Review for Exam 1 (2/22)

Link to description of Exam 1 Format.
Link to all course notes. (See the `.ipynb` files for web versions and `.pdf` files for printable versions.)
Link to sample decision tree and coding questions for Exam 1.
Link to solutions of sample coding questions above. (Please do not read before you spent at least 30 minutes on each problem.)
Link to all homeworks and solutions.
Link to reference document for all Python constructs potentially covered.

Exam Logisitcs

- The exam is 75 minutes, and will take place on Tue. 2/27, unless you made special arrangements to reschedule during the first 3 weeks of class. No rescheduling is possible at this point.
- At the beginning of the class session, before the exam, please help the professor to arrange tables into four rows, the first three having 7 tables and the last having 5. There should be 2 chairs per table. Please also help rearrange the room back after the exam.
- No laptop/tablet/phone use is allowed during the exam. You may bring any amount of books or notes or cheatsheets you made for yourself. You may also use a handheld calculator. (Banning of electronic devices is to prevent cheating.)
- No bathroom break during the exam. Please go to the bathroom before the exam if you need to, or go after you turn in your papers. (This is to avoid cheating.)
- No leaving the room in the last 15 minutes of the exam. If you finish early, you may leave within the first 60 minutes, or you must wait until the end. (This is to prevent distractions to students who are time pressed.)
- No talking with anyone other than the professor during the exam. Any talking will be considered cheating and you will both fail the course with no second chances.
- No talking about the exam to any student who has not taken it (for example students in the other section).

Most urgent questions

In-class activity:

- Work in groups of 2-3 and share about how you are preparing for Exam 1.
- Write down a list of questions or topics that you need help in but cannot simply figure out by yourself through additional review and practice.
- **Choose one question that you believe to be the most urgent among your group, or you think would be important to many others in the class, and input your question in the Qualtrics form here**. The professor will either answer the question in class, or type up an answer in this course notes afterward.

Answers to questions

Below are answers to the survey questions that were not addressed in class. I organized the question according to topic and rephrased for ease of reading.

General Exam Preparation

Q: How do I begin studying?
You can following the steps below.

1. Review the homework questions relevant to the exam (see Exam 1 format above), and understand the solutions.
2. Do additional practice problems. Decision tree and coding problems are given in the link at the beginning of this document. Probability problems are similar to those at the end of DMD chapters 2 and 3. **Don't jump to the solutions but try to spend significant time solving. (Trying to figure out while stuck is the most efficient way to learn for retention.)** After 30 minutes or so you can read notes and the book to learn what is missing. Only read the solutions as a last resort.
3. Read through the Python constructs reference document (link is at the top.)
4. Read through the course notes as a review. The link to all course notes is http://nbviewer.jupyter.org/github/pengshi46/usc-dso-570/tree/master/Course%20Notes/?flush_cache=true/. The `.ipynb` files are for web viewing and the `.pdf` files are for printing.

Q: Can you give a summary of the Python packages/commands?
The four packages we will be using are:

- `numpy`: for numpy arrays, which allows for efficient calculation of lists or tables of numbers. (If we had used `list`, then we cannot for example add 3 to every number at once, but have to go through the list one at a time and modify. This efficient operation at once is called **vectorized** operations.)
- `scipy.stats`: this is for defining distributions and generating random samples. The command `rvs` generates the random samples.
- `matplotlib.pyplot`: this is used for plotting.
- `pandas`: this is used to work with data in table format with row/column labels. We will not be focusing on this package in this class.

We will also be using certain Python commands that are in the basic language, without the need to import anything. Read the "Exam 1 Format" for list of commands. **See the Python construct reference document linked at the top of this page for explanations of each.**

Q: Can you give more practice problems for decision tree and coding?
See the link at the top of the page for practice problems similar to those in the exam. Posted there are 2 decision tree problems and 3 coding problems that are new (have not appeared in previous homeworks). Review also the decision tree problems in Homework 2 and 3, and the coding problems in course notes to "08-Simulation Modeling", as well as in Lab 2 and Homework 5.

Decision Tree

The steps to take on a decision tree question is:

1. Read the prompt carefully and note all pieces of data (you can underline on the exam).
2. Write down decisions and events in chronological order (to help you decide which node comes after which when drawing the tree.)
3. Draw the shape of the tree (on paper in the exam).
4. Write down the value of each outcome node (the end notes of the tree).
5. Write down the probabilities at each event node.

6. Solve the tree by backward induction. For event nodes, take the expected value of the immediate children. For decision nodes, take the value of the best children.

Q: I am concerned about how much time it takes to do a decision tree.
The decision tree in the exam will be on the less complex side. See the sample problems at the top of this page for examples. I will try to make the text to the point and not give extraneous useless information to help with reading. You should also practice creating and solving decision trees to be faster.
Q: How to know if a decision tree is complete?
If you follow the steps above, then the key is to do step 1 thoroughly. If you didn't follow the above steps and jumped ahead to draw the tree without reading all information carefully, your mind may already be locked into an incorrect understanding of the problem context that it will be very difficult to correct it. Hence, follow the steps above and read everything carefully before constructing the tree.

