Machine Learning Programming Assignment

Mehrgan Khoshpasand

May 15, 2020

Abstract

In this experiment 6 of the most famous machine learning algorithm were implemented in python; ID3, Adaboost, Naive Bayse, Bagged Naive Bayes, Random farost, and KNN. The performance of these algorithms were testes on 5 datasets available in the UCI machine learning repository. 10 times 5-fold accuracy is used to report the accuracy of these algorithms. In this document, an overview of these algorithm as well as an analisys over the experiment result is presented.

1 Introduction

Supervised learning is a family of machine learning algorithms that learn to predict the target value by experiencing a labeled dataset. This document presents an overview of six supervised learning algorithm and goes over the overall structure of the python implementation of these algorithms. The implementation of these algorithms was tested on 5 relatively small datasets and the results are available in section 4. Section 5 provides a conclusion and Future works.

2 Theory

Here give a brief summary of ID3, Adaboost, Naive Bayse, Bagged Naive Bayes, Random farost, and KNN algorithms and provide necessary theoretical background.

2.1 ID3

ID3 [2] is a supervised learning algorithm that builds a tree-like structure called decision trees. Decision trees classify instances by sorting the instances from the root to some leaf node. The leaf nodes that an instance ends up in specify the class of the instance. ID3 builds the tree starting from the root and adda one layer at the time by choosing an attribute to split the instances. This decision is made based on the information gain. Information gain of an attribute, measures the expected reduction in entropy by splitting the instances using the attribute. Entropy measures the impurity of the instances:

$$Entropy(S) = \sum_{c \in C} -p_c log_2 p_c \tag{1}$$

where C is the set of all possible classes, and p_c is the proportion of instances belonging to the class c

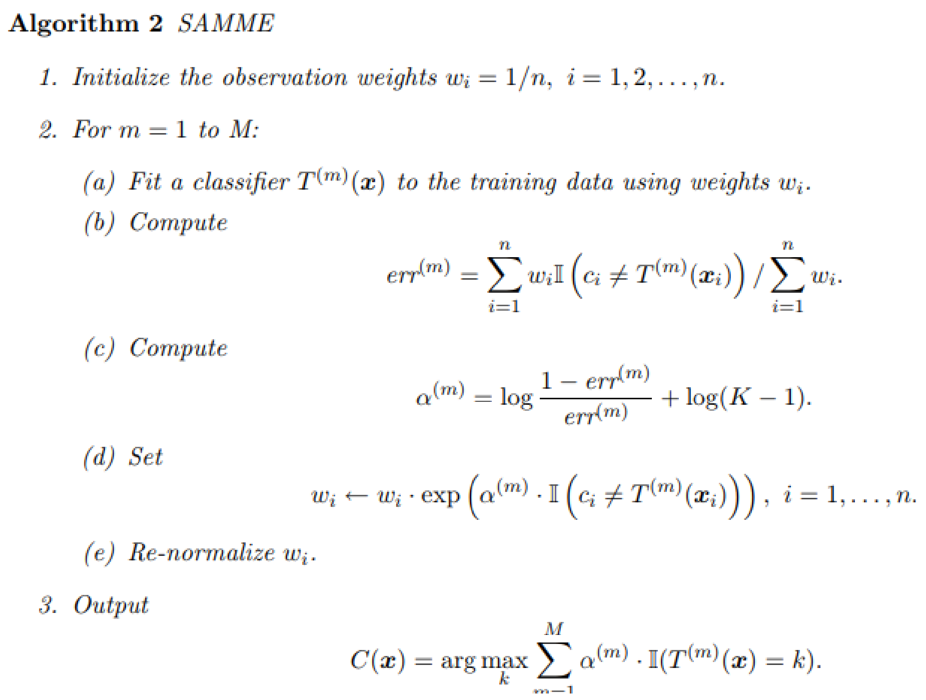
$$InfoGain(S, A) = Entropy(S) - \sum_{v \in V_A} \frac{|S_v|}{|S|} Entropy(S_v) \tag{2}$$

where S_v is a subset of instance that attribute A has the value v and V_A is the set of all the value that A can get.

At each step. ID3 choose an attribute with the highest information gain on the set of instances on that level. Then, it splits the instances based on that attribute and adds n branch(node) to the tree where $n = |V_A|$. Afterwards, if the nodes are perfectly classified, it gives them a label; and if not, it finds another attribute and split the instance based on that attribute. If ID3 runs out of attributes and the node still is impure, it chooses the most repeated class.

