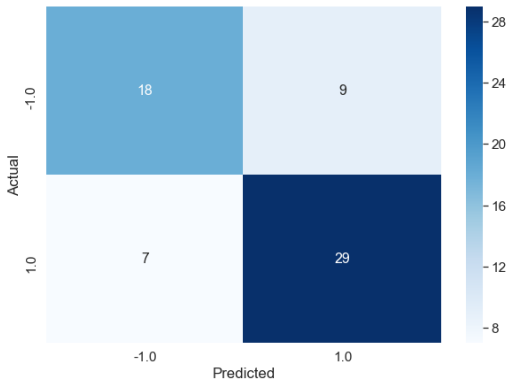

Figure 3: Confusion matrix of dataset 2 AdaBoost classifier with 10 estimators.

Algorithm Explanation

In multi-class classification, the label values include m different groups. When there are m groups in a multi-class grouping, y becomes a k dimensional vector instead of a single variable with a range of $\{-1, 1\}$. After some research we found SAMME(Stagewise Additive Modeling for Multi-class Exponential loss function) multi-class classification algorithm (Hastie et al. 2009). The SAMME is an extension of the AdaBoost algorithm. The algorithm 2 shows the SAMME algorithm's steps. From algorithm 2 we see that,

Algorithm 2: SAMME Algorithm

Result: $\{h^{(1)} \cdots h^{(K)}\}$ and $\{\alpha^{(1)} \cdots \alpha^{(K)}\}$
$w^{(i)(1)} \leftarrow \frac{1}{N} \;\; i \in \{1, \cdots, N\}$;
for $k = 1$ **to** K **do**
  $h(k) \leftarrow$ a weak classifier ;
  $\alpha^{(k)} \leftarrow \ln\left(\frac{1-e^{(k)}}{e^{(k)}}\right) + \ln(m-1)$;
  for $i = 1$ **to** N **do**
    if $x^{(i)}$ *is a misclassified example* **then**
      $w^{(i)(k+1)} \leftarrow w^{(i)(k)} \sqrt{\frac{1-e^{(k)}}{e^{(k)}}}$;
    else
      $w^{(i)(k+1)} \leftarrow w^{(i)(k)} \sqrt{\frac{e^{(k)}}{1-e^{(k)}}}$;
    end
  end
end

only the classifiers weights $\alpha^{(k)}$ are changed.

$$\alpha^{(k)} \leftarrow \ln\left(\frac{1-e^{(k)}}{e^{(k)}}\right) + \ln(m-1) \tag{7}$$

Besides this, the main difference between AdaBoost and SAMME is the performance of the weak classifiers. As we mention, the AdaBoost algorithm works for two classes and allows a weak classifier to have an accuracy of greater than or equal to $1/2$ at any point. While, SAMME deals for m groups and needs only that the weak classifier outperforms the random guess, which in this case is $1/m$ instead of $1/2$. In the following, we present the reason for this. From (7) we

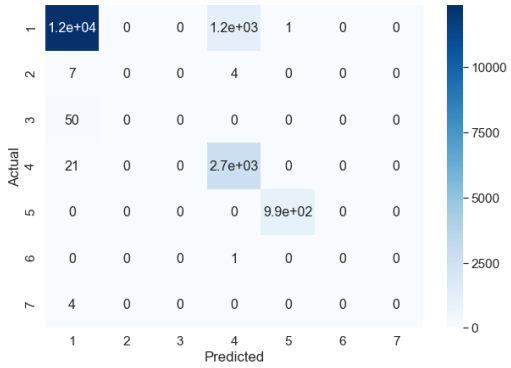

Figure 4: Confusion matrix of dataset 1 SAMME classifier with 10 estimators.

have

$$\alpha^{(k)} \leftarrow \ln\Big(\frac{1-e^{(k)}}{e^{(k)}}(m-1)\Big), \tag{8}$$

and for the algorithm performance, (9) should be positive and it means that the inner part of the logarithm should be larger than 1.

$$\begin{aligned} \frac{1-e^{(k)}}{e^{(k)}}(m-1) &> 1, \\ 1-e^{(k)} &> \frac{1}{m}. \end{aligned} \tag{9}$$

The algorithm repeats the loop a significant amount of times. Finally, to find a classifier for new example $\vec{x}$, the result of classification would be

$$C(\vec{x}) = \underset{m}{\operatorname{argmax}} \sum_{k=1}^{K} \alpha_{(k)}.1(h^{(k)}(\vec{x}) = m). \tag{10}$$

Algorithm Implementation

To evaluate the algorithm performance, we show the results of the implemented algorithm for two different datasets.

