Homework 4: Distributions

This assignment includes all problems in Wk.11.2x in the online tutor. For any of the sections of the assignment that require you to write code, please do your work in the file hw4Work.py.

You can discuss this problem, at a high level, with other students, but everything you turn in must be your own work.

1 Introduction

In this homework, you will implement some types of distributions that will come in handy when we look at robot localization and mapping.

Read section 7.5 of the course readings on modeling with distributions.

See the last section for information on debugging.

2 Part 1: Basic distributions

In the rest of the problems, we will build a simple parameterized set of discrete distributions on integers, and a method for combining them.

Step 1. Define a procedure squareDist:

squareDist(lo, hi, loLimit = None, hiLimit = None)

that constructs and returns an instance of dist.DDist that assigns the same probability value p to each integer between lo and hi-1, such that the distribution sums to 1. If loLimit is defined, then do not assign any probability to values lower than loLimit; instead, assign any probability that should go to a lower value to loLimit itself. So, for example, if lo and hi are both less than loLimit, then loLimit should be assigned probability 1. Treat hiLimit similarly. There is an example shown in the notes.

Look at the documentation for the function util.clip. This should make handling loLimit and hiLimit much easier.

Here are some examples:

```python
>>> squareDist(2, 4)
DDist(2: 0.500000, 3: 0.500000)
>>> squareDist(2, 5)
DDist(2: 0.333333, 3: 0.333333, 4: 0.333333)
>>> squareDist(2, 5, 0, 10)
DDist(2: 0.333333, 3: 0.333333, 4: 0.333333)
>>> squareDist(2, 5, 4, 10)
DDist(4: 1.000000)
>>> squareDist(2, 5, 3, 10)
```
Step 2.

Wk.11.2.1 Define a procedure squareDist.

Step 3. Define a procedure triangleDist:

\[ \text{triangleDist(peak, halfWidth, loLimit = None, hiLimit = None)} \]

where peak and halfWidth are integers. It should construct and return an instance of dist.DDist, which has maximum probability value at peak and has linearly decreasing values at each of halfWidth–1 points on either side of the peak. It should be clipped at loLimit and hiLimit in a way similar to squareDist.

\[
\text{>> triangleDist(5, 1)}
\]

DDist(5: 1)

\[
\text{>> triangleDist(5, 2)}
\]

DDist(4: 0.250000, 5: 0.500000, 6: 0.250000)

\[
\text{>> triangleDist(5, 3)}
\]

DDist(3: 0.111111, 4: 0.222222, 5: 0.333333, 6: 0.222222, 7: 0.111111)

\[
\text{>> triangleDist(5, 3, 0, 10)}
\]

DDist(3: 0.111111, 4: 0.222222, 5: 0.333333, 6: 0.222222, 7: 0.111111)

\[
\text{>> triangleDist(5, 3, 3, 10)}
\]

DDist(3: 0.111111, 4: 0.222222, 5: 0.333333, 6: 0.222222, 7: 0.111111)

\[
\text{>> triangleDist(5, 3, 3, 10)}
\]

DDist(4: 0.333333, 5: 0.333333, 6: 0.222222, 7: 0.111111)

\[
\text{>> triangleDist(5, 3, 4, 10)}
\]

DDist(5: 0.666667, 6: 0.222222, 7: 0.111111)

\[
\text{>> triangleDist(5, 3, 6, 10)}
\]

DDist(6: 0.888889, 7: 0.111111)

Step 4.

Wk.11.2.2 Define a procedure triangleDist.

3 Part 2: Mixtures of distributions

We can make a new distribution by defining it to be a mixture of two existing distributions. By specifying two distributions, d1 and d2, and a mixture probability p, the mixture distribution assigns to each element x, p times the probability of x in distribution d1 plus (1 – p) times the probability of x in distribution d2.

Implement a new class, MixtureDist, whose __init__ method takes the two distributions and the mixture probability. It needs to provide a method prob(self, x), which returns the probability assigned to x in the mixture distribution, and a method support(self), which returns a list of the elements that have non-zero probability in the mixture distribution. Don’t modify the __str__ method we have given you.
There are plots of several examples of mixtures in the readings. Here are a couple of very simple examples:

```python
>>> MixtureDist(squareDist(2, 4), squareDist(10, 12), 0.5)
MixtureDist({11 : 0.25, 2 : 0.25, 3 : 0.25, 10 : 0.25})
>>> MixtureDist(squareDist(2, 4), squareDist(10, 12), 0.9)
MixtureDist({11 : 0.05, 2 : 0.45, 3 : 0.45, 10 : 0.05})
>>> MixtureDist(squareDist(2, 6), squareDist(4, 8), 0.5)
MixtureDist({2 : 0.125, 3 : 0.125, 4 : 0.25, 5 : 0.25, 6 : 0.125, 7 : 0.125})
```

Step 5.

**Wk.11.2.1** Implement the class `MixtureDist`.

4 **Plots for debugging**

At the bottom of `hw4Work.py` there are some instructions and commented-out definitions that will allow you to plot the distributions you are constructing. This can be very helpful, especially for triangles and mixtures. You will need to run this using `idle -n`. 