

# **Bayesian Decision Homework**

T. Nathan Mundhenk and Laurent Itti  
University of Southern California  
Department of Computer Science

Copyright (c) 2006 T. Nathan Mundhenk, Laurent Itti

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

## CS561 Homework 5

**Due: By class Tuesday, April 25, 2006**

### Guidelines:

This assignment has a written and programming component. You can see how much each part of the assignment is worth by the percentage next to it. For the written part, please turn in typewritten answers. You are not English majors, however blatant spelling and grammatical errors may cost you points, so be sure and run the spell check at least once. Answers on the written part will require justification and a simple yes/no answer will almost certainly garner you little or no points. In general, it is safer to write longer answers than shorter ones, but stay focused as a long but off-topic answer will not work either. This way, we can discern your train of thoughts better and grant partial credit for a wrong answer, if you were on the right track. Graphs must be neat and tidy. Hand-drawn graphs are OK, but computer-drawn graphs (e.g., made with Adobe Illustrator or Microsoft Power Point) are preferred. If you draw a graph by hand, be sure and use a straight edge (ruler) so it looks neat.

For the programming part, you will be provided sample inputs and at least one sample output. Since each homework is checked via an automated Perl script, your output should match the example format *exactly*. Failure to do so will most certainly cost some points. Since the output format is simple and there is an example on the web, this should not be a problem. Additionally, if your code generates a large number of warnings during compilation, you may lose points, so try and eliminate compile-time warnings. Additionally, your code should be well documented. If something goes wrong during compile and grading, if the fix proves easy, the amount of points lost will be far less. As such, documentation makes fixing easier, so it is to your advantage to do so.

You will be provided with a stub Perl script called “stubby.pl”, which works like the grading Perl script. You can run this script to make sure that your project will work properly. Thus, it is expected that your project will run through the grading Perl script without problems. Pedantically, we assert it will cost you points if your code does not run on the Perl script correctly. You will be able to tell if your output is correct if stubby shows each of the lines from your program output is exactly the same as from the comparison file which shows what output you should be getting.

To run stubby, copy it to your code directory along with the input, output and comparison files. Be sure to be in your code directory then type “perl stubby.pl”. The script is loaded with all sorts of output and feedback, so you should be able to see what it is doing if for some reason it isn’t working for you. If you don’t understand the script and want to know more about Perl, go to <http://www.perl.org/>.

Small amounts of extra credit (not more than 10% total) are given for all sorts of things. So long as you meet the base homework requirements anything creative, fun, interesting or outright cool

will most likely earn you extra credit. What is cool and earns you extra credit is somewhat subjective and bound to the whim of the grader.

### **Handing in the Assignment:**

Since the class is very large it is important to hand in your homework as directed. *The homework is due before class on the due date shown.* There are two parts of the homework, which you must hand in. These are:

- (A) Your code tar/gzipped in **one** file. Do not send the binaries. - This should just include your uncompiled code and a readme file called readme.txt. When I run tar/gzip to uncompress your file, it should uncompress into its own directory. To keep things uniform, do not use bzip2. I should then be able to run my stubby Perl script and grade your code. See below on how to tar/gzip your code.
- (B) Your written part on paper. *Iff* you are a DEN student, you may submit your written part in the DEN digital drop-box (remember your cover sheet). Otherwise, you must submit a paper copy either in class or in a drop-box in front of my office in HNB (NOTE: the HNB building closes at 5:00pm). Late submissions will be noted. Be sure your homework is stapled together and is generally tidy. Electronic submissions of the written part from non-DEN students will not be accepted.

Be sure that your name is *clearly visible* on all material you hand in. To hand in your code you can compress it with tar/gzip with a command like:

```
“tar -czvf my.name.hw5.code.tgz my.name.hw5.code”
```

You might also try:

```
“tar -cvf my.name.hw5.code.tar my.name.hw5.code” then type “gzip my.name.hw5.code.tar”
```

This will compress the contents of the directory named “my.name.hw5.code” into a single file my.name.hw5.code.tgz. If you need more info on this, try “man tar”.

### **Electronic Submission of code:**

You need to submit the assignment electronically via the DEN digital drop-box. Do this by accessing the DEN web page and then the DEN digital drop-box. Be sure to name your code with a name such as:

```
my.name.hw5.code.tgz
```

Be sure and substitute “my.name” with your own name. The reason for naming your submission is to make it much easier to match up projects with the students who submit them. Otherwise this task becomes difficult for the grader. When you have uploaded your code, be sure and click submit.

## **DEN students using electronic submission of the written part:**

For DEN students handing in the written part electronically, if you use Word files, please compress the file using .zip or .rar.. Don't worry about compressing Acrobat documents since they are already compressed. Thus, I should see two files if you are doing an all-electronic submission, one with a name like bob.bobbinopolis.hw5.written.zip and the other with a name like bob.bobbinopolis.hw5.code.tgz. If you send me a file with a name like hw5.zip, I will put a hex on your credit cards. Of course this does not apply to your .cpp/.h files which should have names that work with the stubby Perl script. Also, be sure your name is on the documents you hand in since we print them up to grade them and it can be hard keeping documents matched. Please make sure all images and material for the written part are collated into one single file. Thus, *do not* send a word document and 30 image files, be sure and place them inside the word document itself.

