Middle East Technical University Department of Computer Engineering

CENG 498

Introduction to Machine Learning

Spring 2017-2018

Homework 2 - Decision Tree

Due date: 29 04 2018, Sunday, 23:59

1 Introduction

The objective of this assignment is learn the details of decision trees and support vector machines (SVM) by implementing classifiers. The homework consists of two parts. In the first part you are going to implement classifiers using decision trees, draw their structure and determine their accuracy. You will create these trees using the ID3 algorithm on the training part of a given dataset. After their creation, you will then draw their shape on a graph and evaluate its accuracy on the test part. You will implement your decision trees using python and you are not allowed to use any non-standard library. After the implementation, you should draw the resulting trees on the report file and report their accuracy. In the second part of the homework, you will use a library to create various SVM classifiers with different kernel functions and tune their hyperparameters to improve their accuracy. In this part, you will use scikit-learn library in python to create your SVM with various kernels, examine and report their results in your reports.

2 PART 1

You will create five decision trees on car evaluation dataset using ID3 algorithm and draw their shape. For the first three decision trees, only your heuristic function should change and the last two trees should be pruned according to the specifications. Finally you will report classification accuracy of each tree. The details are given in the following subsections.

2.1 Dataset

In these dataset, you will be given 6 discrete attributes and 1 class attribute. Using the 6 attributes about the cars, you will create your decision tree. You can examine the details from `https://archive.ics.uci.edu/ml/datasets/Car+Evaluation`. The dataset is divided into training and test parts and converted into python arrays. These arrays are encapsulated using pickle and will be provided to you for training and test results. Use the following code to load the dataset:

```
import pickle

data = pickle.load( open('car.pkl', 'rb'))

train = data['train']
test = data['test']
```

These arrays are explained below:

- `train` is the training data array of shape (1150, 7) where first 6 are attributes and final value is the class. All values are in string format.
- `test` is the test data array of shape (578, 7) with similar composition.

Possible values for each attribute is given below in their corresponding order:

- buying: vhigh, high, med, low
- maint: vhigh, high, med, low
- doors: 2, 3, 4, 5more
- persons: 2, 4, more
- lug_boot: small, med, big
- safety: low, med, high
- class: unacc, acc, good, vgood

2.2 Trees and Report

You will implement 5 different decision trees using ID3 algorithm with three different heuristic (or classifier) functions and two pruning methods using the same heuristic function. These functions should be used to select the attributes to create a split. There will be no pruning in the first 3 trees. Three functions you are going to use are given below:

- Information Gain:

$$E(S) = -\sum_{c} p_c \times log_2(p_c)$$

$$Gain(S, A) = E(S) - \sum_{i} \frac{|S_i|}{|S|} \times E(S_i)$$

  where S is the samples, A is an attribute, p_c is the proportion of examples for a given class and (S_i) is the subset of samples for a given i^{th} value of an attribute. These values are used in other function definitions as well.

- Gain Ratio:

$$IntI(S, A) = -\sum_{i} \frac{|S_i|}{|S|} log(\frac{|S_i|}{|S|})$$

$$GR(S, A) = \frac{Gain(S, A)}{IntI(S, A)}$$

- Gini Index:

$$Gini(S) = 1 - \sum_c p_c^2$$

$$Gini(S, A) = \sum_i \frac{|S_i|}{|S|} \times Gini(S_i)$$

First two functions are to be maximized and gini index should be minimized for attribute selection.
The last 2 trees should have Gain Ratio as their heuristic function. In the first tree, you should use the chi-square pre-pruning method[1] and the reduced error post-pruning method[2, 3] in the second one. To use post-pruning method, you need to create a validation subset from the train portion of the dataset. You should create a validation set with $2:1$ ratio for each class (2k training, 1k validation size). After the creation of this subset, you will use the post-pruning method to simplify your tree. After your implementation, you should draw the final decision tree and put it in the given place provided in your latex template. Each non-leaf node of the tree should be labeled with an attribute and each edge should be labeled with values from that attribute. Leaf nodes should show the probability distribution of that node for each class. There is no need to put classes with zero percent probability at leaf nodes.
Finally, you should test your decision tree using the test data. When testing, select the classes with highest probability when you have reached a leaf. Report the classification accuracy in your report for each tree.

3 PART 2

In this part, you will use scikit-learn library to create various SVM classifiers. You will use the SVM implementation already present in the scikit-learn library. Your task is to tune your hyperparameters to achieve best classification accuracy. You are going to work on two different datasets and given complete freedom over tuning. After you receive your results, you will explain your experiments and how you achieved your results. You also need to explain why certain experiments that you conducted succeded where others failed. Details of the datasets and what we expect from you in hyperparameter tuning is given in the following subsections.

3.1 Datasets

In this part of the homework you are going to work on seismic bumps and website phishing datasets.