2.2 AdaBoost

Ensemble methods are methods machine learning that use a combination of the results of many classifier to classify an instance. Boosting is one of the main categories of ensemble learning. Boosting reduces the bias by paying more attention to the instances that were misclassified before. Adaboost with tree stumps uses an ensemble of tree stumps(trees with one splitting node) and assign a weight to each one of them. Before training the next tree stump, these weight can increase or decrease based on whether that instance was correctly classified by the previous weak-learner. Finally, the final prediction of Adaboost is based of the classification of all of these treestumps and the variable α that shows how well each weak learner performed. The following psudocode is adopted from the work by Hastie et al. [1]:



```
Algorithm 2 SAMME
1. Initialize the observation weights $w_i = 1/n,\ i = 1, 2, \ldots, n$.
2. For $m = 1$ to $M$:
    (a) Fit a classifier $T^{(m)}(\boldsymbol{x})$ to the training data using weights $w_i$.
    (b) Compute
            $err^{(m)} = \sum_{i=1}^{n} w_i \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right) / \sum_{i=1}^{n} w_i.$
    (c) Compute
            $\alpha^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}} + \log(K - 1).$
    (d) Set
            $w_i \leftarrow w_i \cdot \exp\left(\alpha^{(m)} \cdot \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right)\right),\ i = 1, \ldots, n.$
    (e) Re-normalize $w_i$.
3. Output
            $C(\boldsymbol{x}) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbb{I}(T^{(m)}(\boldsymbol{x}) = k).$
```

Figure 1: multi-class Adaboost [1]

2.3 Random Forest

Bagging is another ensemble method that uses the average of the results of multiple classifier that were trained on random subset of the training data. Random Forest is bagged decision trees with one modification, each tree is trained on a random subset of the attributes.

2.4 Naive Bayes

Naive Bayes is a generative machine learning algorithm that assumes given a class c, all the features are independent that is called conditional independence:

$$P(a_1, a_2, ..., a_n|v_j) = \prod_{i=1}^{n} P(a_i|v_j) \tag{3}$$

By having this assumption, Naive Bayes prediction given an instance $x = (a_1, a_2, ...a_n)$ becomes:

$$v_{nb} = \underset{v_j \in V}{\text{argmax}}\, \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i|v_j) \tag{4}$$

where $\hat{P}(v_j)$ and $\hat{P}(a_i|v_j)$ are estimated from the dataset.

2.5 Naive Bayes with Bagging

Bagging was introduced in section (2.3) and Naive Bayes was introduced in the previous section. Naive Bayes with bagging is training multiple Naive Bayes classifiers on random subset of the dataset and combine their prediction for classification.

2.6 K-Nearest Neighbor

K-Nearest Neighbor (KNN) for classification uses the majority of the k nearest instances in the dataset in order to classify an instance. The distance of two sample is calculate using an distance function. A simple example for a distance function can be the Euclidean distance on a normalized dataset.

3 Implementation

In this section, a brief overview of implementation details are discussed. Its is worth noting that the code was implemented only by using python standard libraries.

3.1 ID3 implementation

For implementing ID3 algorithm, we used a tree structure with linked list, composing from instances of *Node*. Each node contains a list for attributes and their values. By using this list we can find a subset of the dataset that this node can represent. Each node also has list of children *Nodes*. These children nodes have all the attributes-values of the parent node in common, however, they have one additional attribute. The value for this attribute is different for each child. Starting from the root *Node*, *id3* subroutine decides what attribute A should be used to split the instances by using the result of *infogain* function. Then it create n children where n is the number of values that A can have. Each child each child gets the same list of attributes as the parent in addition to A, v_i, a attribute-value tupple; where v_i is one of the values that attribute A can have. Afterwards, *id3* method is called for each of the children. Finally, when the subset is perfectly pure or we run out of attributes, id3 assigns a class for that $Node$ and sets its state to a $leaf$.

3.2 AdaBoost Implementation

Adaboost uses the *id3* function with one slight modification, the maximum depth of the tree is set to one(tree stump). Then, it calculates the error rate and α according to figure 1. Weights are updated by using the α and normalized. Then, this process happens again, until we reach the maximum number of weak-learners. For the prediction, we take the weighted-vote of the weak-learners according to figure 1. The number of weak-learners is different for each datasets and was choosen using cross-validation.

3.3 Random Forest Implementation

Implementing a random forest algorithm was trivial since ID3 was already implemented. The only difference is training multiple decision trees on random subset of dataset and a random subset of attributes. And then taking a majority vote of these decision trees for classification. The length of the subset of dataset is chosen to be one third of the overall dataset.