Additionally, DEN students should remember to include a *cover sheet* with their submissions.

### **What exactly is Stubby?**

*This is very important.* We use a Perl script to automate the grading process. Stubby is a Perl script we give to you so that you can check and make sure that your project conforms to specifications. That is, you can use Stubby to make sure that your assignment when handed in will run with our grading script. Thus, we have our own grading script *like* stubby, which we use to grade your project. By checking to make sure your project works with stubby, you make sure that your project will work with our automated grading script.

It is important to note that we will not use your version of stubby. We have our own script. As such, if you edit stubby to work with your code and not the other way around, this will not be very helpful. Additionally, since there are so many projects to grade, it is imperative that your program work with the grading Perl script. *If your program does not work with the grading Perl script you will lose points*

### Question 1 (20%)

You are a fabulously wealthy playboy staying at the ritzy Casino Cosmo Royal on Rigel V. You have decided to bet a large sum of credits on a game called O'lgleck. In it, you are connected up to machine, which interfaces directly with your brain. You then place your hand inside a green satin bag and draw out different pieces called Chu'nks with your hand. The skill comes in manipulating good Chu'nks closer to your hand using the brain interface. It's a very difficult game, which requires a great deal of intuition in order to feel the Chu'nks with your mind and move them around. In the game you can sort of feel a Chu'nk with your mind and grab at it and draw it out of the bag. In addition to there being a finite number of Chu'nks in the bag, there is always a probability of grabbing a "wrong" Chu'nk either because you feel it wrong, or you move the wrong piece with your mind into your hand somehow. Being a smart chap, you are very familiar with statistics, which you will use to help you play the game with more skill.

You have just drawn a Goo Chu'nk which is not the best draw, but if you can draw a Yu'mme chunk everyone around you will yell "O'lgleck Weeee" and you win the jackpot. There are 10 unique Chu'nks left in the bag. However, drawing out a Goo chunk causes great mental fatigue. So, given that you have drawn a Goo Chu'nk, there is only a 50% chance you can manipulate a Yu'mme Ch'unk into your hand if it is one of the 10 Chu'nks you try to manipulate. Otherwise, it slips out of your hand and your wind up drawing another Chu'nk instead. For all the other Chu'nks, they slip out with 0% probability in this situation. That is, they never slip out. The odd thing is that, in reality, the probability of drawing any Chu'nk under general circumstances is completely random. People just think there is some skill to it. This is part of how casinos make major CASH. Unfortunately, if you draw out any Chu'nk other than the Yu'mme chunk, in this situation you lose. So, this is an important draw.

(3 points each)

- (1) In this example, what elements are the prior, conditional and posterior probabilities?
- (2) Compute the posterior probability of being able to draw a Yu'mme chunk in this situation. Show your work of course.
- (3) If you find yourself in this situation 1000 times and bet 100 credits each time, with the jackpot being 10 times what you bet, how much money would you have at the end.
- (4) Compare your answer in (3) with how much money you would have if you just stuffed all your credits under a mattress. What is the ratio of money won over money stuffed in the mattress? How might you extend this to an idea of risk and cost?

On another try, you draw a Wizz'o' Chu'nk, which feels a great deal like a Wazz'o' Chu'nk. Because of this it's much easier to draw a Wazz'o' chunk because you are familiar with how it feels. There are 15 unique Chu'nks left in the bag and in this situation, you always have a 50% chance of feeling and grabbing the Wazz'o' Chu'nk regardless of the bags contents. Otherwise, without the Wizz'o' Chu'nk the probability of grabbing a Wazz'o' Chu'nk is equal to any other Chu'nk.

- (5) Why is the 50% chance of grabbing a Wazz'o' chunk in this example different than the 50% chance in the first Yu'mme Chu'nk example? Give an explanation that shows how Bayes rule computation differs in the two situations.

To answer the next question we review a few more rules of the game:

- a. Chunks are not placed back in the bag after a draw.
- b. When you draw the second Chu'nk (any Chu'nk). With the exception of a Wizz'o' Chu'nk, all probabilities are equal of drawing any Chu'nk after that.
- c. The first and second draw can never win, so you will *always* draw a third Chu'nk.
- d. Standard Rigel O'gleck always starts with 18 unique Chu'nks.
- e. Unless otherwise specified so far, the probabilities of drawing any Chu'nk are equal.

(5 points)

- (6) Given what you know of the current probabilities, if your first draw is a Goo Chu'nk. What is the probability that your third draw will be a Wazz'o' Chu'nk? Show your work.

