CS550 Homework 3

Berat Biçer

December 20, 2019

Part 1: Overview

In this homework, I train a decision tree classifier using Gini impurity as splitting criteria with samples weighted inversely proportional to class frequencies in the input data. For this classifier, I use *sklearn* library implementation. The reason for choosing the decision tree as my classifier is I was unable to normalize the input data. As the dataset contains both binary and real-valued features, normalizing the dataset would result in invalid feature values (such as a binary features having non-binary values). Moreover, even normalizing real-valued features introduces a bias as a classifier may assign different weights to the normalized features. A decision tree, however, is expected to work nonetheless on such a dataset and is faster to train due to the size of the dataset. Training time needs to be as small as possible since I will be training many classifiers in this assignment.

Using the entire dataset, the classwise test accuracies obtained are as follows:

Table 1: Classwise test accuracies

Class 1	Class 2	Class 3
0.904	0.806	0.986

Our joint cost function has three parts: the classification accuracy and feature selection cost. We're maximizing classwise classification accuracies while minimizing feature cost and variations in classwise accuracies. Feature selection cost is defined as the sum of costs of used features. Classification accuracy score is calculated as a weighted sum of classwise accuracies: first two classes have a weight of 1 where the third one has a weight of 0.5, implying the significance of correctly classifying the first two classes. Variation in classwise accuracy score is a weighted sum of absolute value of difference between classification accuracy of each pair of classes with a Weight of 1. Joint classification cost is a weighted sum of these three scores where weights are 0.0175 for feature cost, 1 for variation score in classwise accuracies, and -2 for classification accuracy score.

For stopping criteria, we train the network until the joint score stops decreasing. This happens when the cost of using additional features outweigh the increase in classification accuracy. At this point, the training stops and we report the final feature set, along with the cost obtained.

Part 2: Forward Selection Algorithm

Implementation Details

1. The algorithm is as follows: (1) We start with an empty set of features. (2) Looping over available features that are not being used already, generate a new feature set as the union of this feature and the current feature set. (3) Train a decision tree classifier using the new feature set. (4) Compute joint cost with accuracy and variation scores calculated on train set. (5) After having gone through every available feature, find the feature set that gives the lowest joint cost and set this as the new current feature set. (6) Repeat from 2 until the joint cost stops decreasing.

2. To prevent overfitting, instead of evaluating the classifier on test set, we apply two major regularization techniques: firstly, we limit the number of samples required to split an internal node to 3. Next, we limit the maximum depth of the tree to 8. The results shared below show that these techniques yield high accuracy on test set and prevents overfitting (since train accuracies are far from 1, which is the case if we don't apply these regularization techniques).

Results

1. Training takes 2 epochs. In first epoch, **TSH** feature is selected. The results are provided in the table below.

Table 2: Intermediary results after selecting **TSH**

Overall Accuracy	Class 1 Accuracy	Class 2 Accuracy	Class 3 Accuracy	Joint Cost
0.970	0.882	0.880	0.977	-4.002

2. In second epoch, **TT4** feature is selected. The results, (obtained as described in epoch 1), are as follows:

Table 3: Intermediary results after selecting **TT4**

Overall Accuracy	Class 1 Accuracy	Class 2 Accuracy	Class 3 Accuracy	Joint Cost
0.980	0.978	0.953	0.981	-4.150

3. At this step, algorithm terminates. The final joint cost on test data is -4.150. The classification accuracies are as follows:

Table 4: Final results after feature selection.

Type	Overall Accuracy	Class 1 Accuracy	Class 2 Accuracy	Class 3 Accuracy
Train	0.980	0.978	0.953	0.981
Test	0.974	0.959	0.966	0.975

Part 3: Genetic Selection Algorithm

Implementation Details

1. Our population consists of binary strings where 1 at position i indicates feature i is used in classification, and 0 indicates otherwise. Initially, we randomly sample 2500 individuals from the set of all binary strings of length 21 (number of all features) having at most five 1s as our starting population. The number of fittest individuals directly selected for the next generation is 500, number of parents that will be mated is 2000, and the number of individuals that will be subjected to mutations is 500. We employ the same joint cost function and stopping criteria as described in Part 1.

2. The complete algorithm, after initialization, is as follows: (1) We calculate the fitness score of each individual in the population and sort the individuals in ascending order with respect to these scores. (2) Fittest first 500 individuals survives directly to the next iteration. (2) Randomly selected 2000 individuals are mated such that at each mating period(once per select-breed-mutate epoch) 5 cross-over points are randomly selected (these points are constant throughout the mating period). The parents die and replaced with the offspring for the next generation. In case of faulty breeding, the children die and the parents survive. (3) Randomly selected 500 individuals from the next generation are subjected to mutations such that randomly selected 5 bits are reversed. These bits are constant throughout the current mutation period(again, once per select-breed-mutate epoch). In case of faulty mutations, the children die and the parents survive. (4) Algorithm continues until the joint cost of the fittest individual stops decreasing.

3. Note that by randomly selecting the crossover and mutation points, we insert stochasticity into the evolutionary model. Also, crowding is prevented by randomly selecting the parents for breeding, as fittest individuals can mate with poor-performing ones. Lastly, the final results vary with each run since the model is stochastic.

Results

Selected features are **age, query_hyperthyroid, goitre, TSH, TT4**. Final joint cost on test set is -4.117. The final accuracies are as follows:

Table 5: Final accuracies after feature selection.

Type	Overall Accuracy	Class 1 Accuracy	Class 2 Accuracy	Class 3 Accuracy
Train	0.980	0.968	0.958	0.982
Test	0.975	0.973	0.955	0.976

Part 4: Algorithm Comparison

Forward selection gives the same result at each run since there is no stochasticity in the algorithm where results of genetic selection can vary with each run since genetic selection is not deterministic (due to stochasticity in selection, mating, and mutation steps). Genetic selection is slower than forward selection since the entire population of individuals are processed at each epoch, and this process is proportional to the size of the population, resulting in the tradeoff where a larger population gives a broader range of values to search and is expected to find better results but it takes significantly more time to finish compared to the case when the population is smaller. Forward selection, however, has the tradeoff that it finishes relatively faster than the genetic search but it follows a greedy approach in selection, and as a result, unlike to genetic search, cannot select orthogonal features if they perform poorly individually (note that genetic search is not guaranteed to do so, but it can). Lastly, there is also a tradeoff between feature cost and model performance: if we penalize feature cost severly, the model avoids selecting highly informative features (which happen to be expensive); yet the opposite might be impractical (due to high cost of diagnosis).