Makeup

6.01 Midterm 2: Fall 2010

Name: Michael Plasmeier	**Section:** 1

Enter all answers in the boxes provided.

During the exam you may:

- read any paper that you want to
- use a calculator

You may not

- use a computer, phone or music player

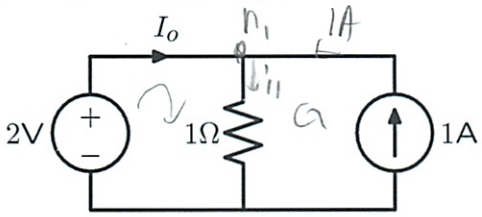
For staff use:

1.	14 /14
2.	14 /16
3.	13 /13
4.	16 /16
5.	9 /14
6.	13 /15
7.	1 /12
total:	80/100

14/14

1 Analyzing Circuits (14 points)

Determine the indicated parameter for each of the following circuits. Write your answers (either a number or numerical expression) in the box provided. For partial credit, you may also provide a brief explanation of your reasoning.



$I_o =$ 1A

Reasoning:

$V = iR$

$2 = i_0 R$

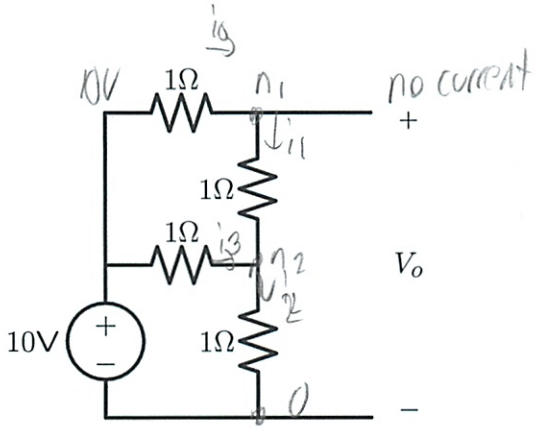
$I_o + 1A = i_1$

$\frac{2V - 0}{1} = i_1$

$\frac{2V}{1} = I_o + 1A$

$2V = I_o + 1$

$I_o = 1A$



$V_o =$ 8V

Reasoning:

$\frac{10V - n_1}{1} = \frac{n_1 - n_2}{1} = i_1 = i_0$

$V_o = ?$

need n_1

$\frac{n_1 - n_2}{1} + \frac{10 - n_2}{1} = \frac{n_2 - 0}{1}$

$i_1 + i_3 = i_2$

$10V - n_1 = n_1 - n_2$

$10V + n_2 = 2n_1$

$n_1 = 5 + \frac{n_2}{2}$

$n_1 - n_2 + 10 - n_2 = n_2$

$n_1 + 10 = 3n_2$

$n_2 = \frac{n_1 + 10}{3}$

$n_1 = 5 + \frac{\left(\frac{n_1 + 10}{3}\right)}{2}$

$2n_1 = 10 + \frac{n_1 + 10}{3}$

$6n_1 = 30 + n_1 + 10$

$5n_1 = 40$

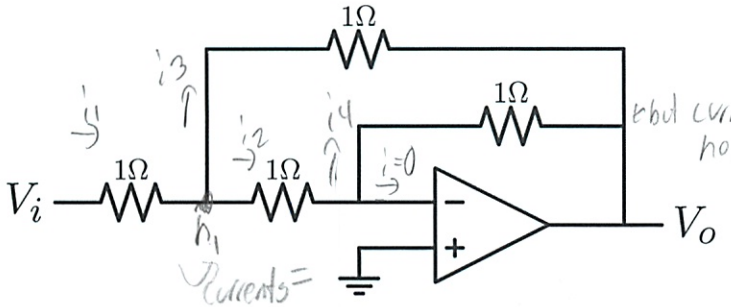
$n_1 = 8$

makes sense

think flowpl out

2 Op Amps (16 points)

Determine the indicated parameter for each of the following circuits. You may assume the ideal op-amp model. Write your answers (either a number or numerical expression) in the box provided. For partial credit, you may also provide a brief explanation of your reasoning.



total current = i_1
no-current divider

$$\frac{V_o}{V_i} = -\frac{1}{4}$$

✓ based off inverting design

Currents =

Solve for V_o

8/8

Reasoning: $i_1 = i_3 + i_2$

$\frac{V_i - n_1}{1} = \frac{n_1 - V_o}{1} + \frac{n_1 - V_-}{1}$

$V_i - n_1 = n_1 - V_o + n_1 - V_-$

$V_i - n_1 = n_1 - V_o + n_1 - 0$

$V_i - n_1 = n_1 - V_o + n_1$

$V_i - n_1 = 2n_1 - V_o$

$V_i - 3n = -V_o$

$V_o = -V_i + 3n$

$V_+ = V_- = 0$

$i_2 = i_4$

$\frac{n_1 - V_-}{1} = \frac{V_- - V_o}{1}$

$n_1 - V_- = V_- - V_o$

$n_1 - 0 = 0 - V_o$

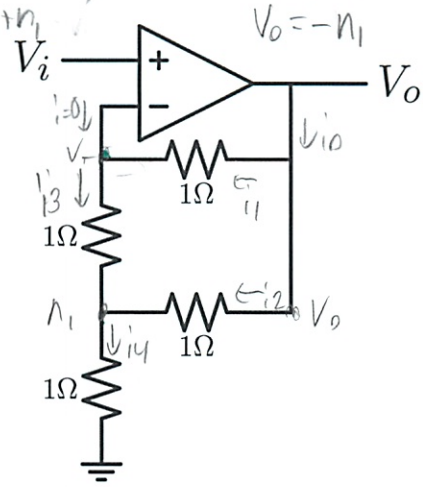
$V_o = -n_1$

$V_o = -V_i - 3V_o$

$\frac{4V_o}{4} = \frac{-V_i}{V_i}$

$\frac{V_o}{V_i} = -\frac{1}{4}$

6/8



$$\frac{V_o}{V_i} = \frac{5}{7}$$

Reasoning: $V_i = V_+ = V_-$ ✓

$i_1 = i_3$

✓ $\frac{V_o - V_-}{1} = \frac{V_- - n_1}{1}$

$i_2 = \frac{V_o - n_1}{1}$

$i_4 = \frac{n_1 - 0}{1}$

~~$i_o = i_1 + i_2$~~

$V_o - V_- = V_- - n_1$

$V_o - V_i = V_i - n_1$

$V_o + n_1 = 2V_i$

$V_i = \frac{V_o + n_1}{2}$

?

$i_3 + i_2 = i_4$

$V_- - n_1 + V_o - n_1 = n_1$

$V_i - n_1 + V_o - n_1 = n_1$ ✓

$V_i + V_o = 3n_1$

$n_1 = \frac{V_i + V_o}{3}$

$V_i = V_o + \frac{\left(\frac{V_i + V_o}{3}\right)}{2}$

$2V_i = 2V_o + \left(\frac{V_i + V_o}{3}\right)$

$6V_i = 6V_o + V_i + V_o$

$5V_i = 7V_o$

$\frac{V_o}{V_i} = \frac{5}{7}$

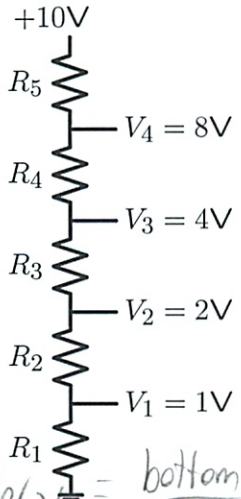
don't remember all this

13/13

3 Designing Circuits (13 points)

Determine the indicated parameters for each of the following circuits. Write your answers (either a number or numerical expression) in the box provided. For partial credit, you may also provide a brief explanation of your reasoning.

Part a. Determine values of R_1, R_2, R_3, R_4, and R_5 so that $V_1 = 1V$, $V_2 = 2V$, $V_3 = 4V$, and $V_4 = 8V$. The answer to this part is not unique. You will get full credit for any valid solution.



$R_5 = 2$

$R_4 = 4$

$R_3 = 2$

$R_2 = 1$

$R_1 = 1$

+

+ 10 ✓

5

Massive Voltage Divider $V_{end} = \frac{\text{bottom } \Omega}{\text{whole } \Omega} \cdot \text{start } V$

Reasoning:

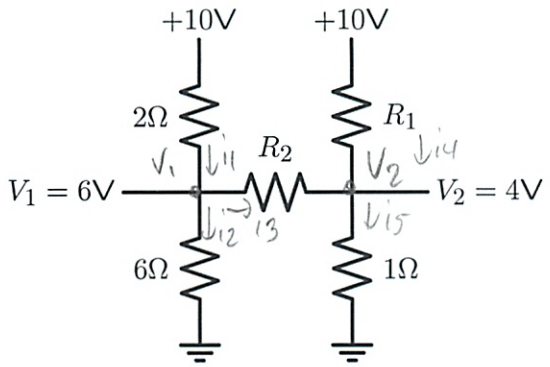
$8 = V_4 = 10 \cdot \frac{R_4+R_3+R_2+R_1}{R_5+R_4+R_3+R_2+R_1}$ $\quad 8 = 10 \cdot \frac{R_4+2+1+1}{10}$ $\quad R_4 = 4$, $R_5 = 2$

$4 = V_3 = 10 \cdot \frac{R_3+R_2+R_1}{R_5+R_4+R_3+R_2+R_1}$ $\quad 4 = \frac{R_3+1+1}{10} \cdot 10$ $\quad R_3 = 2$, $R_4+R_5 = 6$

$2 = V_2 = 10 \cdot \frac{R_2+R_1}{R_5+R_4+R_3+R_2+R_1}$ ← next here $\quad 2 = \frac{R_2+1}{10} \cdot 10$ $\quad R_2 = 1$, $R_3+R_4+R_5 = 8$

$1 = V_1 = 10 \cdot \frac{R_1}{R_5+R_4+R_3+R_2+R_1}$ ← start here $\quad \frac{R_1}{R_5+R_4+R_3+R_2+R_1} = \frac{1}{10}$ $\quad R_1 = 1$, $R_2+R_3+R_4+R_5 = 9$

Part b. Determine values of R_1 and R_2 so that $V_1 = 6V$ and $V_2 = 4V$.



