CSC 425/525 Artificial Intelligence
Project 3: Applying Decision tree, SVM, and Genetic Algorithms

Semester: Fall 2020

Instructor: Dr. Junxiu Zhou

Introduction

The goal of this project is to apply the learned machine learning algorithms to solve problems. In this project, three machine learning algorithms need to be applied: decision tree, SVM classifier, and Genetic algorithm. You are required to use Python and its libraries (such as Scikit-learn) to implement these three algorithms. In other words, you don't need to write these algorithms from scratch. In the sample code provided, Python and Scikit-learn are used. By using the support libraries, you are expected to apply all these three machine learning algorithms to solve related problems/tasks.

Decision Tree (30 points)

In this sub-project, your task is to implement one of the common machine learning algorithms: Decision Tree. You will train and test a decision tree with the dataset the instructor provided.

1. Apply ID3 decision tree learning algorithm (15 points)

For decision tree, in this project, we use ID3 (***information entropy*** based) decision tree algorithm. It is particularly interesting for its representation of learned knowledge, its approach to the management of complexity, and its potential for handling noisy data. ID3 represents concepts as decision trees, a representation that allows us to determine the classification of an object by testing its values for certain properties/attributes.

2. Dataset (in the attachment folder)

The dataset we use is the Titanic dataset. We have split the date set into training set and test set stored in two csv files. Titanic_training.csv has 891 samples, Titanic_training.csv has labels for the samples named "Survived". Titanic_test.csv has 418 samples and has no label provided in the file. Each row in Titanic_training.csv or Titanic_test.csv representing a sample with 10 attributes of the passenger in addition to passenger id. These characteristics are: *Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin,* and *Embarked*. Each sample is also associated with a label provided in Titanic_training.csv or Titanic_test.csv. For more information about the dataset, please click https://www.kaggle.com/dmilla/introduction-to-decision-trees-titanic-dataset. (You can use this link page as reference, please don't copy and paste.)

Note that for the missing values of the attributes, you may assign an "unknown" value for them.

Pseudo code for the ID3 algorithm

```
function induce_tree (example_set, Properties)
begin
if all entries in example_set are in the same class
    then return a leaf node labeled with that class
    else if Properties is empty
        then return leaf node labeled with disjunction of all classes in example_set
        else begin
            select a property, P, and make it the root of the current tree;
            delete P from Properties;
            for each value, V, of P,
              begin
                create a branch of the tree labeled with V;
                let partitionv be elements of example_set with values V for property P;
                call induce_tree(partitionv, Properties), attach result to branch V
              end
        end
end
```

3. Evaluate and draw the binary decision tree (15 points)

Visualize the decision tree you just trained on the training set. If the tree is too big or too complicated, you can stop at depth 3 of the tree (The root node of the tree has a depth of 0).

Then answer the following questions:

i. Briefly describe the steps involved in applying the ID3 decision tree for this classification task.

ii. What is the total number of nodes in the tree?

iii. What is the total number of leaf nodes in the tree?

iv. What are the classification accuracies on the training set and the test set, respectively? For test set accuracy, please submit your results on the related Kaggle challenge page (https://www.kaggle.com/c/titanic) by following the instruction file at appendix, and then attach your results (screenshot) to answer this question.

(**Note that,** you may not need to write code to draw a diagram of the tree. But you do need to write code to answer the above questions.)

4. Decision tree sub-project bonus work (10 points)

Implementing the ID3 algorithm yourself. (You cannot use any external Python libraries, except the following third-party libraries. NumPy: for creating arrays and using methods to manipulate arrays; matplotlib: for making plots; pandas or other packages for visualizing data or graphs).

Support Vector Machine (SVM) (30 points)

A simple illustration of SVM is in Fig. 1. In Fig. 1, SVM is used to distinguish two class of data. The SVM in particular defines the criterion to be looking for a decision surface that is maximally far away from any data point. This distance from the decision surface to the closest data point determines the **margin** of the classifier. The **support vector** is just the points set used for determine the decision hyperplane.

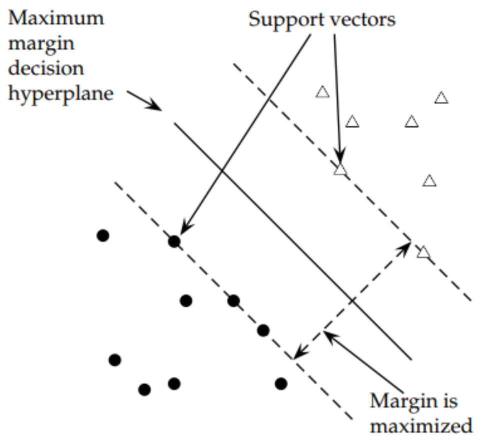

Fig. 1. A simple illustration of SVM

Detail can be referenced to text book and Chapter 15 of Introduction to Information Retrieval.