3.1.1 Seismic Bumps Dataset

In these dataset, you will be given 19 attributes including the class attribute. Using the 19 attributes about the cars, you will train your SVMs. There are 4 discrete and 14 real attributes in this dataset. Discrete attributes are converted into discrete integers between $[0, n)$ where n is the number of values that attribute can have. The class distribution is fairly unbalanced and the data is not processed other than converted discrete attributes. You can examine the details of the attributes from `https://archive.ics.uci.edu/ml/datasets/seismic-bumps`. The dataset is divided into training and test parts for input and output, and later converted into python arrays. These arrays are encapsulated using pickle and will be provided to you for training and test results. Use the following code to load the dataset:

```
import pickle

data = pickle.load(open(seismic-bumps.pkl, 'rb'))

train_x = data['train_x']
train_y = data['train_y']
test_x = data['test_x']
test_y = data['test_y']
```

These arrays are explained below:

- `train_x` is the training data array of shape (1722, 18) where 18 attributes are either raw integer or floating point values directly from the data. All values are in floating point format.
- `test_x` is the test data array of shape (862, 18) with similar composition to training.
- `train_y` is the training output array of shape (1722, 1) with two different classes represented with an integer in range $[0, 1]$.
- `test_y` is the test data array of shape (862, 1) with similar composition to training.

3.1.2 Website Phishing Dataset

In these dataset, you will be given 10 attributes including the class attribute. Using these 10 attributes about websites, you will train your SVM classifiers. All attributes are converted integer attributes. The class distribution is slightly unbalanced. You can examine the details of the attributes from `https://archive.ics.uci.edu/ml/datasets/Website+Phishing`. The dataset is divided into training and test parts for input and output, and later converted into python arrays. These arrays are encapsulated using pickle and will be provided to you for training and test results. Use the following code to load the dataset:

```
import pickle

data = pickle.load(open(phishing.pkl, 'rb'))

train_x = data['train_x']
train_y = data['train_y']
test_x = data['test_x']
test_y = data['test_y']
```

These arrays are explained below:

- `train_x` is the training data array of shape (901, 9) where all values are integer values converted to floating point number.
- `test_x` is the test data array of shape (452, 9) with similar composition to training.
- `train_y` is the training output array of shape (901, 1) with three different classes represented with an integer in range $[-1, 1]$.
- `test_y` is the test data array of shape (452, 1) with similar composition to training.

3.2 Tuning and Report Expectations

We have several experimental and explanatory expectations from you during your tuning. They are listed below:

1. Did you preprocess the data? Why was it necessary or why it was not? Did you come by this conclusion with reason or experimentation?(You can use preprocessing package from verb|scikit-learn| for preprocessing.)
2. Did you do anything about the class inbalance? Why, why not?
3. Did you create a validation subset? How did you create it? Was it necessary, why or why not? If you did not create a validation subset, what is your optimization criteria?
4. Use all kernels present in the library in your experiments. What are their results? Why did you get the results that you did?

5. Experiment with as much of the hyperparameters as you can. Explain the range of values and intervals that you experimented with on these parameters. If you had not experimented with a variable, explain the reason. Try to explain as much as possible, why certain values of your hyperparameters achieved the results that they did.

6. Record the hyperparameters and all the steps that you had taken to achieve your best results. Note your final classification accuracy.

7.

4 Specifications

- The codes must be in python. You are not allowed to use any non-standard library for the first part. For the second part, scikit-learn is choosen for SVM implementations for convenience. You are free to use any library for the second part of the homework or implement it yourself. However, make sure that your implementation is correct if you decided to do it.

- You programs should be able to run on Ubuntu operating system as long as the it contains the necessary libraries. You are not required to print anything to the screen when running your codes. Good coding practices and commenting are highly encouraged. Try to make your codes as understandable as possible to avoid misunderstandings and avoid coming to objections unnecessarily.

- You are highly encouraged to draw your decision trees using scalable graphics and make them smaller to fit the page. For example, you can use Powerpoint to draw your tree and save it as a pdf file. You can then include the pdf file in your reports and scale it according to your needs. If you are not using scalable graphics, please make sure the details of your trees are visible and clear after zooming.

- Falsifying results, changing the composition of training and test data are strictly forbidden and you will receive 0 if this is the case. This does not include the creation of validation set. Your programs will be examined to see if you have actually reach the results and if it is working correctly.

- You have total of 3 late days for all your homeworks. For each day you have submitted late, you will lose 10 points. If your late days exceed 3, any homework you have submitted late will be 0. Your used late submission days will be displayed together with your homeworks on cow.

- Using any piece of code that is not your own is strictly forbidden and constitutes as cheating. This includes friends, previous homeworks, or the internet. This is especially critical for the first part of the homework since you are doing all the implementation. The violators will be punished according to the department regulations.

- Follow the course page on cow for any updates and clarifications. Please ask your questions on cow instead of e-mailing if the question does not contain code or solution.

5 Submission

Submission will be done via COW. You will submit a tar file called "hw2.tar.gz" that contain all your source code together with your report in a pdf format compiled from the given latex file. You are highly encouraged to separate your source code. You should include a README file to explain your folder structure and your python scripts.

References

[1] M. F. Zibran, "Chi-squared test of independence," *Department of Computer Science, University of Calgary, Alberta, Canada*, 2007.

[2] J. R. Quinlan, "Simplifying decision trees," *International journal of man-machine studies*, vol. 27, no. 3, pp. 221–234, 1987.

[3] N. Patel and S. Upadhyay, "Study of various decision tree pruning methods with their empirical comparison in weka," *International journal of computer applications*, vol. 60, no. 12, 2012.