$R_1 = 2$

$R_2 = 2$

8

√realistic answer

Reasoning:

$i_1 = i_2 + i_3$

$$\frac{10-6}{2} = \frac{6-0}{6} + \frac{6-4}{R_2}$$

$$2 = 1 + \frac{2}{R_2}$$

$$1 = \frac{2}{R_2}$$

$$R_2 = 2$$

$i_4 = i_5 - i_3$

$$\frac{10-4}{R_1} = \frac{4-0}{1} - \frac{6-4}{R_2}$$

$$\frac{6}{R_1} = 4 - \frac{2}{R_2}$$

$$\frac{6}{R_1} = 4 - \frac{2}{2}$$

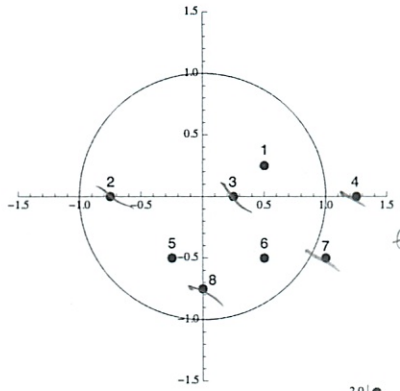
$$\frac{6}{R_1} = 3$$

$$6 = 3R_1$$

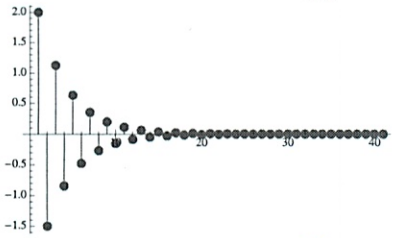
$$R_1 = 2$$

4 Pole Position (16 points)

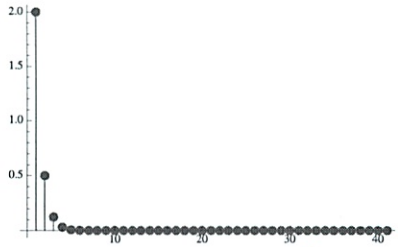
Consider eight poles located at the following locations in the z plane. The plots below show the unit-sample responses of eight linear, time-invariant systems. Match them with the dominant pole for each system (remember thta the system may have more than just one pole).



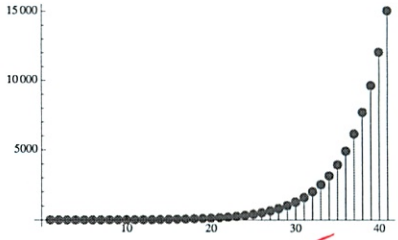
7 - ossilatory divergent
4 = diverge monotonically
3 = converges monotonically
← thought we just care about 1st quadl
2 = every other value neg - no 0 stop



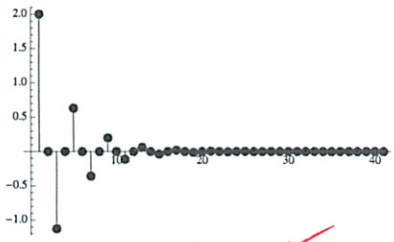
A = 2



B = 3

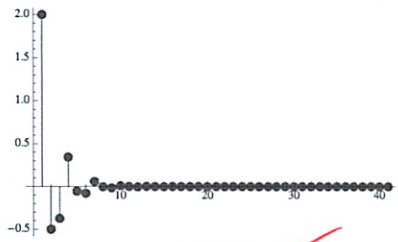


C = 4

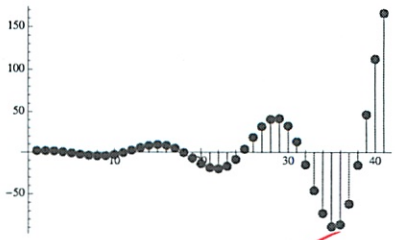


D = 8

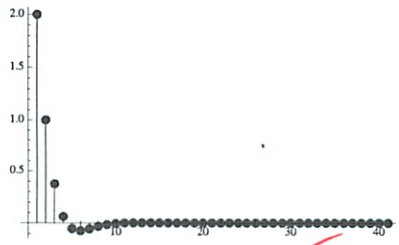
E = 6



F = 5



G = 7



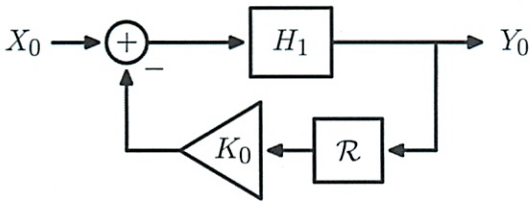
H = 1

?3?1

8 = pluy imaginary
osilates perfectly
1, 5, 6 - ossilate converges
higher imag → faster ossilation
? diff 1 + 6 ?

5 System functions (14 points)

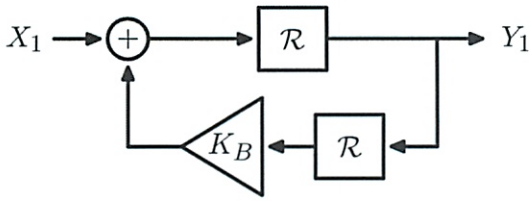
Let H_1 represent a subsystem that is part of the larger system shown below.



The system function for the larger system can be written as

$$H_0 = \frac{Y_0}{X_0} = \frac{H_1}{1 + K_0 R H_1}.$$

Assume that $H_1 = H_{1B} = \frac{Y_1}{X_1}$ as shown below.



$Y_1[n] = x[n-1] + y_1[n-2]k_B$

$Y_1 = XR + k_B Y_1 R^2$

$Y_1(1 - k_B R^2) = X_1 R$

What is the system function for H_{1B}?

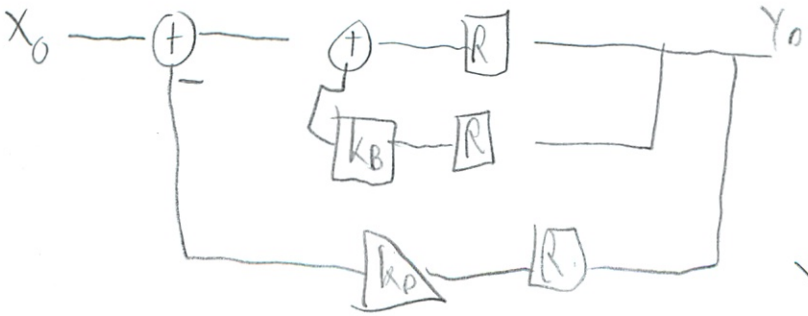
$H_{1B} = \frac{Y_1}{X_1} = \frac{R}{1 - k_B R^2}$

Determine the system function H_0 for the larger system when $H_1 = H_{1B}$.

$H_0 = \frac{Y_0}{X_0} = \frac{X}{1 - (k_B - k_0)R^2}$

Oh fits in

$Y[n] = x[n-1] + k_B Y[n-2] - k_0 Y[n-2]$



$Y = XR + k_B Y R^2 - k_0 Y R^2$

$Y = XR + (k_B - k_0) Y R_2$

$Y(1 - (k_B - k_0)R_2) = XR$

Under what conditions on K_0 and K_B is this system stable? Explain.

The system will be stable if the magnitude of the dominate pole is < 1

what is actual condition on k_0, k_B

−2

Under what conditions on K_0 and K_B does the unit-sample response decay monotonically? Explain.

When the ~~magnitude of the~~ dominant pole is real and positive and magnitude < 1

−3

$$\text{Denom} = 1 - (k_B - k_0)R^2$$

$$z^2 - (k_B - k_0) = 0$$

So in both cases need to find the value of $k = -(k_B - k_0)$ such that the magnitude when you take the root matches the condition

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$\frac{\pm\sqrt{-4(1)(-k_B + k_0)}}{2 \cdot 1} = \begin{array}{l} |\text{mag}| < 1 \text{ for stable} \\ |\text{mag}| \text{ real and positive for decay monotically} \end{array}$$

6 State machines (15 points)

You are going to design a state machine that takes a string of characters as input, and outputs at each time step the largest number of times a character is repeated in a row in the sequence as observed so far. The state variable will have the following form:

```
[lastCharacter, timesSeenConsecutively, best]
```

"; where is best shown"

that is, the state will be a list of the previous input character (except for the first time step, see below), the number of times that character has occurred consecutively up to this point, and the length of the longest sequence of repeated characters observed so far.

Complete the following definition:

```
Class SubSeq(sm.SM):
    startState = None
    def getNextValues(self, state, inp):
        if state == None:
            #part 1
        elif state[0] == inp:
            #part 2
        else:
            #part 3
```

first state return([inp, 1, 1], 1)

if matches

~~return[inp, times seen~~

timesSeenConsecutively = tSC + 1

if tSC > best:

best = tSC

return ([inp, tSC best], best)

so that it has the following example behavior:

```
>>> test = SubSeq()
>>> test.transduce('abaabcdccccc')
[1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 4, 5]
```

Pth always outputes best

return ([inp, 0, ~~best~~ state[2]], state[2])

answer next page →

Supply the code for part 1:

```
return([inp, 1, 1], 1)
```

3/3

Supply the code for part 2:

```
timesSeenConsecutively = state[1]
best = state[2]
timesSeenConsecutively = timesSeenConsecutively +1
if timesSeenConsecutively > best:
    best = timesSeenConsecutively
return([inp, timesSeenConsecutively, best], best)
```

9/9

Supply the code for part 3:

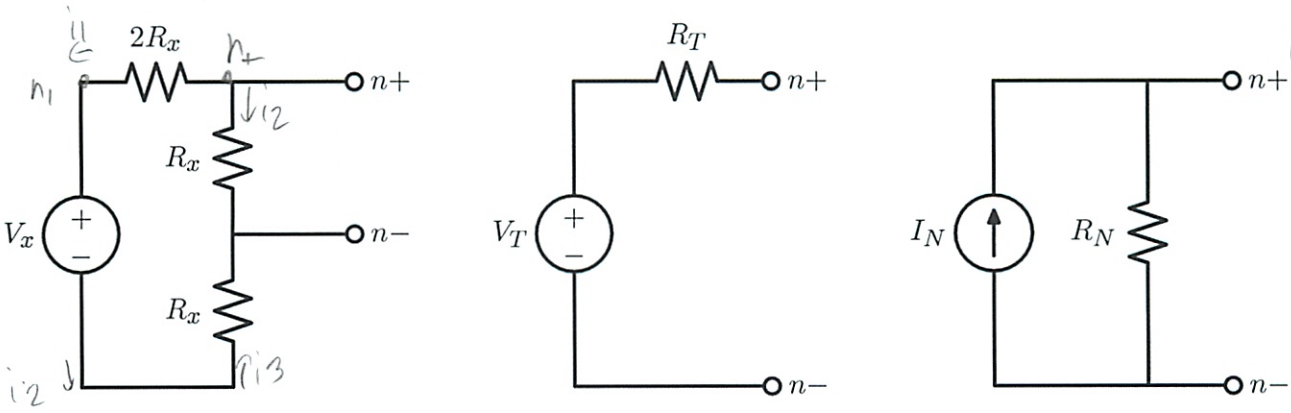
```
best = state[2]
return([inp, ~~0~~ 1, best], best)
```

1/3

7 Equivalent Circuits (12 points)

Find the Thevenin and Norton equivalent circuits for the circuit shown in the lower left panel, looking into the port labeled n+ and n−. Determine values for V_T, R_T, I_N and R_N in terms of V_x and R_x. For partial credit, you may also provide a brief explanation of your reasoning.

heavy practiced
- guess just do the normal
also abstract



no norton example!!!

Original Thevenin Norton

$V_T = \cancel{5/2}$ $I_N = \cancel{2/5}$

$R_T = \cancel{2.5R_x}$ $R_N = \cancel{2.5R_x}$

+1

Reasoning:

first open circuit voltage

$\frac{10-n_+}{3R_x} = \frac{n_+-0}{R_x}$

$(10-n_+)R_x = 3R_x(n_+)$

$10R_x - n_+R_x = 3R_xn_+$

$10R_x = 4R_xn_+$

$n_+ = \frac{10R_x}{4R_x} = \frac{5}{2}$

Close circuit (bypass R_x)

$\frac{10-n_+}{3R_x} = -i$

$10-n_+ = 3R_xi$

$\frac{\frac{5}{2}-0}{2.5R_x} = i_n$

$\cancel{i_n = R_x}$

$10 - \frac{5}{2} = 3R_xi$

$7.5 = 3R_xi$

$i = 2.5R_x$

$i_n = I_n + \frac{5/2}{R_n}$??

$V = (I + I_0)R$

$I = -I_0$

$V = 1/R$

$R_N = R_T$

$I_N = \frac{1}{V_T}$

No clue on norton

6.01 Midterm 2: Fall 2010

Name:	**Section:**

Solutions: Not correct for the make-up exam.

Enter all answers in the boxes provided.

During the exam you may:

- read any paper that you want to
- use a calculator

You may not

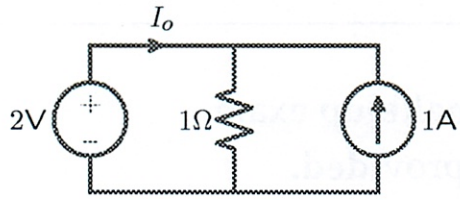
- use a computer, phone or music player

For staff use:

1.	/14
2.	/16
3.	/13
4.	/16
5.	/14
6.	/15
7.	/12
total:	/100

1 Analyzing Circuits (14 points)

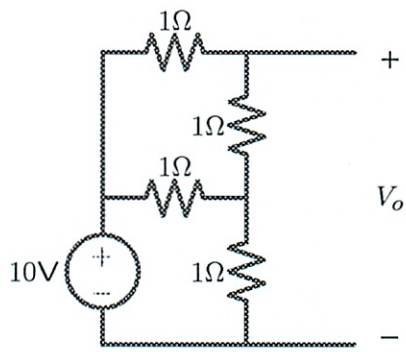
Determine the indicated parameter for each of the following circuits. Write your answers (either a number or numerical expression) in the box provided. For partial credit, you may also provide a brief explanation of your reasoning.



$I_0 = 1A$

Reasoning:

The voltage across the resistor is 2V, making the current through the resistor 2A. Applying KCL at the top node leads to $I_o + 1A = 2A$. So $I_o = 1A$.



$V_0 = 8V$

Reasoning:

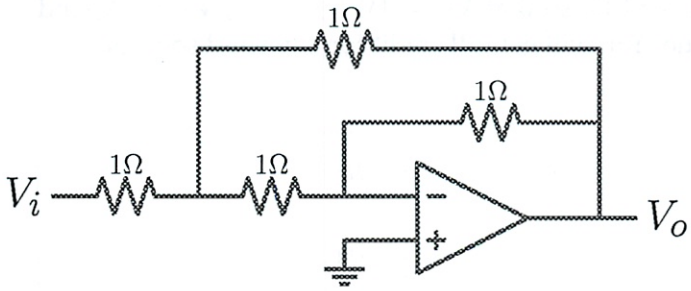
Label the node at the center-right as V_1, then:

- KCL at top-right node: $(V_o - 10) + (V_o - V_1) = 0$.
- KCL at center-right node: $(V_1 - V_0) + (V_1 - 10) + V_1 = 0$.

Solve to find that $V_0 = 8V$ and $V_1 = 6V$.

2 Op Amps (16 points)

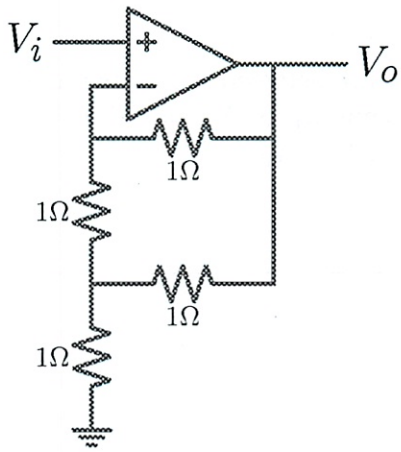
Determine the indicated parameter for each of the following circuits. You may assume the ideal op-amp model. Write your answers (either a number or numerical expression) in the box provided. For partial credit, you may also provide a brief explanation of your reasoning.



$$\frac{V_o}{V_i} = -\frac{1}{4}$$

Reasoning:

The ideal op-amp model determines $V_- = 0$, because $V_+ = 0$ and there is no voltage difference between these nodes. Then the node between V_i and V_- is $V_x = -V_o$, by applying KCL at the negative input to the op amp. Applying KCL at that node yields $(V_x - V_i) + (V_x - V_o) + V_x = 0$, which shows that $V_i = -4V_o$.



$$\frac{V_o}{V_i} = \frac{5}{4}$$

Reasoning:

Label the node between V_- and ground as V_x.

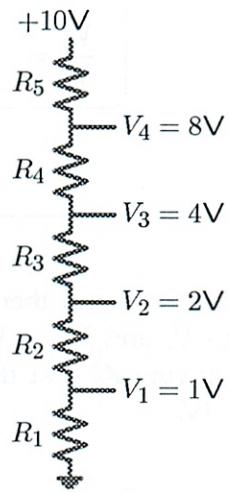
- KCL at V_-: $(V_- - V_o) + (V_- - V_x) = 0$
- KCL at V_x: $(V_x - V_-) + (V_x - V_o) + V_x = 0$

Solving yields $V_- = \frac{4}{5}V_o$. But $V_- = V_+ = V_i$. Thus $V_o = \frac{5}{4}V_i$.

3 Designing Circuits (13 points)

Determine the indicated parameters for each of the following circuits. Write your answers (either a number or numerical expression) in the box provided. For partial credit, you may also provide a brief explanation of your reasoning.

Part a. Determine values of R_1, R_2, R_3, R_4, and R_5 so that $V_1 = 1V$, $V_2 = 2V$, $V_3 = 4V$, and $V_4 = 8V$. The answer to this part is not unique. You will get full credit for any valid solution.



$R_5 = 2R_0$

$R_4 = 4R_0$

$R_3 = 2R_0$

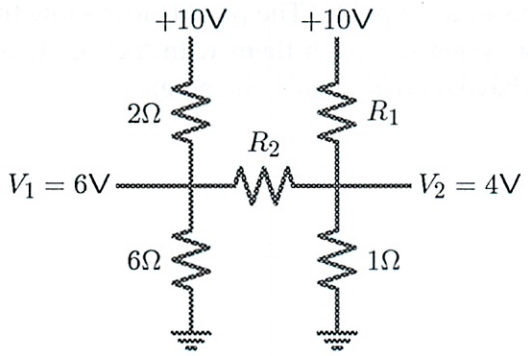
$R_2 = R_0$

$R_1 = R_0$

Reasoning:

Because the resistors are in series, the resistance between successive nodes will be proportional to the voltage between the nodes. Thus $R_1 \propto V_1 = 1V$; $R_2 \propto V_2 - V_1 = 1V$; $R_3 \propto V_3 - V_2 = 2V$; $R_4 \propto V_4 - V_3 = 4V$; $R_5 \propto V_5 - V_4 = 2V$.

Part b. Determine values of R_1 and R_2 so that $V_1 = 6V$ and $V_2 = 4V$.



$R_1 = 2\Omega$

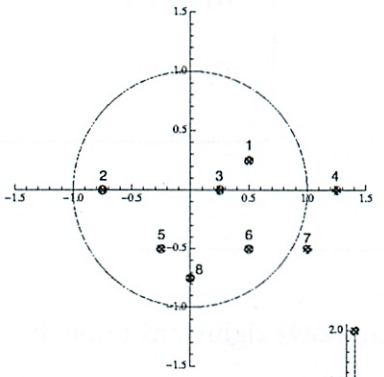
$R_2 = 2\Omega$

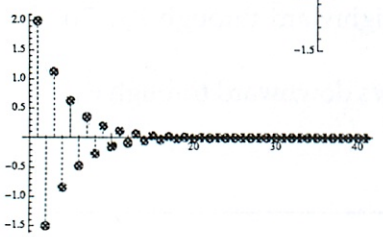
Reasoning:

KCL at the left-center node determines that a 1A current flows rightward through R_2. To make $V_1 - V_2 = 2V$, it follows that $R_2 = 2\Omega$.
KCL at the right-center node then determines that a 3A current flows downward through R_1. To make $10V - V_2 = 6V$, it follows that $R_1 = 2\Omega$.