## Question 2 (Coding 60%, Related Questions 20%)

Cyberdyne Systems manufactures a variety of robotic and industrial sorting machines. In many industrial settings, this is made easier by the fact that many items that need to be sorted are designed to make them easier to sort. So for instance, Agile Puppy Farms LLC places bar codes on all its puppies collars so that the Cyberdyne systems Insta RoboPupSort 2000 can instantly recognize a dogs correct breed from its bar code and make sure that for instance, poodles are correctly placed in poodle crates and not in a Pit bull crate.

Recently, Cyberdyne has received many requests from customers who need items to be sorted that are not feasible to be manufactured so that it is easier for a sorter to recognize them. For instance, Bobs Used Glass Emporium Inc. receives large volumes of used glass. They need the glass to be sorted into sharp pointy unsafe glass and smooth safe glass. Since this is an unsafe job for a human, it is an ideal candidate for the Cyberdyne Systems Pointy Thing 'o' Matic. Since the glass comes in shipped *as is*, there is no way to label or otherwise process the glass before the robot can sort it. The task in this case is made more difficult by the fact that it can be hard to determine some sharp pointy pieces of glass from safe smooth glass. That is, sometimes they look quite a bit alike.

Cyberdyne Systems can purchase a variety of sensors for its robots in order allow them to sense the features of objects which a customer wishes to sort. For instance, a Cyberdyne engineer has suggested Cuspidate Technologies third generation pointy thing detector PTD-3G. It uses a combination of computer vision and laser measurements to return the pointyness of an object. The problem with the PTD-3G and many other sensor devices that Cyberdyne uses is that in many cases they are still not accurate enough to tell 100% of the time one object from another. Your job as a crack engineer at Cyberdyne is to create a Bayesian sorter. You will be given samples of *features* from objects of different *classes*. You will then use those samples to formulate a Bayesian decision boundary on which to sort objects.

As an example, someone was brave enough to sort a truckload of used glass into sharp pointy glass and smooth safe glass. They then count how many samples are in each group and measure each piece with the PTD-3G. From these measurements, if you receive a new piece of glass you have never seen before, you will then be able to determine, given some reading on the PTD-3G, what type of glass it is most likely to be and have the robot sort it into the correct bin.

For this assignment you will implement a **Simple Sample based Bayesian Classifier**. You can assume that sample features are *normally distributed* and sampled *unbiased*. Additionally, you can use the *online tutorial* to help you implement this project. You will receive feature sample readings from different classes of objects and will need to make a classifier that can make a judgment about what class new objects belong to.

**Input:** You will receive feature samples of objects you will need to sort as well as feature samples from a second set of objects you will test your feature detector against. The first line is the type of item you will be sorting, the second line is the feature you will be keying off of. This is followed by classes and samples as such.

### **Example1.training.txt**

```
# Glass_bits
# Pointyness
# Sharp_unsafe
1.0
1.1
1.25
1.3
1.2
# Smooth_safe
0.8
0.7
0.56
0.8
1.1
# Sort_of_sharp
1.1
1.0
0.9
```

Notice that each class is noted by starting a line with a hash mark. Thus, after the first two lines, a hash mark denotes a new class. All sample values are numerical floating-point values as in this example.

You will then open a file with testing samples, which you will classify from the Bayesian classifier you have just created from the training data.

### **Example1.testing.txt**

```
1.25
0.7
```

The naming convention will be standardized as *name.training.txt* and *name.testing.txt*. At the command line you will only have to type in:

```
> Proj5 name
```

The training set can contain any number of samples and classes. So be sure to keep your code flexible. In all likelihood, input files will be much larger than this one and contain many more samples.

**Output:** You will output a file, which contains the classification of each item in the testing set. For instance:

### **Example1.decision.txt**

Sharp\_unsafe  
Smooth\_safe

Here we decide that the first item with a pointyness of 1.25 is a sharp and unsafe and the second item with a pointyness of 0.7 is a smooth safe piece. Notice that we make a single decision for each item in the testing file.

**Primer:** There is a tutorial online to help you with this problem.

## Questions 2 (20%)

(5 points each)

- (A) Suppose that your sampling was not random and that even though class A and class B have the same prior probability of occurring, you sample class A far more often. How does this affect your classifier?
- (B) Suppose that it's not good enough to simply classify objects, but you also need to give some confidence in your classifiers decisions, how might you do this? Give some examples.
- (C) What do you suppose would happen if samples were not normally distributed? As an example, you measure the temperature of some class A and most of the samples in A are either very hot or very cold, but few are warm (in general this is known as a "U" shaped distribution). Give an idea of what would happen if you proceeded to make inferences as if the data was normally distributed.
- (D) Thinking about what it means to draw samples, suppose that each time you drew a sample from some class, the probability of drawing a sample from that class increased. For instance, you are trying to get samples from different types of nails passing on a conveyer, which are mixed in with other metal parts. Class A is comprised of rusty short nails. After you see your first rusty short nail you have a better idea of how they look and it becomes easier to spot them compared with other types of nails. How might this affect the effectiveness of your sampling? How might you compensate for this?
- (E) Give yourself a pat on the back, you're DONE!