Modifying continuous ID3 Decision tree algorithm for binary classification with using Principle component algorithm to transform features coordinate

Amirkhosro Vosughi, Ctps 570 Research Project report

***Abstract*— This project is concern about the modification of ID3 algorithm for continuous-feature binary classification in order to reduce the depth of the tree by allowing making decision based on linear combination of the features rather than restricting decision based on value of one feature. The suggestion way to deal with that is to use principle component analysis (PCA). The necessary formulation for this goal will be stated and simulation using Matlab 2017 demonstrate the effectiveness of proposed technique.**

Keywords: ID3 Algorithm, PCA, decision based on linear combination of features

I. INTRODUCTION AND MOTIVATION

This project tries to expand ID3 algorithm to make in more sufficient in for the case of continuous-value features. The main idea is that in ID3 we are only allowed to separate the data in each decision point, based on value of one feature. That hugely limits of ability to deal with the cases where classes are inherently representable based on linear combination of features. For example see figure 1. clearly, the data is separable based with linear boundary $x + y \geq 0$. However, if we want to use traditional ID3 to classify this data, our ID3 will have big depth to be able to classify this data easily and at the end, we will not have good generalization for new data. So it would be quit interesting if we can separate the labels based on linear combination of features in each decision.

However, the main challenge is that given a set of data, how we can find the appropriate linear combination of features and comparing threshold for dividing Data in each node of the decision tree. I think we can simplify the above question in two steps. First we can find the best direction for dividing the data. Knowing that direction is equal to knowing the coefficient of features in linear combination of feature. By knowing that slope, we can rotate the coordinate of features of all samples around the origin with appropriate angle. So we will have all sample in new coordinate. In next step, we can easily deploy our traditional ID3 to find comparing threshold constant.

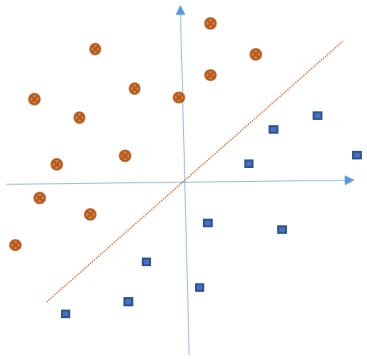

Fig. 1. A simple example when ID3 algorithm failed to give good result.

Therefore, the main question is that how to find appropriate rotation angle in first step. To find a convincing solution, I have tried many approaches. Maybe the simplest way is dividing the range of all possible rotation angles to some interval and for each value rotate the features and find the minimum entropy and choose the rotation angle that leads to minimum entropy. However, it is easy to see that this approach fail when features have big dimension. On other thing that cross from my mind is to find the minimum entropy based data classification from each feature and try to kind of estimate the appropriate rotation angle based on comparing minimum entropy along different features. This was the approach that I initially suggested to solve the problem in my proposal.

Electrical Engineering and Computer Science at Washington State University.

However, it turns out that it works well only for simple cases and there are serious technical issue about extracting optimal rotation angle based entropy value from each feature. Finally, I find very good way to find that rotating angle based on principle component analysis (PCA) [1]. I get familiar with this unsupervised learning algorithm in one of my other courses this semester. PCA tries to find the direction that data has most variation.

To use PCA to find the best rotation angle, for given data based on their class labels, we follow the following steps. First we consider the subset of Data that have same class label and for each of them we find first component direction. Then we calculate bisector between two vector obtained for each class and it gives us for most cases the best rotation angle. The obtained angle will be used to transform data to the new coordinate and in new coordinate, the comparing threshold will be find. The above procedure consist some more details that will be stated explicitly in sections 3 and 4.

The efficiently of this approach has be shown with implementation and comparing results with regular ID3 algorithm considering the depth of the tree.

As far as I review the literature briefly, I have not seem any method with similar approach for combining ID3 and PCA. There are many approaches, they use PCA to reduce the dimension of feature and then use new feature in decision three such as [2], [3]. Nonetheless, what we are doing in this paper is different in the sense that we use PCA to find first component of each class data rather than reducing dimension with linear combination of features.

The rest of this paper is outlined as follow. First clear problem formulation is given in section 3. Then in section 4, solution to the problem is suggested. Section 5 shows the implementation and results of the proposed method and finally we conclude the paper in section 6.

