CSC421: Assignment 3

Student: Jordan Yu, V00727036

Date: March 9th, 2015

Instructor : George Tzantakis

Table of Contents

[Question 1: Probability Theory 2](#_Toc413442970)

[Question 2: Email Categorization 3](#_Toc413442971)

[Model Definition 3](#_Toc413442972)

[Implementation 4](#_Toc413442973)

[Sample Emails 5](#_Toc413442974)

[Question 3: Bayesian Network 6](#_Toc413442975)

[Application and Data The application area chosen for the Bayesian network: 6](#_Toc413442976)

[Sample Queries 8](#_Toc413442977)

[Question 4: Learning and Decision Trees 10](#_Toc413442978)

[Full Decision Tree 11](#_Toc413442979)

[Information Gain Heuristic 11](#_Toc413442980)

[Weka 12](#_Toc413442981)

[Code Listing 13](#_Toc413442982)

[Question 2 Code 13](#_Toc413442983)

[Question 4 Code 17](#_Toc413442984)

[Data (Spreadsheets + ARFF) 19](#_Toc413442985)

[Training Data for Question 2 19](#_Toc413442986)

[ARFF File for Question 4 20](#_Toc413442987)

[References 21](#_Toc413442988)

# 

# Question 1: Probability Theory

To begin we determine the probability of winning and losing if we

1. Always Switch

2. Always DO NOT switch

The values can be calculated using Bayes theorem [1].

To calculate the probability of winning a car given that we always switch:

Door 3 has the car, the user has chosen door 1; therefore the host will always reveal door 2.

The car can be behind any of the 3 doors.

(1) (2) (3)

(1) If the car is behind door 1, then the host can show us either door 2, or door 3

(2) If the car is behind door 2, therefore the host will never show us door 2

(3) If the car is behind door 3, then the host will always show us door 2

To calculate the probability of winning a car given that we don't switch:

Door 1 has already been chosen by us, therefore the host only has 2 options; either door 2 or door 3

The car can be behind any of the 3 doors

(1) (2) (3)

(1) If the car is behind door 1, then the host can show us either door 2, or door 3

(2) If the car is behind door 2, therefore the host will never show us door 2

(3) If the car is behind door 3, then the host will always show us door 2

Therefore

Solutions to the two variants are therefore as follows:

Variant I

Variant II

# Question 2: Email Categorization

## Model Definition

Let C be the classification of the document.

Therefore

Let D be the document to classify.

Each document is represented as a vector

Each entry represents the presence/absence of a keywords

Therefore . Note the discreteness of the attributes.

The model we desire is the function

To determine this function we use a naive Bayes model to represent the problem. The naïve Bayes model allows us to assume that each keyword/attribute in the document is conditionally independent given the classification. We also make an assumption that the classification of a document is mutually exclusive.

To begin we investigate the simpler case of identify a document d given only a single classification.

This represents the probability that given document d the classification of the document is .

To classify for the more general case we calculate the probability that document D for all the classifications in C and choose the classification which has the highest probability.

Therefore the final function is:

We can drop because it is constant and does not affect the relative probabilities between the classifications.

We expand the document d into its vector form

We note that given our naive bayes assumption can be easily calculated

Therefore the model we wish to construct is the conditional probability table for each given

These can easily be calculated using the training data set.

Let n be the number of classified documents in the data set.

Let be the number of documents classified as in the data set.

## Implementation

The training data was created using my personal emails as samples.

The three categories chosen are:

**School** – Emails which relate to course work, or team projects. Tuitions and school events are also included in this category.

The keywords use for school email: [connex, uvic, assignment, student]

**Personal** – Emails which originate from family members, or are dealing which personal finances and services attached to the email.

The keywords used for personal email: [steam, programming, lily, peter]

**Professional** (Prof) – The category of emails which belong to correspondence with employers or businesses. This may include recruiter emails, interviews, and professional profiles.

The keywords used for professional email: [linkedin, coop, interview, university]

The main issues faced during the implementation:

1. How to deal with zero frequency keywords when calculating

The solution.

Add an extra factor when calculating the

Therefore becomes

This ensures that the weights from each probability will at-least greater than zero and the relative weights of each keyword does not change.

1. What do you multiply if the keyword is not present?

When calculating the if the keyword is not present in the document then we ignore the weight from the probability.