Dataset 1: shuttle

This dataset has seven classes of labels and ten different features. Fig. 6 shows the confusion matrix of SAMME algorithm classification. Furthermore, Fig. 5 illustrates the effect of estimators number in the SAMME algorithm. As we can see, the accuracy performance of the SAMME algorithm is higher than the AdaBoost algorithm.

Dataset 2: Page blocks

This dataset has five classes of labels and eleven different features. Fig. shows the confusion matrix of the SAMME algorithm for this dataset.

Comparison of performance

In this part, we want to compare our SAMME implemented classifier with the Python Sklearn library. Fig. 7 illustrates the accuracy performance of them in comparison with each other.

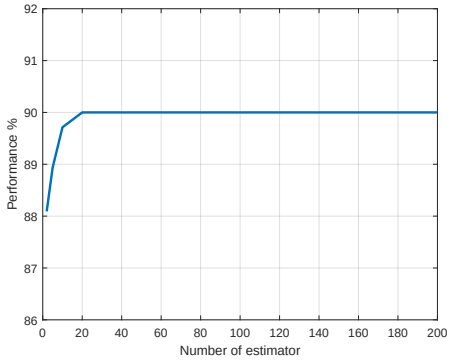

Figure 5: Accuracy performance of SAMME classifier vs. different number of the estimator.

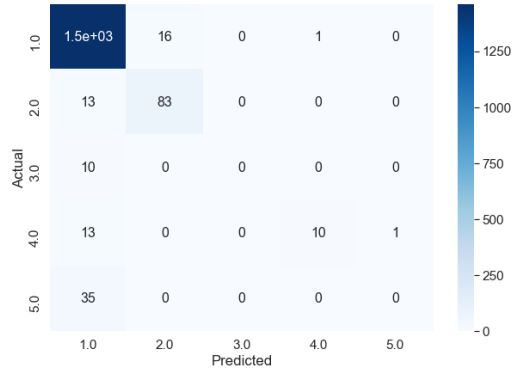

Figure 6: Confusion matrix of dataset 2 SAMME classifier with 10 estimators.

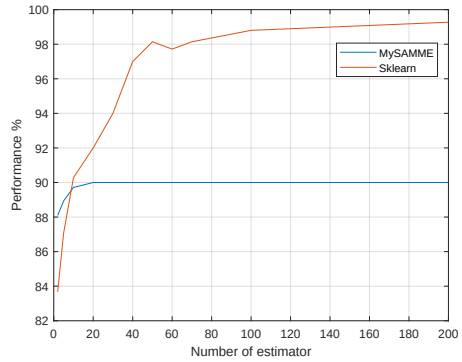

Figure 7: Comparing the accuracy performance of two implemented algorithm MySAMME and Sklearn-SAMME classifiers vs. different number of the estimator.

Conclusion

In this paper, we discuss the AdaBoost classification algorithm and learn how to solve the classification problems and find the effect of the number of estimators in the accuracy performance of the AdaBoost classifier. Also, we discuss the multi-class classification SAMME algorithm as an extension of the AdaBoost algorithm. We implement the AdaBoost and SAMME classifiers in Python and test our model with two different datasets for each of them. Finally, we compare the results of the implemented models with the Sklearn library models to validate our implementation models.

References

Hastie, T.; Rosset, S.; Zhu, J.; and Zou, H. 2009. Multi-class adaboost. *Statistics and its Interface* 2(3):349–360.