II. Problem Setup

We are considering binary classification for continuous-value features using decision tree with ID3 algorithm. Our desire is to extend ID3 algorithm for increasing its efficiency in depth sense. One assumption that is made here is our data has only two class label. we will refer to traditional ID3 algorithm, as regular ID3 algorithm and to our new approach, extended ID3 algorithm.

In the rest of this paper for sake of simplicity of presentation, we have assumed that we have two continuous features. However, the provided result are general in sense of number of features and can be extended for as many features as we want.

Consider a dataset with two features (x_1^j, x_2^j) and class label $y^j \in \{-1, 1\}$ where $j \in 1, \cdots, M$. We are interested to use extended ID3 algorithm for classifying samples based on their class label. In extended ID3 algorithm we will be allowed to use linear combination of feature for making decision in each node to decision tree. That means each decision boundary is as follow:

$$a_1 x_1 + a_2 x_2 = b \tag{1}$$

Fig 2 shows are a sample decision node of extended ID3 algorithm.

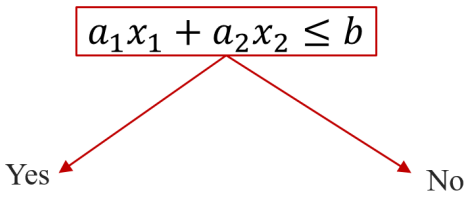

Fig. 2. Decision boundary in a sample node of the extended ID3 tree with two continuous features

Having decision boundary in above form, make us able to reduce the depth of the tree significantly when the data distribution is naturally separable with linear combination of features (Fig. 1). Now the question is that, how to find the constant a_1, a_2, b in each node. In traditional ID3, the decision boundary is function of one feature. On that case, we use the concept of Entropy to find a decision boundary that minimize the entropy. In extended ID3 algorithm, even for moderate number of sample, the number of boundaries that can divide the data is too big and it is impractical to find and check all of them. So we need another approach rather than exhaustive search between all possible value of a_1, a_2, b that divide our samples.

On way to reduce the complexity of above algorithm is to collapse it in two steps. In first step we rotate the coordinate system by appropriate

angle θ and in second step, we deploy regular ID3 to find the exact line that divides of samples with minimum entropy. Suppose for now that we know the desired value of θ angle and let complete our formulation from here. Figure 3 shows the rotation of coordinate by θ. The coordinate of point (x_1, x_2) in new coordination system is (x'_1, x'_2) where:

$$\begin{bmatrix} x'_1 \\ x'_2 \end{bmatrix} = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (2)$$

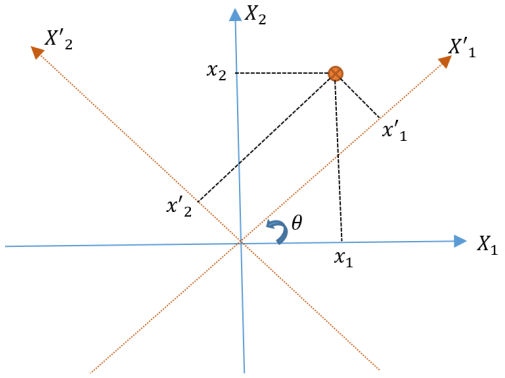

Fig. 3. New coordinate of a point under coordinate rotation around the origin

After obtaining the coordinate of all sample in we can execute regular ID3 in Data and from there find the comparing threshold. Pay attention that a_1 and a_2 are only function of θ and b depends on both θ and minimum entropy value boundary that is obtained in (X'_1, X'_2) coordinate.

for the case that decision boundary in new coordinate is parallel for axis X'_2 (Figure 4), we have:

$$a_1 = \frac{-tan(\theta + \frac{\pi}{2})}{r}, a_2 = \frac{1}{r} \quad (3)$$

$$b = \frac{-b_1 cos(\theta) tan(\theta + \frac{\pi}{2}) + b_1 sin(\theta)}{r} \quad (4)$$

where r in normalizing value such that $a_1^2 + a_2^2 = 1$ and b_1 is comparing threshold in (X'_1, X'_2) coordinate demonstrated in Figure 4.