1. Tedious to classify all emails for the training set.

There was no clean solution for this problem. I just went through every email in my inbox, and extracted out the relevant keywords and classified into one of the three categories.

To use the classifier

|  |
| --- |
| b = Bayes();  b.setParameters({  "school" : ["connex","uvic","assignment","student"],  "personal" : ["steam","programming","lily","peter"],  "prof" : ["linkedin","coop","interview","university"]  });  b.train("training\_data.txt")  print(b.predict(“email”)) |

An example usage of the model, the source code and the training data used can be found under the code listings section of this report.

## Sample Emails

Five sample emails for each classification was generated using the model and keywords from the previous section. The resulting emails are listed here.

Classification: School

Email 1: assignment uvic connex steam student connex uvic programming connex connex

Email 2: assignment university student student assignment programming lily peter connex student

Email 3: university steam coop programming assignment programming connex connex uvic university

Email 4: connex student coop steam lily student uvic student university coop

Email 5: peter coop uvic assignment connex linkedin connex student uvic student

Classification: Personal

Email 1: university linkedin interview linkedin steam programming coop lily connex lily

Email 2: linkedin uvic lily steam interview interview lily programming connex programming

Email 3: uvic peter steam lily connex coop student steam programming lily

Email 4: uvic coop coop programming peter university lily coop steam peter

Email 5: programming linkedin peter lily lily steam connex steam steam student

Classification: Professional

Email 1: coop coop university coop coop connex interview linkedin assignment coop

Email 2: linkedin coop student lily university linkedin student coop student student

Email 3: uvic coop linkedin peter linkedin student peter uvic university linkedin

Email 4: linkedin interview coop linkedin linkedin interview interview student interview assignment

Email 5: lily university uvic connex coop coop coop linkedin linkedin coop

# Question 3: Bayesian Network

## Application and Data The application area chosen for the Bayesian network:

Cooling down oneself if one is overheated.

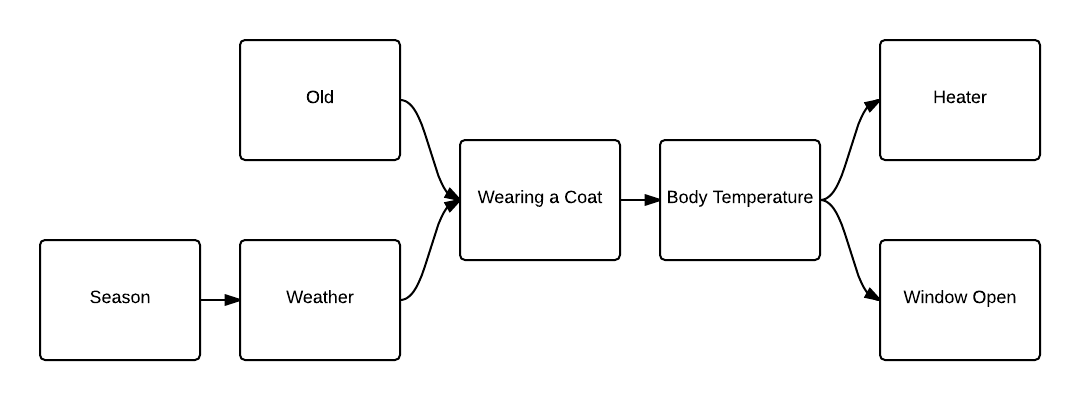


Figure : Topology of the bayesian network

Table : Conditional probabilities of the discrete variables

This is a discrete Bayesian network with the following random variables

The probabilities could be calculated by questionnaire.

For instance, sampling the weather given the season can be done through observation of past weather data. Samples for old or not old can be found by questioning young individuals, and asking how often they wear a coat in doors, and whether or not they turn on the heat, or open the window.

Sample Queries

For simplification let

Season = S Weather = WE Old = O

Coat = C Body Temp = B Window = WI Heater = H

The following are four sample queries that can be made using the network.

Calculations using exact inference by enumeration.

1.)

2.)

Calculations using variable elimination

3.)

4.)