1. Apply SVM algorithm with the MNIST dataset (15 points)

For Support Vector Machine (SVM) classifier sub-project – use python libraries to train an SVM to classify the handwritten digits. The dataset we use for this sub-project is the MNIST dataset, i.e. mnist_subset.csv (in the attachment folder). There are total 10,000 samples in this subset. You need to split it into training set and test set based on your decision of the ratio value.

Fig. 2. Samples from the MNIST dataset

Note that, you may directly use the image raw pixel values as your classification features, since we did not cover any ***feature extraction content*** in this AI class.

2. Evaluate and answer the following questions (15 points)

Answering the following questions:

i. Briefly describe the steps involved in applying the SVM for this classification task.

ii. Train your SVM with increasing sizes of training set, say 10%, 20%, …. Test it with the test set. Make a plot to show how training and test accuracies vary with different ratios of the training and test split.

iii. Test your SVM model using different types of SVM kernels, then for different SVM kernels please training them using increasing sizes of training set, say 10%, 20%, …. Test the trees with the test set. Make plots to show how training and test accuracies vary with number of training samples for different kernels. (Note that at least two different SVM kernels should be tested.)

3. SVM bonus work (10 points)

Use pipeline and grid search to automatically optimize the parameters of SVM and run it on your own dataset or other dataset you like (such as the *Fashion MNIST* dataset) (10%).

Genetic Algorithm

Genetic algorithm is a kind of algorithm that mimic the natural producing of creatures in the world. The design of genetic algorithm is relatively casual (highly depend on individual).

1. Implement Genetic algorithm. (40 points)

Implement a genetic programming system as follows.

Write a system which can take a program's source code as input, make random changes to the program through genetic operations, evaluate the changes through a fitness function and select the best one(s) to become a parent for a new generation. The fitness function will first require that the child programs be compiled and then run on a collection of test data and evaluate the number of correct and incorrect outputs. Genetic operations can change the order of instructions, mutate assignment statements within a restricted. Amount (you can change + to *, etc. or change a variable to another variable of the same type) and you can cross-over same-typed instructions (e.g., two arithmetic expressions). Write a few partially functioning programs (ones that have the right set of instructions but perhaps not in the right order or with some instructions being incorrect) such that when run on the test data, they get a few of the test data correct but not a majority. See if your GP/EP system can evolve a correct program. Find cases where it works well and cases where it doesn't work well or at all.

NOTE: since do real time code execution in JAVA is a little tedious, you have to compile and then execute a string source program. Python is what we use here. As an example program, you might write a program that is supposed to perform sequential search but is slightly erroneous (logically) and see if you can evolve a working version.

1) Initialize the parents.
2) Producing off-springs.
3) Mutate and cross-over between parents.

   Though the instructor provides the example code, called genetic algorithm.txt, but you need to modify the following parts by yourself (to implement the logic that I mentioned in previous paragraph)

   - Crossover function
   - Mutate function
   - Fitness function
4) Substitute the parents with the off-springs

```
Procedure genetic algorithm;
begin
set time t:= 0;
initialize the population P(t), offspring size;
while the termination condition is not met do
begin
produce the offspring from P(t)
genetic operations:
    (1) Cross-over;
    (2) Mutation;
    (3) …
evaluate fitness of each member of offspring;
select members from offspring based on fitness;
replace, based on fitness, candidates of P(t), with these offspring;
set time t := t +1
end
end.
```

2. Genetic algorithm bonus work. (10 points)

Using JAVA to implement the genetic algorithms.

Hand-In Requirements

(1) The starter code is provided in Python language. You must follow the provided starter code to finish this project. There are two reasons for this:

  a. To prevent plagiarism;
  b. Reading and understanding other people’s code should be one the ability of CS students.

(2) If you completed this project, you need to hand in the **source code** been completed by you or your group. Your source code compressed to a single ZIP file. The name of the code archive should be **MLapplication_YourTeamname.zip.**

The code should be well commented, and it should be easy to see the correspondence between what's in the code and what's in the report. You don't need to include executable or various supporting files (e.g., utility libraries) whose content is irrelevant to the assignment. If the instructor finds it necessary to run your code in order to evaluate your solution, the instructor will get in touch with you.

(3) A project report in PDF format

- The report should briefly describe your implemented solution and fully answer all the questions posed above. **Remember**: *you will not get credit for any solutions you have obtained, but not included in the report*.
- All group reports need to include paragraph of individual contribution, i.e., which group member was responsible for which parts of the solution and submitted material. The instructor reserves the right to contact group members individually to verify this information.
- Describe the solution and answer the questions of each subproject.
- The name of the report file should be **MLapplication_YourTeamname.pdf**. Don't forget to include the names of all group members and the number of credit units at the top of the report. (*You can find the report template on the provided zip file together with the starter code.*)
- Finally, which part you think can be improved. Or the challenges you encountered during this project.

Note that, the instructor reserves the right to give **bonus points** for any advanced exploration or especially challenging or creative solutions that you implement. **If you submit any work for bonus points, be sure it is clearly indicated in your report.**

WARNING: You will not get credit for any solutions that you have obtained, but not included in your report!