On the other hand, for the case that decision boundary in new coordinate is parallel for axis X'_1 (Figure 5), we have:

$$a_1 = \frac{-tan(\theta)}{r}, a_2 = \frac{1}{r} \quad (5)$$

$$b = \frac{b_2 sin(\theta) tan(\theta) + b_2 cos(\theta)}{r} \quad (6)$$

where r in normalizing value such that $a_1^2 + a_2^2 = 1$ and b_2 is comparing threshold in (X'_1, X'_2) coordinate demonstrated in Figure 4, 5 in both possible directions.

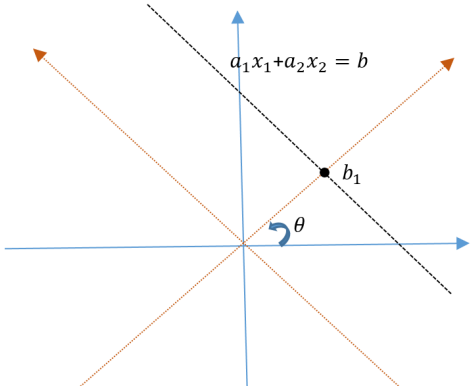

Fig. 4. The minimum entropy boundary in rotated coordinate when comparing threshold b_1 is on axis x'_1

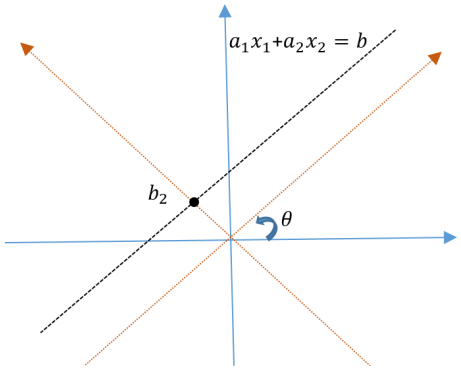

Fig. 5. The minimum entropy boundary in rotated coordinate when comparing threshold b_2 is on axis x'_2

So if we know the appropriate rotation angle θ, we can easily find variable a_1, a_2, b using regular ID3 and a little geometric. Now the main question is how to find the best value for θ. I tried many approach for that. Exhaustive search is the simplest solution but it fails when number of features increase. Estimating θ based on minimum entropy for horizontal and vertical decision boundary is tried but the results were not satisfying. Finally I find PCA a good tool to estimate the appropriate θ angle. In following section, we will to through this approach for finding θ and in section 5 the implementation of this method is shown.

III. Solution Approach

As explained in previous section, finding parameters of decision boundary in extended ID3 algorithm can be collapsed to finding appropriate θ rotation angle and using regular ID3 for finding comparing threshold. The main question is that how we can find θ. In this section we want to explain how to find θ using principle component analysis PCA.

PCA find the linear combinations of features that along which a given data set has most variation. First principle component is the line that has minimum distance from all sample. We want to use this property of first principle component to find the slope of decision boundary.

To do so, we divide the given data into two subset based on their class labels (we have assume that we have only two class labels) calls S_1 and S_2. For each subset, we need to normalize the data in order to make the mean value of each subset equal to zero. Pay attention we do not normalize the variance here. Then, first principle component for subsets are founded that calls V_1 and V_2 respectively. The slope of the appropriate θ is equal to slope of bisector between V_1 and V_2. That make sense because the directions V_1 and V_2 are the lines that samples of subset S_1 and S_2 have minimum distance from them. Therefore, their bisector is a good choice for separating samples of S_1 and S_2. One point to pay attention is that there is four bisector between two lines V_1 and V_2. However, it our algorithm, it make no different to choose with of them. That is because we will look for minimum entropy in each horizontal and vertical line in new coordinate and if we choose any of four bisector, the sign and name of the axises would be different and minimum entropy is same and when we find a_1, a_2 and b, all of them result is same values. Practically, we will always choose $0 \leq \theta \leq \frac{\pi}{2}$ for simplicity. After finding θ we will rotate the coordinate of features and by regular ID3 in new coordinate we will find a_1, a_2 and b using formulation provided in section 3. This procedure will be continued until all of the leaf of the tree is pure. We probably want to do pruning to vanish the effect of outliers. Figure 7 show the proposed algorithm. In next section, we will implement this approach to see how it deal with different subsets.