# Question 4: Learning and Decision Trees

The application area chosen is described as such:

**Goal Predicate**: Should I attend class today?

**Attributes:** Type: Lecture, Lab, Tutorial

Midterm: T, F

Time of class: Early, Afternoon, Late

Enjoyable : Yes, No

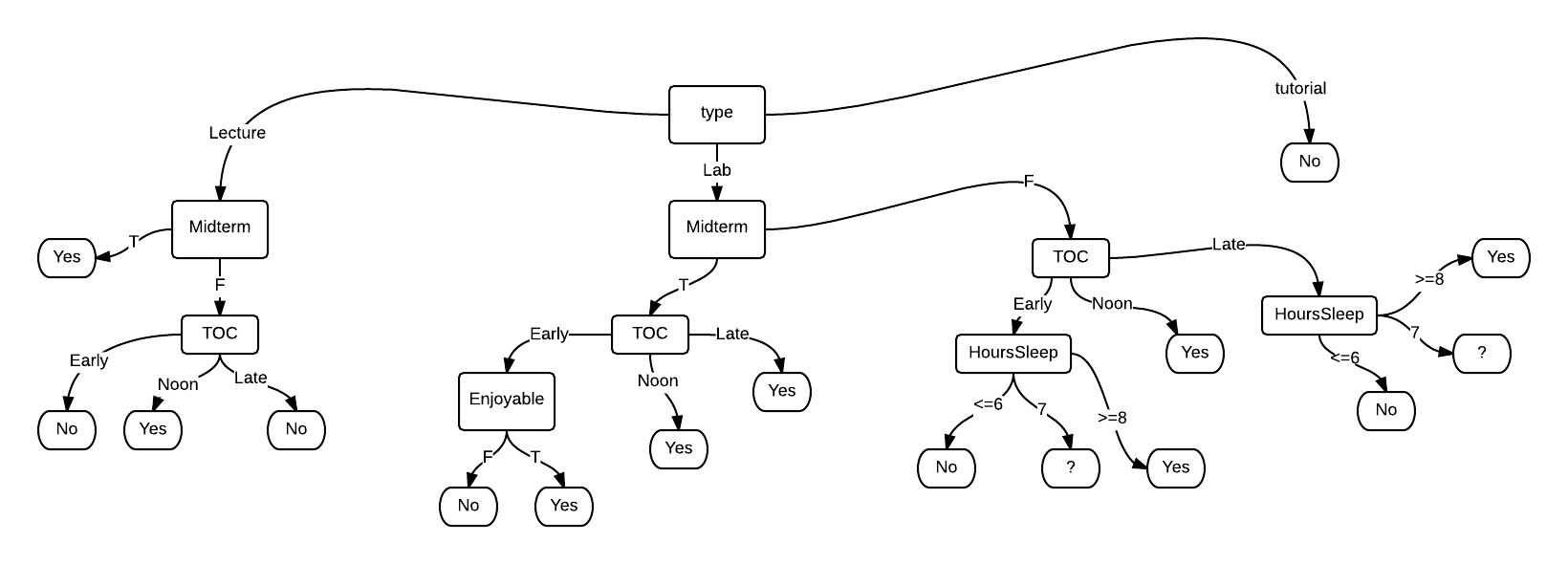
Hours of Sleep Last Night : <=6, 7, >= 8

The test data used for the decision tree is show below.



## 

## Full Simple Decision Tree



## Information Gain Heuristic

Using the information gain heuristic we apply an algorithm to choose the best attribute to use for each level of the decision tree. The heuristic informs us of the amount of entropy gained by choosing each specific attribute.

A program was written to calculate the Gain() from each attribute. This entropy gain is used to inform the decision tree which attribute would be most beneficial to check next. In this way a smaller decision tree can be created.

The following table shows the entropy gain on each attribute for the initial root node.

|  |  |
| --- | --- |
| Attribute Name | Gain |
| Enjoyable | 0.016529 |
| Time of Class | 0.032285 |
| Type | 0.396274 |
| Amount of Sleep | 0.235241 |
| Midterm | 0.052168 |

The highest entropy gain in this case comes from choosing the ‘type’ attribute.

This process is then repeated for each sub-tree of the graph.