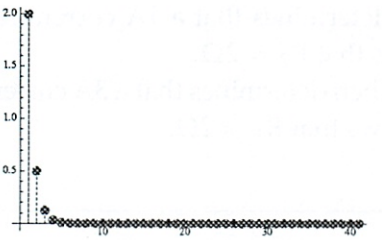
4 Pole Position (16 points)

Consider eight poles located at the following locations in the z plane. The plots below show the unit-sample responses of eight linear, time-invariant systems. Match them with the dominant pole for each system (remember thta the system may have more than just one pole).

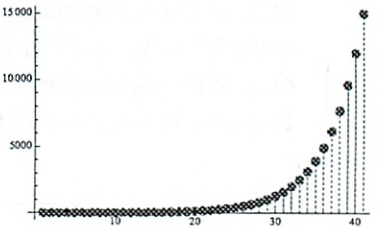




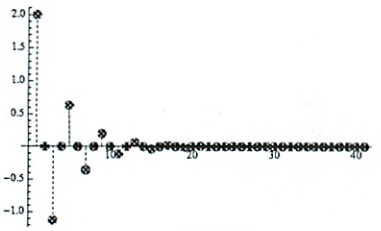
A = 2



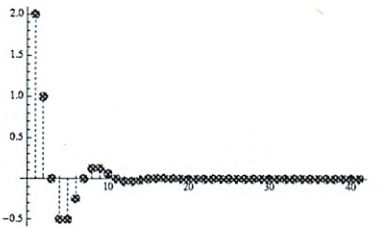
B = 3



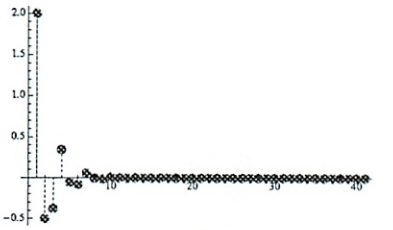
C = 4



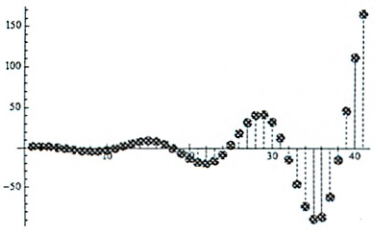
D = 8



E = 6



F = 5

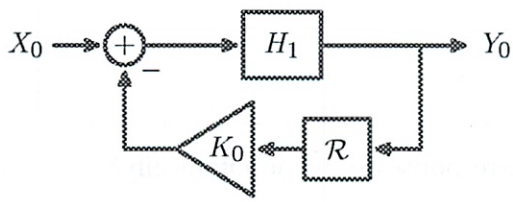


G = 7

H = 1

5 System functions (14 points)

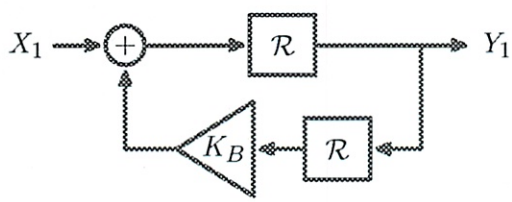
Let H_1 represent a subsystem that is part of the larger system shown below.



The system function for the larger system can be written as

$$H_0 = \frac{Y_0}{X_0} = \frac{H_1}{1 + K_0 R H_1}.$$

Assume that $H_1 = H_{1B} = \frac{Y_1}{X_1}$ as shown below.



What is the system function for H_{1B}?

$$H_{1B} = \frac{\mathcal{R}}{1 - K_B \mathcal{R}^2}$$

Determine the system function H_0 for the larger system when $H_1 = H_{1B}$.

$$H_0 = \frac{\mathcal{R}}{1 + (K_0 - K_B)\mathcal{R}^2}$$

Under what conditions on K_0 and K_B is this system stable? Explain.

There are poles at $z = \pm\sqrt{K_B - K_0}$. To be stable, the poles should all have magnitued less than 1. Thus the system is stable if $|K_B - K_0| < 1$.

Under what conditions on K_0 and K_B does the unit-sample response decay monotonically? Explain.

None. For monotonic convergence, both poles must have magnitudes between 0 and 1 (since there are two poles of equal magnitude). If $K_B < K_0$ then the poles have non-zero imaginary parts, and the response oscillates. If $K_B > K_0$ then one pole is on the positive real axis and one is on the negative real axis. The pole on the negative real axis causes the unit sample response to alternate. Thus there are no values of K_0 and K_B for which there is monotonic decay.

6 State machines (15 points)

You are going to design a state machine that takes a string of characters as input, and outputs at each time step the largest number of times a character is repeated in a row in the sequence as observed so far. The state variable will have the following form:

```
[lastCharacter, timesSeenConsecutively, best]
```

that is, the state will be a list of the previous input character (except for the first time step, see below), the number of times that character has occurred consecutively up to this point, and the length of the longest sequence of repeated characters observed so far.

Complete the following definition:

```
Class SubSeq(sm.SM):
    startState = None
    def getNextValues(self, state, inp):
       if state == None:
          #part 1
       elif state[0] == inp:
          #part 2
       else:
          #part 3
```

so that it has the following example behavior:

```
>>> test = SubSeq()
>>> test.transduce('abaabcdccccc')
[1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 4, 5]
```

Supply the code for part 1:

```
return ((inp, 1, 1), 1)
```

Supply the code for part 2:

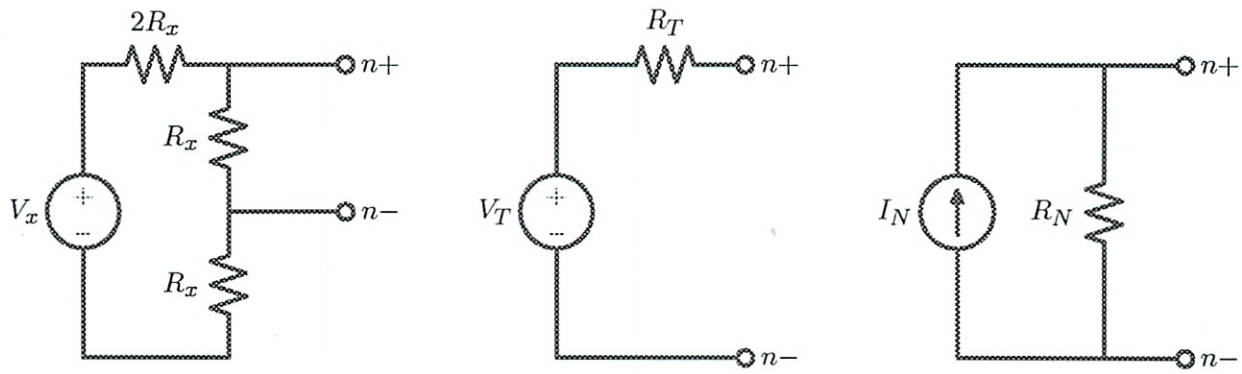
```
if state[1] == state[2]:
   return ((inp, state[1]+1, state[1]+1), state[1]+1)
else:
   return ((inp, state[1]+1, state[2]), state[2])
```

Supply the code for part 3:

```
return ((inp, 1, state[2]), state[2])
```

7 Equivalent Circuits (12 points)

Find the Thevenin and Norton equivalent circuits for the circuit shown in the lower left panel, looking into the port labeled $n+$ and $n-$. Determine values for V_T, R_T, I_N and R_N in terms of V_x and R_x. For partial credit, you may also provide a brief explanation of your reasoning.



Original Thevenin Norton

$$V_T = \frac{V_x}{4} \qquad I_N = \frac{V}{3R}$$

$$R_T = \frac{3R_x}{4} \qquad R_N = \frac{3R}{4}$$

Reasoning:

Open circuit equations

$\frac{V_x - n_+}{2R} = \frac{n_+ - n_-}{R}$ and $\frac{n_+ - n_-}{R} = \frac{n_-}{R}$

Solving these equations leads to difference between n_+ and n_- of $\frac{V_x}{4}$.

Closed circuit gives current of $\frac{-V_x}{3R}$, since only one path in circuit.

Solve linear equation between two points, find slope is $\frac{4}{3R}$, so reciprocal is resistance of $\frac{3R}{4}$

Prob similar to 6.04 - see how they introduce it

11/9

6.01: Introduction to EECS I

Discrete Probability and State estimation

Week 10 *November 9, 2010*

Reading: Chapter 8

Planning

PCAP Framework for Managing Complexity

Python:

- **procedures** are abstractions that combine primitive operations to capture common patterns
- **classes** are abstractions that associate data (attributes) and procedures (methods) that are related

Signals and Systems:

- **signals** are abstractions that collect all of the samples at different times into a single object
- **system functions** are abstractions that combine operations (delays, gains, and adders) to capture common patterns

Circuits:

- **Thevenin** and **Norton** circuits are abstractions that reduce combinations of resistors and sources to simpler structures

All of these abstractions suppress detail, and provide elements that behave in a deterministic way. w/o ~~the~~ worrying what inside

P() = 1

world is not certain

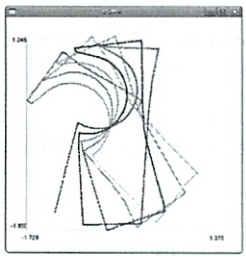
How can we look ahead when we don't know where we are?

have some knowledge
can sense things

- Driving in Boston – how do I get to Next House, when I don't know where I am ?
- Uncertainty in results of actions

know what actions you took

robot tail from odometry



- Uncertainty in results of sensing: What do we know when sonar number 2 returns a reading of 1.2?

likely something 1.15 - 1.25
multiple readings = more trust

Important ideas

PCAP:

- Combine individual pieces of noisy information into a bigger picture that improves over time.
- Combine simple probability distributions to make more complex ones (HW 4).

plant = robot
before all smarts were in controller
now bringing plant into controller

Motivation: Traditional approaches try to design robust controller, under expectation that designer can plan for all contingencies.