3.4 Naive Bayes Implementation

During the learning process of Naive Bayes algorithm, $\hat{P}(v_j)$ and $\hat{P}(a_i|v_j)$ are estimated from the dataset:

$$\hat{P}(v_j) = \frac{n_{v_j}}{n} \tag{5}$$

where n is total number of samples and n_{v_j} is the number of samples belonging to the class v_j and

$$\hat{P}(a_i|v_j) = \frac{n_{a_i,v} + 1}{n_v + k} \tag{6}$$

where k is the number of attributes, $n_{a_i,v}$ is the number of samples belonging to the class v and having the attribute a_i. after the table for these probabilities is learned predicting is trivial using the equation 4.

3.5 Naive Bayes with Bagging Implementation

This program uses the code from section 3.4 by only one small difference, it trains multiple naive Bayes classifiers on random subset of data and takes the majority vote.

3.6 k-NN Implementation

For the prediction, k-NN goes over all the training data and find the k nearest samples to the instance we want to classify. Since 3 of the datasets are categorical and 2 are numerical, I implemented 3 different distance functions. For the numerical datasets the following distance functions are used:

$$D_1(S, S') = \sum_{a_i, a'_i \in S, S'} |a_i - a'_i|^2 \tag{7}$$

$$D_2(S, S') = \sum_{a_i, a'_i \in S, S'} |a_i - a'_i| \tag{8}$$

and for the categorical datasets, I used the following distance function:

$$D_3(S, S') = \sum_{a_i, a'_i \in S, S'} \mathbb{I}(a_i \neq a'_i) \tag{9}$$

Then the majority vote between these k nearest neighbors decides the output of the algorithm.

4 Datasets

In this section, an overview of the five datasets is presented. The datasets are Car Evaluation Data Set(C), Letter Recognition Data Set(L) , Mushroom Data Set(M), Breast Cancer Wisconsin (Diagnostic) Data Set(B), and Ecoli Data Set(E). The overview of the datasets are presented in table 1.

Table 1: Overview of the datasets

	Size	Attributes	Numerical	Number of Classes	Missing Attributes
C	1728	6	N	4	N
B	569	9	Y	2	Y
E	336	7	Y	8	N
M	8124	22	N	2	Y
L	20000	16	N	26	N

The missing attributes of the Ecoli datasets were replaced with the most common value of the attribute. However, For Mushroom dataset, since most of the values of one attribute were missing, that attribute was eliminated for the training.

Numerical values of the Ecoli datasets were categorized into 4 categories(A,B,C,D) using a simple discretization function. This function splits the range of the values of each attribute into four categories and replace the values with the corresponding label. However, the attributes of the Breast Cancer datasets were treated as categorical, since the values were integers between 1 to 10. It should be mentioned that when a numerical distance function was used in kNN algorithm, no discretization happened on the numerical attributes. You can see the comparison of using discretization or treating these attributes as numerical in kNN algorithm in appendix A.

5 Results and Analysis

In this section, the experimental results presented. Here, you see the performance of the pure algorithms, without problem specific modifications. We tested the 6 algorithms on 5 datasets available on UCI machine learning repository. These algorithms are trained on Car Evaluation Data Set(C), Letter Recognition Data Set(L) , Mushroom Data Set(M), Breast Cancer Wisconsin (Diagnostic) Data Set(B), and Ecoli Data Set(E). The accuracy is calculated using 10 times 5-fold cross-validation, and the average accuracy and standard deviation is reported.

The accuracy results are presented in table 2.

Table 2: Average accuracy of 10 times 5-fold cross-validation

	C	B	E	M	L
ID3	90.2608%	92.9496%	77.61194%	100%	76.0150%
RF	89.2173%	95.9712%	94.9253%	100%	66.0500%
AB	81.4492%	96.2589%	80.2985%	98.5221%	47.4950%
NB	61.6811%	97.2661%	80.5970%	96.4039%	72.4650%
BNB	70.5507%	96.9784%	75.5223%	94.9876%	70.2725%
kNN	76.6376%	97.1223%	84.4776%	100%	87.4400%

It is worth noting that for kNN, the result of the best distance function is reported. See appendix A for the complete results.

The standard deviation of the 50 results obtained for each algorithm on each dataset is available in table 3.

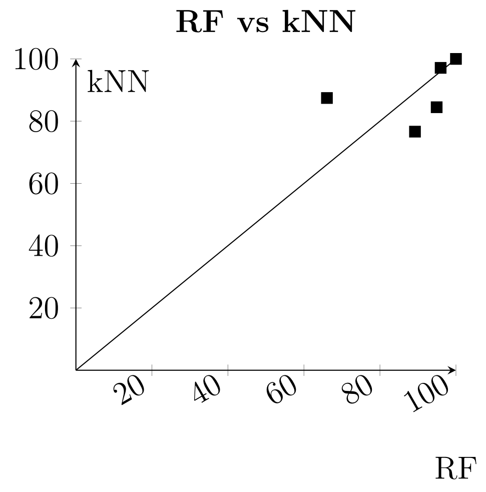
Table 3: Standard deviation of the 50 results