Therefore the gain will need to calculated for the remaining attributes

|  |  |  |
| --- | --- | --- |
| Type = Lecture | Type = Lab | Type = Tutorial |
| Enjoyable | Enjoyable | Enjoyable |
| Time of class | Time of class | Time of class |
| Amount of Sleep | Amount of Sleep | Amount of Sleep |
| Midterm | Midterm | Midterm |

We can observe that *Type* and *Amount of Sleep* provide the most gain when creating the decision tree.

The full program used to generate this output can be seen under the Code Listings section, under Question 4 Code.

## Weka

Having run the training set through Weka, the following decision tree and evaluation output was created.

=== Run information ===

Scheme:weka.classifiers.trees.Id3

Relation: ATTENDCLASS

Instances: 30

Attributes: 6

Type

Midterm

Time

HoursSleep

Enjoyable

GoToClass

Test mode:10-fold cross-validation

=== Classifier model (full training set) ===

Id3

Type = Lab

| HoursSleep = <=6

| | Time = Early

| | | Enjoyable = T: T

| | | Enjoyable = F: F

| | Time = Noon: T

| | Time = Late: F

| HoursSleep = 7: T

| HoursSleep = >=8: T

Type = Lecture

| Midterm = T: T

| Midterm = F

| | Time = Early: F

| | Time = Noon: T

| | Time = Late: F

Type = Tutorial: F

Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances 24 80 %

Incorrectly Classified Instances 6 20 %

Kappa statistic 0.6

Mean absolute error 0.2

Root mean squared error 0.4472

Relative absolute error 39.5455 %

Root relative squared error 88.3807 %

Total Number of Instances 30

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure ROC Area Class

0.8 0.2 0.8 0.8 0.8 0.8 T

0.8 0.2 0.8 0.8 0.8 0.8 F

Weighted Avg. 0.8 0.2 0.8 0.8 0.8 0.8

=== Confusion Matrix ===

a b <-- classified as

12 3 | a = T

3 12 | b = F

Important observations from the classifier includes:

1. Classifier exhibited an 80% correctness percentage. This is a modest result given the small dataset. Looking at the confusion matrix we can see that for both Boolean cases the classifier was equally bad.
2. Looking at the decision tree we see that the top 2 levels of the tree use the attributes “Type”, and “HoursSleep” which conforms to the our expected attributes as measured by the entropy gains.

# Code Listing

## Question 2 Code

import sys

import random

from pprint import pprint,pformat

class Classification:

def \_\_init\_\_(self,classifier\_name,words):

self.classifier\_name = classifier\_name

self.prob\_classifier = 0

self.prob\_keyword = {}

for w in words:

self.prob\_keyword[w] = 0

def \_\_str\_\_(self):

return "'{}':{}\n{}\n".format(self.classifier\_name,self.prob\_classifier,

pformat(self.prob\_keyword))

def \_\_repr\_\_(self):

return self.\_\_str\_\_()

class Bayes:

def \_\_init\_\_(self):

self.classifiers = {"classification": ['keywords']}

self.probs = { 'classification': {'keyword':0.4}}

self.words = set()

"""

@param d - dictionary containing the classifications and keywords

for this naive bayes model.

Assumption is everything is lower case.

"""

def setParameters(self, d):

self.classifiers = d

# create a set of the keywords

self.words = set()

for c in d:

self.words = self.words.union(d[c])

# create Classification objects

self.probs = {}

for c in self.classifiers:

self.probs[c] = Classification(c,self.words)

"""

@param training\_data (string) - filename of training data.

the format is the following

<classification> <word>,...,<word>

"""

def train(self, training\_data):

f = open(training\_data,"r")

# temp data structure to hold the number of occurences of each

# classification as well the count on the number of words

data = {}

for x in self.classifiers:

data[x] = {

"count":0,

"words":{}

}

num\_entries = 0

# read through the file and fill in the counts

for line in f:

line = line.rstrip('\n')

words = line.split()

# error checking

if(len(words) <= 0):

continue

class\_type = words[0].lower()

words = words[1:]

num\_entries += 1

# increment a count on a class\_type

data[class\_type]["count"] += 1;

# go through all the words for this line and keep a count on them

for w in words:

w = w.lower()

if w in data[class\_type]["words"]:

data[class\_type]["words"][w] += 1

else:

data[class\_type]["words"][w] = 1

f.close()

# parse through data and calculate the probabilties

alpha = 1

for k,c in self.probs.items():

c.prob\_classifier = (float(data[k]['count']) + alpha)/(num\_entries + alpha\*len(self.words))

for w in c.prob\_keyword:

word\_count = 0 if not w in data[k]['words'] else (data[k]['words'][w])

c.prob\_keyword[w] = float(word\_count + alpha)/(data[k]['count'] + alpha\*len(self.words))

"""

@param doc(string)- the filename of the document to classify

@return (string) - string representing the classifcation of the document.