Modeling: Better solution: Put a model of the plant inside the controller. The **system** reasons about the outside world: provides a way for planning under uncertainty, by incorporating noisy observations.

Let's make a deal

what are expected belief about the state of the system?
- not just long term

Probability

Probability theory lets us

- Make precise statements about uncertainty – assign numerical estimates of uncertainty to events
- Combine uncertain information to reach consistent (uncertain) conclusions – do "calculus" on uncertainties to deduce information about the state of the system
- Win bets Observations help refine beliefs

Subjective view of probability as personal belief:

- Probability is not restricted to long run frequency of events;
- Probability can provide a framework for capturing beliefs about the state of the world.

Elements of probabilistic reasoning

- Need to define the set of variables that describe the world being modeled x,y, heading
- Need to define the range of values those variables can assume – **state space**
- Need to describe the probability of events occuring in state space
- Need ways of deducing information about variables, even if they cannot be directly observed

Types of State Spaces

(can't subdivide further

- Atomic: states don't have any further detailed structure.
  - A state space described by one value from a list, e.g. ('closed', 'closing', 'open', 'opening').
  - A state space described by a single integer.
  - A state space described by a single real number (infinite).
- Factored: a state is described by values of more than one (state) variable. (functionally
  - A state space described by a seconds variable in {0,...,59} and a minutes variable in {0,...,59}.
  - A state space described by X position and Y position (may be infinite).

Pcan't be easily described

~~"point"~~

State Spaces Ω

- Single coin flip (atomic): {H, T},
- Sequence of three coin flips (factored – 3 variables): {HHH, HHT, HTH, HTT, THH, THT, TTH, TTT} oh that is factored
- Randomly generated document (factored – 1000 variables): all strings of 1000 words drawn from a particular dictionary.
- Robot navigating in a city (atomic): set of intersections in the city.
- Robot navigating in a city (factored – 2 variables): GPS coordinates (longitude, latitude).

Events

An event is a subset of the sample space U (the state space) containing zero or more atomic events (atomic states).

- An event in the three-coin-flip space: there are at least two heads: $\{HHH, HHT, HTH, THH\}$
- An event in the robot navigation space: the robot is within one mile of MIT.
- An event in the document space: the document contains, in sequence, the words '6.01' and 'rules'.

Axioms of probability

A probability **distribution** Pr assigns numbers to events. It satisfies the following axioms:

state space inclusive → $\Pr(U) = 1$

empty → $\Pr(\{\}) = 0$

map things in space (events) to #

$$\Pr(E_1 \cup E_2) = \Pr(E_1) + \Pr(E_2) - \Pr(E_1 \cap E_2)$$

Sum of the probabilities assigned to the atomic events must be 1.

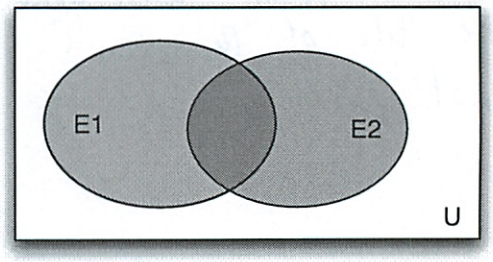
- A probability distribution will be very valuable in enabling us to reason about a system;
- However, decisions about how best to describe probability of events is critical, as they can influence the model's accuracy
  - empirical - experiments
  - theoretical model
  - physical intuition

but how to build?

This is all how to manipulate

Conditional Probability

$$\Pr(E_1 \mid E_2) = \frac{\Pr(E_1 \cap E_2)}{\Pr(E_2)} .$$



Restrict attention to E_2

Ask: In what fraction of E_2 is E_1 true?

speedy introduction!

Check Yourself

What is the conditional probability of getting a die roll greater than 3, given that it will be odd?

1. $\frac{1}{2}$
2. $\frac{1}{3}$
3. $\frac{1}{4}$
4. $\frac{1}{5}$
5. none of above

1 2 3 4 5 6 or $\frac{1/6}{1/2} = \frac{1}{3}$

two ways to do it

prob that will be odd

Random variables

Some states more easily described in factored form (e.g,. 10 coins each heads or tails rather than all 2^{10} possible sequences).

Similarly, may want to describe probability distributions over one or more variables.

A **random variable**, V, is specified by:

- a domain, D_V, which is the set of values, v_i, it can take on
- a probability distribution, $\Pr(V = v_i)$, which specifies the probability of it taking on any of its values

$$\sum_{v_i \in D_V} \Pr(V = v_i) = 1$$

entire sum is 1

A probability distribution over one state variable.

Notation

- A: capital letters stand for random variables
- a: small letters stand for possible values of random variables; a is an element of the domain of random variable A
- $A = a$: the event that random variable A has value a

Example random variables

- The number of students who will come to software lab
- The license plate number of the car in spot 12 of the Stata Garage
- The outcome of some die roll
- The amount of rain we'll have today

Joint probabability distributions

A joint probability distribution

- Defined over two or more random variables; e.g., V and W
- Written $\Pr(V, W)$, where V and W are random variables
- Domain is *cartesian product* of D_v and D_W

  $\{(v, w) \mid v \in D_V, w \in D_W\}$
- Probability is assigned to (v, w) pairs
- The sum of the probabilities must be 1

we are generally interested in **dependent** variables, not independent variables.

diff than 6.041

not $P(V)P(W)$ ← independent

Conditional Probability Distributions

if only have partial information

A conditional probability distribution

- Written $\Pr(V \mid W)$, where V and W are random variables,
- Is a function from values, w, of W, to probability distributions on V.

$$\Pr(V \mid W) = \lambda w. \Pr(V \mid W = w)$$

where λ is the Python `lambda` for creating functions.

knowing you are ~~sent~~ somewhere on W what is the prob dist for V?

Conditional Probability

Consider two random variables:

- D, which has values: *disease*, *nodisease*
- T, (a test result) which has values: *pos*, *neg*

Efficacy of test is specified by $\Pr(T \mid D)$
a conditional distribution on the test results given whether a person has the disease.

false negative

$$\Pr(T \mid D) = \begin{cases} \{pos : 0.99, neg : 0.01\} & \text{if } D = disease \\ \{pos : 0.001, neg : 0.999\} & \text{if } D = nodisease \end{cases}$$

what is prob dist

each d

false positive

Python Representation

```
class DDist:
    def __init__(self, dictionary):
        self.d = dictionary
    def prob(self, elt):
        if self.d.has_key(elt):
            return self.d[elt]
        else:
            return 0

dist.DDist({(0, 0) : 0.5, (0, 1): 0.2, (1, 0): 0.1, (1, 1): 0.2})

def TgivenD(D):
    if D == 'disease':
        return dist.DDist({'pos' : 0.99, 'neg' : 0.01})
    elif D == 'nodisease':
        return dist.DDist({'pos' : 0.001, 'neg' : 0.999})
```

then ~~combine~~ do the test multiple times

Operations on Distributions

- Constructing a joint distribution
- Marginalization
- Conditioning

- reason about 1 by summing away the others

Dental distribution

- C: *cavity*: $D_C = \{T, F\}$
- A: *toothAche*: $D_A = \{T, F\}$
- domain: $D_C \times D_A = \{(T,T), (T,F), (F,T), (F,F)\}$

$P(A=F, C=T) = P(C=T)\,P(A=F \mid C=A)$
$.15 \cdot 1-.33$

Constructing a Joint Distribution

Given

- a distribution $\Pr(V)$ and
- a conditional distribution $\Pr(W \mid V)$

Produce a **joint distribution** by computing, for every pair of values v in the domain of V and w in the domain of W, a product of the two factors.

(the thing I always forget!)

$$\Pr(V = v, W = w) = \Pr(V = v)\Pr(W = w \mid V = v)$$

similarly

$$\Pr(V = v, W = w) = \Pr(W = w)\Pr(V = v \mid W = w)$$

Check Yourself

- $\Pr(C = T) = 0.15$ prob of cavity
- $\Pr(A = T \mid C = T) = 0.3333$ have toothache given have cavity
- $\Pr(A = T \mid C = F) = 0.0588$ " " but no cavity

		C: T	C: F	
A	T	.05	.05	
	F	~~.85~~	.8	
		.1		

$P(A=t, C=t) = P(C=t)\,P(A=t \mid C=t)$
$= .15 \cdot .33 = .05$

$P(A=t, C=F) = P(C=F)\,P(A=t \mid C=t)$
$= 1-.15 \cdot .33 = .05$

Oh must do each possibility, well here

Marginalization

Given
- a joint distribution
- index of one of the random variables

Produce a **marginal distribution** over the chosen random variable by summing over all assignments to the other random variables in the joint distribution.

$$\Pr(A = a) = \sum_c \Pr(A = a, C = c)$$

$$\Pr(C = c) = \sum_a \Pr(A = a, C = c)$$

Check Yourself

		C: T	C: F	
A	T	0.05	0.05	.1
	F	0.1	0.8	.9
Sum		.15	.85	

↑given

"filling in the margins"

Fill in the marginal distributions.

NOT INDEPENDENT

(I should have a much better intuitive sense of this)

Conditioning

Given
- a joint distribution
- index of one of the random variables
- a value for that random value

Produce a **conditional distribution** over the **other** random variable(s) conditioned on event that the specified random variable has the specified value.

For example

for all possible cases

$$\Pr(C \mid A = T) = \{\frac{\Pr(A = T, C = T)}{\Pr(A = T)}, \frac{\Pr(A = T, C = F)}{Pr(A = T)}\}$$

joint one you know

This is a distribution on C.