	C	B	E	M	L
ID3	01.01478	01.6824	05.0994	0.0	00.3168
RF	02.0571	01.2670	02.0451	0.0	01.3958
AB	01.3980	01.0680	05.1703	02.4390	01.1152
NB	06.8752	01.1627	03.9315	00.2853	00.4919
BNB	02.1081	01.0680	08.1805	00.6831	00.0901
kNN	05.5972	00.9192	03.8850	0.0	00.5728

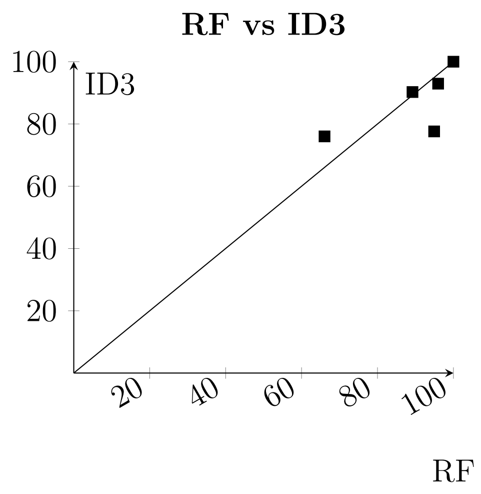
Implementing all of these algorithms does not any require knowledge about the tasks that are being learned. These algorithms also run reasonably fast on the five datasets. As you can see in table 2, Random Forest achieved the best results. In figure 2.a, you can see the comparison of the two algorithms with the best performance. In figures 2.b, 3.a, and 3.b the effect of Ensemble methods is demonstrated. As you can see, in general, ensemble methods increase the performance. However, Adaboost performs poorly on Letter dataset which might be the result of higher dimensionality of the datasets and size of the dataset.

6 Conclusions

In this project, six of the most common machine learning algorithms were implemented and tested on five publicly available datasets. As was demonstrated by the results, Ensemble methods and kNN are a powerful algorithms that can achieve a high performance without much knowledge about the tasks. Now, we better understand the reason why random forest and boosting algorithms are the winners of machine learning competitions most of the times. Finally, By completing these project and implementing these algorithms, I have a better appreciation of these algorithms and the science behind each of them.

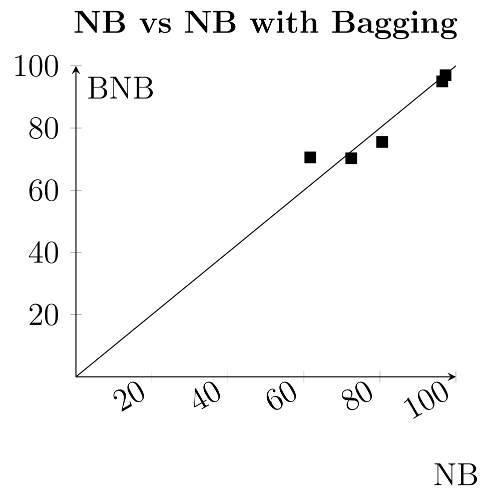


(a) Two of the best algorithms

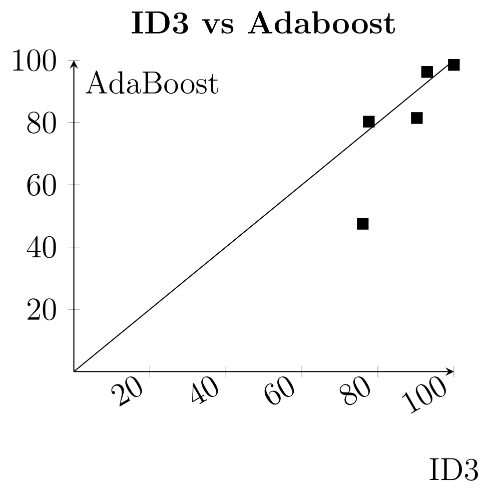


(b) Showing how random forest improves the performance

Figure 2: Comparison between Random Forest and ID3 and kNN



(a) Effect of bagging



(b) Effect of Boosting

Figure 3: Showing the effect of Ensemble methods

Appendices

.

A kNN Distance Functions

In table 4, the performance of kNN with the 3 distance function is presented. D_1, D_2, and D_3 are according to equations (7) to (9) accordingly

Table 4: Standard deviation of the 50 results

	D_1	D_2	D_3
C	N/A	N/A	76.6376%
B	97.1223%	94.6836%	95.2517%
E	84.1791%	83.8805%	78.2089%
M	N/A	N/A	100%
L	N/A	N/A	87.4400%

References

[1] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[2] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.