EECS 461 / ECE 523 PA2
Path Finding – Is there a path?

Traditionally when we are faced with the task of determining whether a given maze has a path or not we resort to various search or path finding algorithms which are able to answer this question with absolute certainty. In this assignment, however, we will take a different approach to solve this problem by building a classifier that determines (to a certain degree) whether a given maze has a path or not.

Files included in this assignment

- skeleton_functions.py : skeleton functions are defined in this file. You need to write your codes in this file. Before you submit the assignment, change the name of the file as **<your first name>_<your last name>_ assignment2.py**.

  Please make sure you renamed the file name as **<your first name>_<your last name>_ assignment2.py**, otherwise your grade will be scaled by 0.5.
- training_set_positives.p : contains 150 mazes that have a path. Each training example in this file has a label of +1.
- training_set_negatives.p : contains 150 mazes that do not have a path. Each training example in this file has a label of -1.

Note

All files with a (.p) extension are in pickle format. Details on how to open these files will be provided below.

The setup for this problem is as follows:

- Your training set has 300 mazes in total
- Each training example is a 2 dimensional array where the green squares are represented as zeros and the black squares are represented as ones (see the figures below). For instance, the first three rows of the maze in Figure 1.1 are represented by the following array, [[0,0,0,1,0,1,0,0], [0,1,0,0,0,1,0,0], [0,0,1,1,0,0,1,1], ...]
- Your training set consists of two dictionaries (one for the positive examples and the other for the negative examples) that have these 2 dimensional arrays as their values.

In order to open the files which are stored in pickle format, you will use the pickle module as follows:

```
import cPickle as pickle
train_positives = pickle.load(open('training_set_positives.p', 'rb'))
# train_positives is a dictionary with the training examples that have a path
```

The same logic can be applied to the negative examples as well.
Below are two visualized instances of your training set.

Figure 1.1

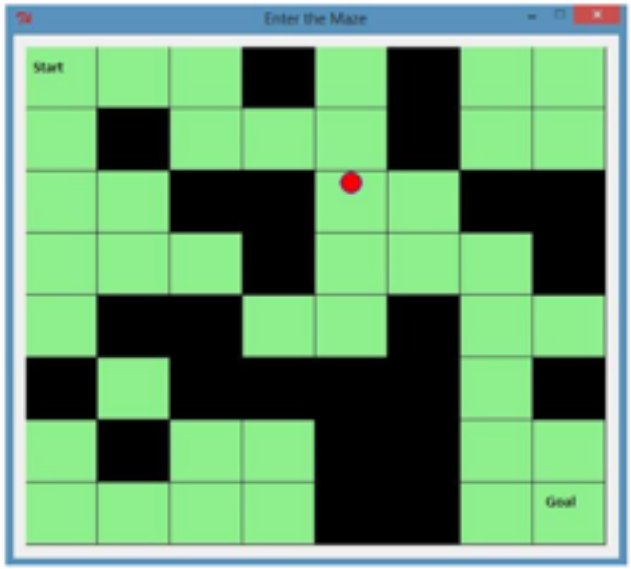
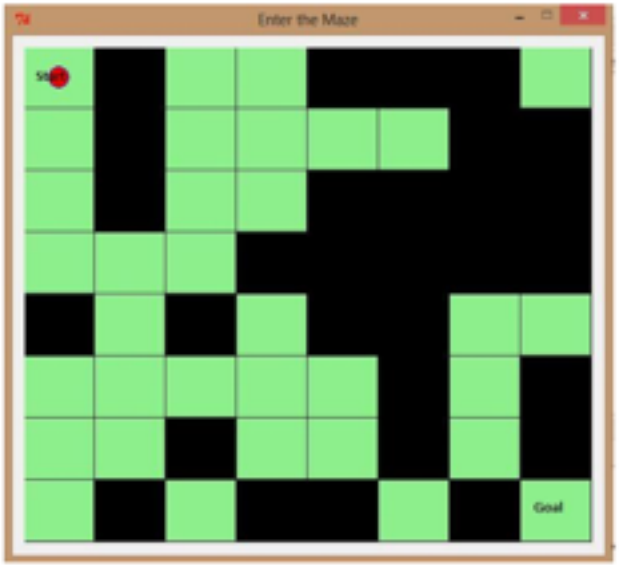


Figure 1.2



(Figure 1.1 shows an instance where there is a path from start to goal. Figure 1.2 shows an instance where a path from start to goal does not exist. The red circle denotes where you are in the maze.)
You have been provided with the python script skeleton_functions.py and within this script you have all skeleton functions defined for you.

a) Feature Extraction – 25 points
Your first task is to extract two features from the given dataset.

The first feature you will compute is the proportion of black squares to the total number of squares in the grid. You will write your code for extracting this feature within the function *feature1(x)*. If the input to your function is the maze in Figure 1.1 then the feature value that should be returned for that maze is 24/64 = 0.375.

The second feature you will compute is the sum over all the rows of the maximum number of continuous black squares in each row. You will write your code for extracting this feature within the function *feature2(x)*. In Figure 1.1 the maximum number of continuous black squares in the first row (from the top) is 1, in the second row 1, in the third row 2, and in the sixth row 4, etc. The value of this feature for this example is therefore the sum of these values, i.e., 1+1+2+1+2+4+2+2 = 15.

b) Preparing Data - 5 points
You will use training_set_positives.p and training_set_negatives.p files to create the training data. You should extract features for each grid in these files and put them into an X matrix and also prepare the corresponding label vector y. Keep the same order in training_set_positives.p and training_set_negatives.p (Put training_set_positives.p examples before training_set_negatives.p examples in the X matrix). X and y should be numpy array.

c) Classification with SGDClassifier - 10 points
You will built a SGDClassifier model with parameters random_state=0. Train this model with the training data that you created in part b. Write a function that uses your SGDClassifier to predict whether a maze has a path or not and return 1 or 0, respectively.

d) Assess the performance of the classifier in part c - 12.5 points
Compute precision, recall, and confusion matrix for the classifier in part c on the training set.

e) Classification with RandomForestClassifier - 10 points
Repeat part c with RandomForestClassifier.

f) Assess the performance of the classifier in part e - 12.5 points
Compute precision, recall, and confusion matrix for the classifier in part e on the training set.

g) Your Own Classification Model- 25 points
Now, prepare your own classifier model with the features of your own, and return the corresponding function as in part c and d. This is the creative part of your assignment where your grade will be determined by how well your classifier works.