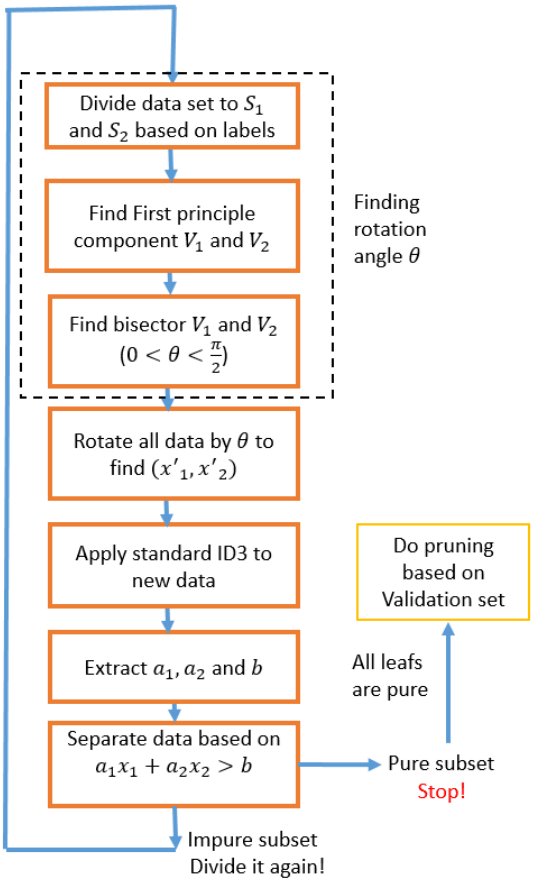

Fig. 6. Algorithm of extended ID3

IV. Experiment and Results

In this section, we want to present the results of implementation of algorithm that introduced in section 3 and 4. I used Matlab 2017 for implementation of extended ID3 algorithm. The scripts for that project is submitted beside this report.

We want to see the performance of extended ID3 compare to regular ID3 especially considering depth of tree. To analysis the performance, I want to analysis the performance in different cases, so I prefer to find a way to generate my own training and validating set randomly based on underlying structure. To find training data, I select random values for pair $(x_1, x_2)^j$ and from the underlying structure, I determined y_j where $j = 1, \cdots, M$ is the number of sample. That means y_j is a function of x_1^j and x_2^j. Also we introducing noise with filliping the class label of some sample with ratio α. Also the validation data is obtained using same procedure that results to $(x_1, x_2)^i$ and y_i for $i = 1, \cdots, N$. Four underlying structure has been

selected from simple to sophisticated ones. The result of those samples can give us good perspective to the performance of proposed algorithm. The list of Dataset with there underlying structure is shown in figure 7. Figure 8 shows some samples from that distribution that has been used for training the classifier.

Beside above Datasets, it seems necessary to compare between the algorithms with some real Dataset. So I use one Dataset from UCI Machine learning repository called Blood Transfusion Service Center Data Set [4]. Features of this data set are continuous. I select two features that first of them is total number of donation and another one is number of month from first donation and the binary output of the classifier is if the person will donate in fall 2007 or not. For that set, 300 sample are used as training set and 100 samples are used as validation set. We refer to that Dataset and Dataset 5 in rest of this section. We have provide the results for two cases with and without pruning.

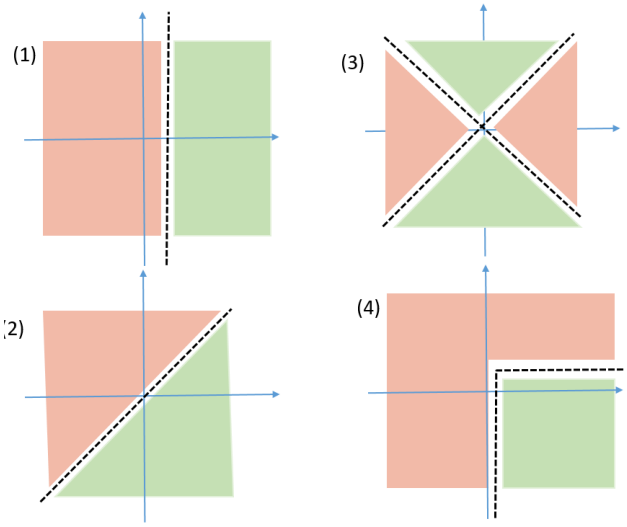

Fig. 7. Underlying structure of artificial Dataset that is used to train the regular and extended ID3, green shows positive and red shows negative class label

The results of experience with $\alpha = 0.05$, $M = 300$, $N = 100$ and without pruning are illustrated in Table 1. Also the results of experience with $\alpha = 0.05$, $M = 300$, $N = 100$ and with pruning are shown in Table 2.