"""

def predict(self, doc):

# read in the file and determine the words that are present

f = open(doc,"r")

words = set()

for line in f:

line = line.rstrip('\n')

words = words.union(line.split())

f.close()

# For all the words present in the document

# take the product of all the probabilties for each classifcation

# and take the classification with the highest probability

max\_c = None

max\_prob = 0

for k,c in self.probs.items():

prob = c.prob\_classifier

for w in c.prob\_keyword:

if( w in words):

# word is present, multiply against the probability

prob \*= c.prob\_keyword[w]

# if the final probablity is larger than everything, then record it

if(max\_c == None or prob > max\_prob):

max\_c = k

return max\_c

def generate(self,class\_type,doc\_len=10):

# extract the array of prob,keyword pairs and sort

# by from lowest to greatest prob

vals = self.probs[class\_type].prob\_keyword.items()

vals = map(lambda x : x[::-1], vals)

vals.sort()

#normalize the probabilities nto the range 0,1

prob\_sum = reduce(lambda x,y : (x[0] + y[0],0),vals)[0]

vals = map(lambda x : (x[0]/prob\_sum,x[1]), vals)

# construct the string from the keywords based on the prob distribution

rs = ""

for i in xrange(doc\_len):

# randomly choose a keyword

r = random.random()

for v in vals:

if r <= v[0]:

rs += v[1] + " "

break

else:

r -= v[0]

return rs

b = Bayes();

b.setParameters({

"school" : ["connex","uvic","assignment","student"],

"personal" : ["steam","programming","lily","peter"],

"prof" : ["linkedin","coop","interview","university"]

});

train\_file = "q2\_training\_data.txt"

predict\_file = "q2\_training\_data.txt"

b.train(train\_file);

print("predict '{}' => {}\n".format(train\_file,b.predict(predict\_file)))

def gen(num,type):

print(type)

for x in xrange(num):

print(b.generate(type,10))

print("")

gen(5,"school")

gen(5,"personal")

gen(5,"prof")

## Question 4 Code

import sys

import math

from pprint import pprint

f = open('q4\_data.txt',"r")

attrs = {

'type':["lecture","lab","tutorial"],

'midterm':["t","f"],

'toc':["early","noon","late"],

'sleep':["<=6","7", ">=8"],

'enjoyable':["t","f"],

}

data = []

neg = 0

pos = 0

for line in f:

line.rstrip('\n')

words = line.split()

words = map(lambda x: x.lower(), words)

a = {

'type':words[0],

'midterm':words[1],

'toc':words[2],

'sleep':words[3],

'enjoyable':words[4],

'goal':words[5]

};

if( a['goal'] == 't'):

pos += 1

else:

neg += 1

data.append(a)

def parse\_for\_pk\_nk(data):

rs = {}

for i in attrs:

rs[i] = {}

for j in attrs[i]:

rs[i][j] = {

'pk': 0 ,

'nk' : 0

}

for row in data:

for attr in attrs:

val = row[attr]

if row['goal'] == 't':

rs[attr][val]['pk'] += 1

else:

rs[attr][val]['nk'] += 1

return rs

data\_pk\_nk = parse\_for\_pk\_nk(data)

def B(q):

if(q == 0):

return 0

else:

return -1\*(q\*math.log(q,2) + (1-q)\*math.log(1-q,2))

def Remainder(attr):

rs = 0

for k in attrs[attr]:

pk = data\_pk\_nk[attr][k]['pk']

nk = data\_pk\_nk[attr][k]['nk']

rs += (float((pk + nk))/( pos + neg ))\*B( float(pk)/(pk + nk))

return rs

def Gain(attr):

return B(float(pos)/ ( pos + neg)) - Remainder(attr)

for attr\_name in data\_pk\_nk:

print(attr\_name)

for attr\_type in data\_pk\_nk[attr\_name]:

print("," + attr\_type + "," + str(data\_pk\_nk[attr\_name][attr\_type]['pk']) + "," + str(data\_pk\_nk[attr\_name][attr\_type]['nk']))

for attr\_name in attrs:

print(str(attr\_name) + "," + str(Gain(attr\_name)))