Conditional Dentistry

		C: T	C: F	
A	T	0.05	0.05	0.1
	F	0.1	0.8	0.9
		0.15	0.85	

- What is $\Pr(C = T \mid A = T)$? $\frac{.05}{.1} = .5$
- What is $\Pr(C = F \mid A = T)$? $\frac{.05}{.1} = .5$

$P(C \mid A=T) = \begin{cases} 1/2 & \text{that } C=T \\ 1/2 & \text{that } C=F \end{cases}$

Total Probability

Given
- a distribution $\Pr(V)$ and
- a conditional distribution $\Pr(W \mid V)$

Produce a **distribution** over $\Pr(W)$

$$\Pr(W) = \text{MarginalizeOut}(\text{Joint}(\Pr(V), \Pr(W \mid V)), V)$$

+ build joint than marginalize out

That is, for each w in domain D_W

$$\Pr(W = w) = \sum_v \Pr(W = w \mid V = v) \Pr(V = v)$$

Bayesian Reasoning

- A random patient walks in to our dental practice
$$\Pr(C = T) = 0.15$$
- She says she has a toothache.
$$A = T$$
- What's the probability that she has a cavity (given $A = T$)?
$$\Pr(C = T \mid A = T)$$
- We know:
$$\Pr(A = T \mid C = T) = .333 \quad \text{and} \quad \Pr(A = T \mid C = F) = .059$$

Form the joint distribution:

$$\Pr(C, A) = \Pr(A \mid C) \Pr(C)$$

Condition:

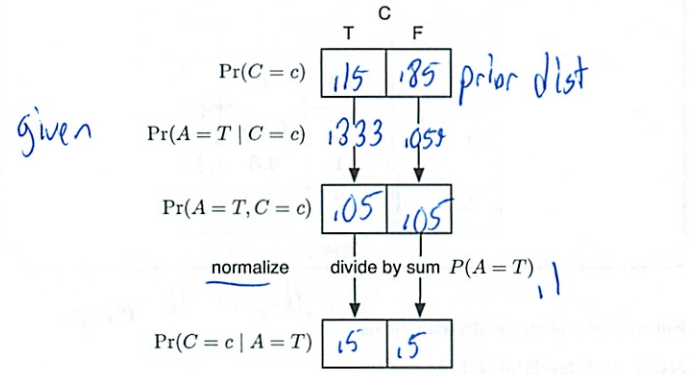
$$\Pr(C \mid A = T) = \boxed{\frac{\Pr(A \mid C) \Pr(C)}{\Pr(A = T)}}$$

Lookup:

$$\Pr(C = T \mid A = T)$$

↑without the table (does same thing)

Graphical Bayes



What about the bet?

- Total number of legos in the bag $= 4$
- Random variable R: number of red legos in the bag.
- Domain $D_R = ?$ $\{0, 1, 2, 3, 4\}$
- Assume *uniform prior* on R (all values equally likely): $\frac{1}{5} = .2$
- Random variable L_0: color of first lego we draw out of the bag
- *Observation model:*

$$\Pr(L_0 = \text{red} \mid R = n) = ?$$

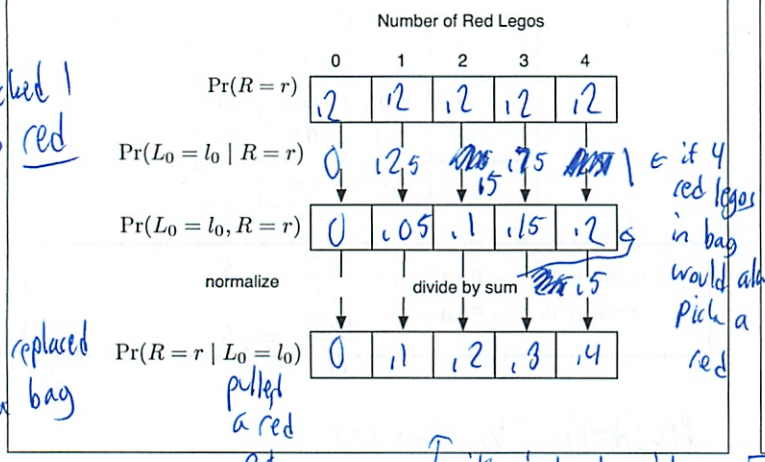
$$\Pr(L_0 = \text{white} \mid R = n) = 1 - \Pr(L_0 = \text{red} \mid R = n)$$

- We want to know:

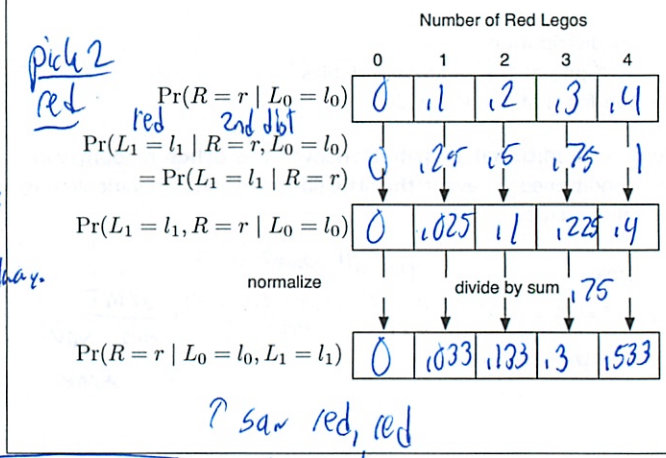
$$\Pr(R = n \mid L_0 = \text{whatever color we observed})$$

Bayes!!

What do we know after first Lego draw?



What do we know after second Lego draw?



So now what is $P(L_2 = \text{red} \mid L_0 = \text{red}, L_1 = \text{red}) = .83225$

It was 3 red/4 not quite what we would predict w/ 2 observations

with 20 observations closer to 3/4

Lecture over

Hidden Markov Models

System with a state that changes over time, probabilistically.

- Discrete time steps $0, 1, \ldots, t$
- Random variables for states at each time: $S_0, S_1, S_2, \ldots$
- Random variables for observations: $O_0, O_1, O_2, \ldots$

State at time t determines the probability distribution:

- over the observation at time t
- over the state at time $t + 1$

Hidden Markov Models

System with a state that changes over time, probabilistically.

- Discrete time steps $0, 1, \ldots, t$
- Random variables for states at each time: $S_0, S_1, S_2, \ldots$
- Random variables for observations: $O_0, O_1, O_2, \ldots$
- **Initial state distribution:**

$$\Pr(S_0 = s)$$

- **State transition model:**

$$\Pr(S_{t+1} = s \mid S_t = r)$$

- **Observation model:**

$$\Pr(O_t = o \mid S_t = s)$$

Inference problem: given actual sequence of observations $o_0, \ldots, o_t$, compute

$$\Pr(S_{t+1} = s \mid O_0 = o_0, \ldots, O_t = o_t)$$

Bayes Jargon

- Belief – a probability distribution over the states
- Prior – the initial belief before any observations

Are my leftovers edible?

- $D_{S_t} = \{\text{tasty}, \text{smelly}, \text{furry}\}$
- $\Pr(S_0 = \text{tasty}) = 1; \Pr(S_0 = \text{smelly}) = \Pr(S_0 = \text{furry}) = 0$
- State transition model:

		S_{t+1}		
		T	S	F
	T	0.8	0.2	0.0
S_t	S	0.1	0.7	0.2
	F	0.0	0.0	1.0

- No observations
- What is $\Pr(S_4 = s)$?

State transition update

$$\Pr(S_{t+1} = s) = \sum_{r \in D_S} \Pr(S_{t+1} = s \mid S_t = r) \Pr(S_t = r)$$

This Week

No software lab

Midterm 2:

- **Tonight!! 7:30–9:00PM:**
  - Abdussabur to Hernandez in 32-141;
  - Honickman to Sepp in 32-155;
  - Simon to Zelko in 56-114
- Any printed material okay; calculators okay
- Computers, cellphones, music players not allowed

Design lab: None due to Institute holiday

HW Week 10

11/13

- Probability
- (so long since I last did this!)
- (I don't think we've ever do this topic in 6.041)

#1 DDist

- represent + manipulate probability ~~dictionaries~~ distributions over discrete sets. DDist stores distributions in a dictionary

- init w/ dictionary → keys are values of sample set
- values are probabilites (all sum to 1)
  (how would I have designed this)

- prob method takes an element + returns probability assiged to it by distribution. If its not there → 0
- (ok just returns the key)
- support returns list of elements w/ 0 probabilities

- Create a really simple DDist
- (should be a print method in DDist
  - support does not do it
    - just prints the items

I like how they test

(2)

ans = [foo.prob('hi'), foo.prob('mid'), foo.prob('lo')]

- could build w/ support

10.1.2 Conditional Dist

$P(B|A)$

- not distributions themselves
- functions from values A can take on distributed over B

- bar = $P(B|A)$
- bar(a) = $P(B|A=a)$
- Year = {1, 2, 3, 4}
- Grade = {a, b, c, d, f}
- P(grade | Year) = PGgY
  - always give a value for year
- PGgY(1) → DDist(a: .3, b: .3, c: .3, d: .07, f: .03)
  - so what is it doing $\frac{P(B \cap A)}{P(A)}$
  - but we don't have #

(3)

Disease = {'disease', 'noDisease'}

Test = {'posTest', 'negTest'}

want conditional distribution ~~of Dis~~ takes value of Disease + returns a DDist that represents dist of Test conditioned on Disease

P(posTest | disease) = .98

P(posTest | noDesease) = .05

So PTgD('disease').prob(posTest) = .98

- ? so what in math notation do they want us to implement exactly ?

So

test	dis	no dis
+	.98	.05
−		

← box must add to 1

So want this Conditional

first P(neg | dis) = 1 − .98 = .02

P(neg | nodis) = 1 − .05 = .95

) no-not Conditional used!