Probability

Q: How do we know when we need to use conditional probability?
Whenever you see a statement or wants to make a statement "the probability is ... when/if ...", then you need to use conditional probability. In other words, the probability is conditional or contingent on some other information. In a deicision tree, when computing the probabilities at an event node, you need to condition on all relevant information that has happened before this event node.
Review the course notes to Decision Trees (1/16) as well as the lecture video for further explanation of conditional probability.
Q: Explanation of CDF, PMF, and PDF.
See course notes to "02 - Probability Distribution".
Q: What is the relationship between random variables and probability distributions?
There is a close relationship between the two. A probability distribution describes what probability each possible outcome a random variable can take. However, there may be multiple random variables of the same distribution. For example, if

$$X = \begin{cases} 0 & \text{with probability .6} \\ 1 & \text{with probability .4} \end{cases}$$

$$Y = \begin{cases} 0 & \text{with probability .6} \\ 1 & \text{with probability .4} \end{cases}$$

Then we have two random variables, but they both have the same distribution (Bernoulli with $p = 0.6$). Every random variable is associated with one distribution, but a distribution can describe many random variables.

Simulation

Q: How to use the `rvs` function to generate scenario?
The `rvs` function creates random samples from a given distribution. In simulation, we need random samples, so we always use the rvs function. For example, to create one Bernoulli sample (with $p = 0.6$), we do

```
bernoulli(p=0.6).rvs()
```

If you don't do the `.rvs()` then the result is not a sample (not 0 or 1), but a Python object. If you do the `.rvs()`, then the result is either 0 or 1.
To draw 10 samples at once, you can put `size=10` as an argument to the `rvs` function, as in

```
bernoulli(p=0.6).rvs(size=10)
```

When generating scenario, we need to generate random samples from distributions, and perhaps combine them in certain ways (adding, stacking, multiplying, etc). These operations are easy to do because the `rvs` function returns numpy arrays, for which we can do vectorized operation (i.e. with one command, add 1 to everything, or add every element of one array to every element of another). You have to know

- How many samples you are generating (perhaps this also depends on another random sample).
- How to combine them.

For the first part, you simply have to read the problem carefully. For the second, you can review numpy array operations (see the reference document at the top).

Q: How to start thinking about writing a simulation function?

Either when you are generating scenario, or when analyzing a given scenario, the steps are:

1. Describe how you would do the task manually by hand in English.
2. Turn your description into a more precise, step by step recipe, where you specify the input and output of each step. Make each step as small as possible (so each step hopefully is at most one line of code).
3. Translate your recipe in step 2 into code.

If you follow the above sequence, then it is much easier to get the overall logic right (describing first what to do in English is much easier than in code). Once you have the logic, it's a mechanical process to translate into code. However, if you jump into writing code, your brain may become fixated into a particular syntax or the wrong logic, and you will spend much more time on the question and probably not get it correctly. The same sequence of step can be used to write any code.

Q: How to generate a mixture of 3 or more distributions?

In the sample problem in class, you generate a mixture of two distributions. If the event is popular, then generate Poisson samples of size 8 and mean 100 each, otherwise, generate with mean 45. (The reason we generate all 8 weeks using one mean is that the event is either popular or not. If it is popular, it will have high demand for all 8 weeks. This is an application of conditional probability.) The sample solution (see above) uses a bernoulli random variable to decide between these. More generally, if you want to generate a mixture of 3 or more distributions, you can follow the below recipe. Suppose you want demand to be `n` samples from distribution `dist1` with probability `0.4`, to be `n` samples of `dist2` with probability `0.5`, and to follow `dist3` with remaining probability.

```
x=uniform(loc=0,scale=1).rvs()
if x<=0.4:
    demand=dist1.rvs(size=n)
elif x<=0.9:
    demand=dist2.rvs(size=n)
else:
    demand=dist3.rvs(size=n)
```

Here, x is uniformly random between 0 and 1. If $x \in [0, 0.4]$ then the first block is executed. If $x \in (0.4, 0.9]$ then the second is executed. If $x \in (0.9, 1]$ then the third is executed.

Homeworks and Course Notes

Q: What is the logic in homework 5, question 6 (about simulating queue)?
See the solution to homework 5, question 6. There I wrote a paragraph describing the logic of the code and why we are taking each step.

Q: Can you explain homework 4, question 4? Can you also explain the third graph in course notes to "08-simulation modeling"?
Both of these are trying to illustrate the central limit theorem, which is that the sum of many independent random variables is approximately normally distributed (see more precise statement of theorem in the above course notes). The homework question is illustrating the examples in the book. These examples involve the sum of identically distributed random variables. In the course notes, we want to give a more general illustration by adding two different types of distributions together, and show that the result is still normally distributed, with mean and standard deviation given by the formula in the central limit theorem.

Mechanically speaking, the codes to these are very similar. The steps are:

1. Generate samples of each of the individual random variables and adding the samples. In the course notes, this is done through the line

   ```
   total=np.sum(type1Demand)+np.sum(type2Demand)
   ```

   In the homework solution, this is done through

   ```
   samples=sum(X.rvs(size=100000) for i in range(n)) # Compute a sample of the sum of n ind
   ```

Here `n` is the number of random variables to add, and we are simply generating n size-100000 samples and adding these arrays of samples. We are using 100000 samples so that the histogram accurately reflect the underlying distribution. (With few samples, the histogram may not be representative of the underlying probabilities.)

2. Calculate what is the mean and standard deviation of the normal approximation (using the CLT formula).

3. Plotting both the histogram and the normal plot on the same graph.

Q: What is the code for optimal promotion pricing in the simulation modeling session?
See the code in the course notes for "8-simulation modeling." The link to all course notes is given at the top.

The logic here is that in `generateScenario`, we first sample how many customers. Then we generate an array of valuations with the length of the array equal to the number of customers sampled. See the description of each distribution we are using to generate at the beginning of the code. In `simulateScenario`, we are counting the number of customers whose valuation is at least the price. (See the numpy section of the Python constructs linked at top of this page for explanation of np.sum and >= when applied to numpy arrays.) We then calculate the revenue by noting that the number of tickets sold is the minimum of how many people are willing to pay and how many tickets we have.