# Data (Spreadsheets + ARFF)

### Training Data for Question 2

school connex assignment uvic seng student students

school uvss uvic students

school student uvic survey university

school graduating grad

school announcement connex class

school workterm reports coursespaces coop university victoria

school undergraduate university victoria v00727036 graduation

school worklog report team project

school uvic course survey instructor university victoria

school project office email ELW

school project lol ELW

school project group class team

school meeting project

school student university victoria

school midterm announcement connex

school connex announcement assignment

school connex announcement midterm grades

school midterm assignment grade

school assignment connex lecture

school assignment grade submission connex

school assignment submission grade v00727036 connex

school exam announcement connex

school grade report submitted

school connex assignment submission

prof linkedin coop engn undergrad connect

prof automatically

prof dear t4 employer regards

prof google recruiter coordinator sincerely

prof interview coordinators onsite questions

prof t4 payroll

prof engineers engn-ugrad student survey regards canada

prof phone interview candidacy hiring feedback

prof regards technical interviews

prof recruit recruiting student graduates software enigneering resume transcript

prof graduation application tuition uvic mypage student

prof workterm logbooks worksite

prof invitations linkedin profile

prof sin canadian passport coop intern employment

prof coop apartment

prof coop intern

prof shortlist interview intern student

prof coop intern engrcoop learninginmotion interview

prof bell bill e-bill

prof engn-ugrad university

prof reset password account

prof engn-ugrad engineering student uvic students

prof linkedin profile skill endorse

prof survey

prof linkedin congratulate network

personal stymphalian steam games game

personal family peter

personal steam game gift

personal programming

personal programming

personal lily

personal peter fatboy

personal steam game

personal school uvic tuition

### ARFF File for Question 4

% Weka data file for Question 4

@RELATION ATTENDCLASS

@ATTRIBUTE Type {Lab,Lecture,Tutorial}

@ATTRIBUTE Midterm {T,F}

@ATTRIBUTE Time {Early,Noon,Late}

@ATTRIBUTE HoursSleep {<=6,7,>=8}

@ATTRIBUTE Enjoyable {T,F}

@ATTRIBUTE GoToClass {T,F}

@DATA

Lab,F,Early,7,F,T

Lecture,F,Early,7,F,F

Tutorial,F,Late,7,F,F

Lecture,T,Late,7,T,T

Lab,F,Noon,7,F,T

Lecture,F,Noon,7,F,T

Tutorial,F,Noon,7,F,F

Lab,T,Noon,7,F,T

Lecture,T,Noon,7,F,T

Lab,F,Early,<=6,F,F

Lecture,F,Early,<=6,F,F

Tutorial,F,Early,<=6,F,F

Lab,T,Early,<=6,F,F

Lab,T,Early,<=6,T,T

Tutorial,T,Early,<=6,F,F

Lab,F,Late,<=6,F,F

Tutorial,F,Late,<=6,F,F

Tutorial,T,Late,<=6,T,F

Lab,F,Noon,<=6,F,T

Tutorial,T,Noon,<=6,F,F

Tutorial,T,Noon,<=6,T,F

Lecture,T,Early,>=8,F,T

Lecture,T,Early,>=8,T,T

Lab,F,Late,>=8,F,T

Lecture,F,Late,>=8,T,F

Lab,T,Late,>=8,F,T

Lab,T,Late,>=8,T,T

Lecture,F,Noon,>=8,F,T

Tutorial,F,Noon,>=8,F,F

Lecture,T,Noon,>=8,T,T

# References

|  |  |
| --- | --- |
| [1] | C. Long, "The Bayes Solution to Monty Hall," 28 June 2012. [Online]. Available: http://angrystatistician.blogspot.ca/2012/06/bayes-solution-to-monty-hall.html. [Accessed 4 March 2015]. |
| [2] | P. N. Stuart Russel, Artificial Intelligence : A Modern Approach Third Edition, Upper Saddle River, New Jersey: Prentice Hall, 2010. |