4

	des	nodes
+		
−		

P(disease, postest) = P(disease) P(pos test | disease)

don't have that

; is this bayes rule

P(pos test | Disease = disease) = .98

oh they just want us to represent

if ~~Dise~~ diseaseValue = "disease":

return DDist (posTest .98 , negTest .02)

else:

return DDist (pos test .05, negTest .95)

better if do if else then else return 0

- thats if;

- just hard code - don't derive;

- maybe later we will do

its like that in lecture notes

⑤ 10.1.3 Joint Dist

$P(A=a, B=b) = P(A=a)\, P(B=b \mid A=a)$

remember we had

$P(\text{posTest} \mid \text{disease}) = .98$

$P(\text{posTest} \mid \text{noDisease}) = .05$

Part 1 JDist Example

if Disease is defined as

Disease = DDist({ disease : .0001, noDisease : .9999

want conditional (PTgD)

what is joint

(ok here is what I wanted to do on last problem)

$P(\text{no disease}, \text{posTest}) = P(\text{no disease})\, P(\text{posTest} \mid \text{no disease})$

$.9999 \qquad .05$

$= .049995$ ②

(so cool - but kinda unintuitive)

(e)

$P(\text{disease}, \text{pos test}) = P(\text{disease})\, P(\text{pos test} \mid \text{disease})$

$= .0001 \quad .98$

$= .000098$ ✓

$P(\text{no disease}, \text{neg test}) = P(\text{no disease})\, P(\text{neg test} \mid \text{no disease})$

$= .9999$ a bit harder

$\cdot (1 - .05)$

$= .9999 \cdot .95$

$= .949905$ ✓

$P(\text{disease}, \text{neg test}) = P(\text{disease}) \cdot P(\text{neg test} \mid \text{disease})$

$= .0001 \cdot (1 - .98)$

$= .0001 \cdot .02$

$= .000002$ ✓

¿would it work in reverse?

$P(\text{negtest}, \text{disease})$

- yeah order does not matter
- its just that table

		des	nodes
test	+		
	−		

$= P(\text{negtest})\, P(\text{disease} \mid \text{negtest})$ ← if had that data ✓ continued in slides

$= P(\text{disease})\, P(\text{neg test} \mid \text{disease})$

(7) Part 2 JDist Marginalization Example

- marginalize it out
- $P(A=a) = \sum_b P(A=a, B=b)$

(these concepts make more sense to me know
- this class taught better than 641)

	des	nodes	
+	.000098	.049995	.050093
−	.949905	.00002	.949907
	.950003	.049997	

sum up rows + columns

? are these values we had before ~~yes~~
yes

	des	no des	
+			P(pos test)
−			P(neg test)
	P(des)	P(nodes)	

just w/ rounding errors

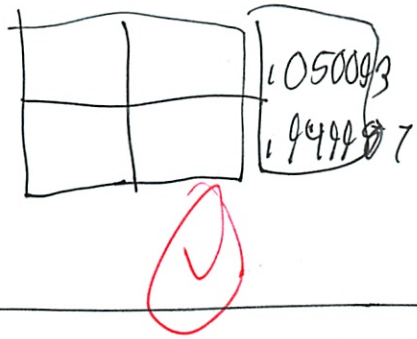
- no can't be

marginalize disease
~~JDist~~ ~~pos~~

? so what is that ??
$P(Des) = \{$ pos test ?

(8) ~~So what is this?~~

So



.050093 = P that test will be ⊕ (not knowing if disease)

.949907 = " ⊖

10.1.4 Operations on Conditional Distributions

- same disease problem as before

Part 1 Bayesian update

In many cases have an original distribution over some RV $P(A)$ and given evidence that RV B has b (ie: we know prob of disease/no disease and then get test results back [with actual ⊕/⊖ result])

Or we know $P(B|A)$ but want $P(A|B=b)$

So we want $P(\text{Disease} \mid \text{posTest})$

Bayesian

$$= \frac{P(\text{pos test} \mid \text{disease})\, P(\text{Disease})}{P(\text{test} = \text{pos test})}$$

(9)

$$= \frac{.98 \cdot .0001}{.050093}$$

← ~~Get P test to post~~ we know test was pos ← just like normal!

? from last problem

(I think there ~~is~~ must be an easier way in this case)

- no not in this one

$= .001956$ ✓

- this is what we actually want in problem, right?

~~P(Disease | neg test~~

$P(\text{No Disease} \mid \text{postest}) =$

$$= \frac{P(\text{pos test} \mid \text{no disease})\, P(\text{no disease})}{P(\text{pos test})}$$

$$= \frac{.05 \cdot .9999}{\cancel{.94997}}$$

See next pg – this is $P(\ominus \text{test})$

$= .052631$ ✗

but I did the same thing as before?

- ah distribution must $= 1$

(10)

~~1 - .0015~~

1 - .001956

.998044 ✓

But why didn't it work out the same then??

Oh duh, I did prob of ⊖ test

$$= \frac{.05 \cdot .9999}{.050093}$$

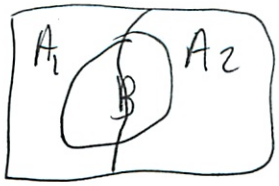
$= .998043$ ✓ much better

Part 2 Total Probability

- law of total probability

$$P(B) = \sum_a P(B \mid A = a)\, P(A = a)$$

from 6.041



$P(B \mid A_1) P(A_1) + P(B \mid A_2) P(A_2)$

$= P(B)$

(11)

Want P(Test) using P(Test | Disease) P(Disease)

↳ { postest
  negtest

↳ DDist [postest ___ , negtest ___]

P(+ test | disease) P(disease) + add the two

P(+ test | no disease) P(no disease) right?

= could just look at marginals, right?

= .001956 · .0001 + .998043 · .9999

= .0000001956 + .9979431 = .997943 ⊗

~~= .998043 · .9999~~

Used wrong #

Used P(disease | postest)

P(+ test | disease) P(disease) + P(+ test | no disease) P(no disease)

.98 · .0001 + .05 · .9999

.000098 + .0499995

.050093 ⊙ Yeah same as adding accross

(2)

$P(\ominus \text{ test} \mid \text{disease})\, P(\text{disease}) + P(\ominus \text{ test} \mid \text{no disease})\, P(\text{no disease})$

$1 - .98 \cdot .0001 + 1 - .05 \cdot .9999$

$.02 \cdot .0001 + .95 \cdot .9999$

$.000002 + .949905$

$.949907$

10.1.5 Where are you?

Trying to locate someone in 2 story building, 4 rooms each (?factored)

f1, f2 r1, r2, r3, r4

- equally likely f1, f2
- f1: equally likely in any room
- f2: $P(r4) = .7$ $P(r1, r2, r3) = .1$

Oh have to format - nice practice

a) P(floor) = DDist(f1: .5, f2: .5) ✓

(13)

b) $P(room \mid floor = f1) = DDist(r1: .25, r2: .25, r3: .25, r4: .25)$ ✓

c) $P(room \mid floor = f2) = DDist(r1: .1, r2: .1, r3: .1, r4: .7)$ ✓

d) Joint Distribution (room, floor) =

DDist (

$f1, r1 = .5 \cdot .25 = .125$

$f1, r2$ "

$f1, r3$ "

$f1, r4$ "

$f2, r1 = .5 \cdot .1 = .05$

$f2, r2 =$ "

$f2, r3 =$ "

$f2, r4 = .5 \cdot .7 = .35$

✓

e) Now we know he is in R1. What is $P(floor \mid room = r1)$

$$= \frac{P(room = r1 \mid floor)\, P(floor)}{P(room = r1)}$$

(14)

So for f1

$$= \frac{P(r1 \mid f1)\, P(f1)}{P(r1)}$$

$$= \frac{.25 \cdot .5}{.175}$$ ← need this $\sum_{n=1}^{2} P(r1 \mid fn)\, P(fn)$

$$= P(r1 \mid f1)P(f1) + P(r1 \mid f2)P(f2)$$

$$= .25 \cdot .5 + .1 \cdot .5$$

$$= .175$$

$$= .714$$

So for f2

$$= \frac{P(r1 \mid f2)\, P(f2)}{P(r1)}$$

$$= \frac{.1 \cdot .5}{.175}$$

$$= .285$$ ✗

$$= .286$$ ✓ ↓ rounding

Was there an easier way to get this?

Could have used table - but still would have been same steps

(15) Optional Problems

10.2.1 A distribution

		B = 0	B = 1	
A	0	.1	.2	.3
	1	.3	.4	.7
		.4	.6	

$P(A=0) = .3$ ✓ marginals

$P(A=1) = .7$ ✓

$P(B=0) = .4$ ✓

$P(B=1) = .6$ ✓

$P(A=0 \mid B=1) =$ just read off table

$= .2$

no - must add to 1

$$\frac{P(A=0 \cap B=1)}{P(B=1)}$$ ← what I got as .2 ← normalize!

$$= \frac{.2}{.6} = \frac{1}{3}$$ ✓

(16)

$P(A=0 \mid B=0) = \frac{.1}{.4} = .25$ ✓

$P(A=1, B=1) = P(A=1 \cap B=1) =$ just off table now

$= .4$ ✓

$P(A=1 \text{ or } B=1)$ so this need to add $\underset{\text{marginals}}{P(A=1) + P(B=1)}$

$= .7 + .6$

$\therefore$ more than 1

oh right double counting

$= \underset{\text{marginals}}{P(A=1) + P(B=1)} - \underset{\text{from table}}{P(A=1 \cap B=1)}$

$= .7 + .6 - .4$

$= .9$

(nice pratice)

(17) 10.2.2 Summing to 1

What will always sum to 1 no matter distribution (no checking!)

A, B, binary (so same as last time)

a) $P(A=0)$ and $P(A=1)$ T ✓

b) $P(A=0 \mid B=0) + P(A=1 \mid B=0)$ ← Should also just see!