As can be seen for dataset 1, the results are mostly same. For Dataset 2, as we expect, the result are so good. As can be seen the depth of tree

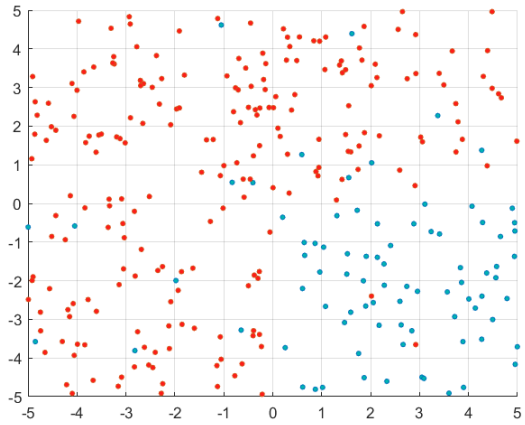

Fig. 8. Some drawing samples from underlying distribution 4 with $\alpha = 0.05$

Regular ID3 without pruning			
Dataset	Training Accuracy	Validating Accuracy	Depth
1	100	94	15
2	100	82	11
3	100	77	12
4	100	87	11
5	100	61	15
Extended ID3 without pruning			
Dataset	Training Accuracy	Validating Accuracy	Depth
1	100	86	10
2	100	94	11
3	100	82	10
4	100	88	11
5	100	65	16

TABLE I

REGULAR AND EXTENDED ID3 WITH $\alpha = 0.05$ AND WITHOUT PRUNING

reduced slightly in extended ID3 for the case with pruning. For dataset 3, yet extended ID3 has better performance. Dataset 4 is designed in the way that it is it is naturally separable with vertical and horizontal lines. In that case, regular ID3 has better results. That is because, in each step, extended ID3 do a greedy choice that is might not the best. For testing the proposed algorithm, it seems necessary to test it in the real data set, so I use [4] to test the performance of extended ID3 in comparison with regular ID3. As can be seen in tables 1-4, extended ID3 leads to just slightly better results. I think that highly connected to nature of data so maybe for some other dataset, it leads to the better results.

Regular ID3 with pruning			
Dataset	Training Accuracy	Validating Accuracy	Depth
1	98	96	14
2	94.67	90	8
3	93.66	86	11
4	94.667	97	2
5	79.2	75	12
Extended ID3 with pruning			
Dataset	Training Accuracy	Validating Accuracy	Depth
1	94.33	94	7
2	96.667	96	6
3	95.667	95	7
4	96.333	91	9
5	79.6	77	11

TABLE II

REGULAR AND EXTENDED ID3 WITH $\alpha = 0.05$ AND WITH PRUNING

V. CONCLUSION AND FUTURE WORK

In this research, we consider the extension of ID3 to be able to make decision based on linear combination of features rather than focusing on only one feature value in each decision. The algorithm is presented for continuous feature dataset, but they can easily be extended for discrete-value features by assigning a real value for each category. Also about extension for multi-class classification, we can use 1 vs. all and 1 vs. 1 paradigms. Also probably one other thing that might leads to better results is using Adaboost and bagging.

If I want to give some suggestion to improve the algorithm, one would be replacing bisector between two first components with a line that decline to the first component that has bigger explained variance with first component. That means If samples in one class is closer to the one of the first principle component, the rotation angle θ must be closer to it.

One underlying assumption that we has done with using PCA is than the data in each class has distribution along the optimal decision boundary. That might not true in all cases. I think using variance normalization before using PCA we can overcome this problem, but formulation becomes really hard on that case and need more time to interpret the results for that case. But I believe we can modify the results if we use variance explanation with first principle and variance normalization and put the effect of normalization on the rotation angle θ in an appropriate way.

REFERENCES

[1] James, Gareth, et al. An introduction to statistical learning. Vol. 112. New York: springer, 2013.

[2] Hu, Juanli, Jiabin Deng, and Mingxiang Sui. "A new approach for decision tree based on principal component analysis." Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on. IEEE, 2009.

[3] Popelnsky, Lubos, and Lubo Popel. "The principal components method as a pre-processing stage for decision tree learning." (2000).

[4] Online resource, UCI Machine Learning Repository https://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center