Can also just do

$\frac{.4}{.4} = 25 + \frac{.3}{.4} = .75$ ← ch I used wrong value F ✗

c) $P(A=0 \mid B=0) + P(A=0 \mid B=1)$

think F

$.5 + \frac{P(A=0 \cap B=1)}{P(B=1)} = \frac{.2}{.6}$ F ✓

d) $P(A=0, B=0) + P(A=0, B=1) + P(A=1, B=0) + P(A=1, B=1)$

all possibilities T ✓

e) $P(A=0) + P(B=0)$

think F

$.3 + .4$ F ✓

(18) 10.2.3 Truth

More of the same

a) $P(A=1) \geq P(A=1, B=1)$

think ~~are~~ true

$.7 \geq .4$ T

b) $P(A=1) \leq P(A=1, B=1)$

Opposet (well = sign) F

c) $P(A=1) = P(A=1, B=1) + P(A=1, B=0)$

think T

$.7 = .4 + .3$ T

d) $P(A=1, B=1) = P(A=1)\,P(B=1)$

Yeah - if ind

but not ind so think F

$.4 = .7 \cdot .6$

$.4 = .42$ F

e) $P(A=1 \mid B=1) \geq P(A=1)$

Can be either way F

$\frac{.4}{.6} \geq .7$

$.33 \geq .7$ F in this case

so F always

19

oh I wrote it wrong

f) $P(A=1 \mid B=1) \geq P(A=1)\,P(B=1)$ $\quad P(A=1 \cap B=1)$

think depends on case again

$\frac{14}{16} \geq \cancel{.7 \cdot .6}$ F

14

So why is that always true

- since we have more info on B in first

~~P(A) will be larger than P(A).~~ bad

I kinda see it

10.2.4 Equavilance

What is = to $P(A=0 \mid B=0)$

So $\dfrac{P(A=0 \cap B=0)}{P(B=0)}$

in our case $\frac{1}{14} = \frac{1}{4}$

a) $\dfrac{P(A=0)}{P(B=0)}$ think F

$\frac{13}{14}$ F

(20) I think most will be false

b) $P(B=0 \mid A=0) =$

$$\frac{P(A=0 \cap B=0)}{A=0}$$ ← good

F

$$\frac{.1}{.3}$$

c) $P(B=0 \mid A=0)\, P(A=0)$

this is $P(B=0)$ F

$$\frac{1}{3} \cdot .3 = \frac{1}{3} \cdot \frac{3}{10} = \frac{3}{30} = \frac{1}{10}$$

d) $$\frac{P(B=0 \mid A=0)\, P(A=0)}{P(B=0)}$$

these are getting more elaborate

oh Bayes rule – yes I think

$$\frac{\frac{1}{10}}{\frac{4}{10}} = \frac{1}{10} \cdot \frac{10}{4} = \frac{1}{4} \quad T$$

e) $$\frac{P(B=0 \mid A=0)\, P(A=0)}{P(A=0, B=0) + P(A=1, B=0)}$$ ← yeah just TPT T

$$\frac{\frac{1}{10}}{.1 + .3} \quad T$$

(2)

f) $\dfrac{P(B=0 \mid A=0)P(A=0)}{P(B=0 \mid A=0)P(A=0) + P(B=0 \mid A=1)P(A=1)}$ ← so this is $P(B=0)$ which is T ✓

$\cancel{\frac{1}{10}}$ not going to ✓

g) $\dfrac{P(A=0, B=0)}{P(A=0)} = \dfrac{P(A=0 \cap B=0)}{P(A=0)}$ F ✓

$\frac{.1}{.3}$ F

h) $\dfrac{P(A=0, B=0)}{P(B=0)}$ T here we go ✓

i) $\dfrac{P(A=0, B=0)}{P(B=0, A=1) + P(B=0, A=0)}$ ← tpt so T ?

$\frac{.1}{.3+.1}$ T ✓

(22) 10.2.5 Buying a Car

Buying a used car

$P(\text{good}) = .7$

Garage tests it

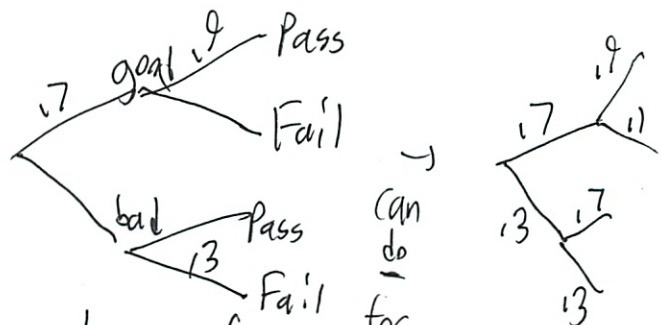
$P(\text{pass} \mid \text{good}) = .9$

$P(\text{pass} \mid \text{bad}) = .3$

So can fill in table

	good	bad
pass		
fail		
	.7	

can also think of it like tree



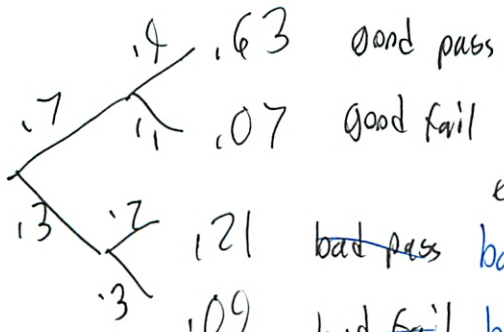
← all can do so far

$P(\text{Pass}, \text{good}) = P(\text{pass} \mid \text{good})\, P(\text{good})$

$.9 \cdot .7$

$P(\text{Pass}, \text{fail}) =$

Oh can just go down tree



.63 good pass

.07 good fail

← i shan prof

.21 ~~bad pass~~ bad fail

.09 ~~bad fail~~ bad pass

(23)

Can now fill in table

	good	bad	
pass	.63	~~.10~~ .21	.09
Fail	.07	~~.09~~	.21

~~a) P(fail | good)~~

		~~.84~~ .72	
		~~.16~~ .28	
.7	.3		

a) $P(\text{Fail} \mid \text{good}) = \dfrac{P(\text{fail} \cap \text{good})}{P(\text{good})} = \dfrac{.07}{.7} = .1$ ✓

b) $P(\text{pass}) = .84$ ✗

c) $P(\text{good} \mid \text{pass}) = \dfrac{P(\text{good} \cap \text{pass})}{P(\text{pass})} = \dfrac{.63}{.84} = .75$ ✗

alt way

b) $P(\text{pass}) = P(\text{pass} \mid \text{good})\, P(\text{good}) + P(\text{pass} \mid \text{bad})\, P(\text{bad})$

$.9 \cdot .7 + .3 \cdot .3$

$.63 + .09$

$.72$ ✓

Oh I flipped things on free fixed

24

c) $= \frac{.63}{.72} = .875$ ✓

11/16

6.01: Introduction to EECS I

Bayesian estimation, etc.

Week 11 *November 16, 2010*

Reading: Chapter 8

Every system up to now has been deterministic
circuit changed a little bit
having more sonar values = more confident

What did we see last time?

- We introduced fundamental concepts from probability
- We saw an example of using Bayes reasoning to refine our belief in the state of a system
- We are going to briefly recap that discussion, then use these tools to reason about systems that operate under uncertain informationn which changes over timme

moving a model of the plant in the controller to let us reason about them

Important ideas

PCAP:

- Combine individual pieces of noisy information into a bigger picture that improves over time.
- Combine simple probability distributions to make more complex ones (HW 4).

Motivation: Traditional approaches try to design robust controller, under expectation that designer can plan for all contingencies.

Modeling: Better solution: Put a model of the plant inside the controller. The **system** reasons about the outside world: provides a way for planning under uncertainty, by incorporating noisy observations.

Probability

Probability theory lets us

- Make precise statements about uncertainty – assign numerical estimates of uncertainty to events
- Combine uncertain information to reach consistent (uncertain) conclusions – do "calculus" on uncertainties to deduce information about the state of the system
- Win bets

Subjective view of probability as personal belief:

- Probability is not restricted to long run frequency of events;
- Probability can provide a framework for capturing beliefs about the state of the world.

here about beliefs

Basic elements

- Given a set of one or more variables that describe system:
  - determine the domain of each variable (set of possible values)
  - determine state space (set of all possible values for variables)
- An event is a subset of the state space
  - it describes a subset of the space of all possible states of the system
- A probability **distribution** Pr assigns numbers to events. It satisfies the following axioms:

$$\Pr(U) = 1$$ domain

$$\Pr(\{\}) = 0$$

$$\Pr(E_1 \cup E_2) = \Pr(E_1) + \Pr(E_2) - \Pr(E_1 \cap E_2)$$ no double counting

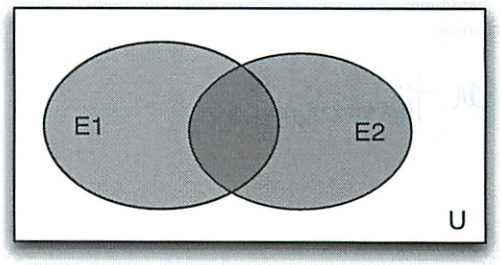
- Sum of probabilities assigned to the atomic events must be 1.
- Calculus of probability distributions will allow us to reason about states of system

Conditional Probability

What is probability of event E_1, given that we know event E_2 is true?

$$\Pr(E_1 \mid E_2) = \frac{\Pr(E_1 \cap E_2)}{\Pr(E_2)} .$$

given that I know something is true



Restrict attention to E_2
Ask: In what fraction of E_2 is E_1 true?

Random variables

when having too many different states

Some states more easily described in factored form (e.g,. 10 coins each heads or tails rather than all 2^{10} possible sequences).

Similarly, may want to describe probability distributions over one or more variables, rather than over space of events.

A **random variable**, V, is specified by:

- a domain, D_V, which is the set of values, v_i, it can take
- a probability distribution, $\Pr(V = v_i)$, which specifies the probability of it taking on any of its values

- So factor/combine RVs
- have 10 coin flips
- 2^{10} results

$$\sum_{v_i \in D_V} \Pr(V = v_i) = 1$$

A probability distribution over one state variable.

- So look at each coin individually

Joint probabability distributions

A **joint probability distribution** multiple dimensions going on at once

- Defined over two or more random variables; e.g., V and W
- Written $\Pr(V, W)$, where V and W are random variables
- Domain is *cartesian product* of D_v and D_W

  $\{(v, w) \mid v \in D_V, w \in D_W\}$
- Probability is assigned to (v, w) pairs
- The sum of the probabilities must be 1

We are generally interested in **dependent** variables, not independent variables. influenced by

- if indp. can seperate the space out

Conditional Probability Distributions

A **conditional probability distribution**

given I know

- Written $\Pr(V \mid W)$, where V and W are random variables,
- Is a function from values, w, of W, to probability distributions on V.

$$\Pr(V \mid W) = \lambda w. \Pr(V \mid W = w)$$

where λ is the Python `lambda` for creating functions.

Conditional Probability

Consider two random variables:

- D, which has values: *disease*, *nodisease*
- T, (a test result) which has values: *pos*, *neg*

Efficacy of test is specified by $\Pr(T \mid D)$:
a conditional distribution on the test results given whether a person has the disease.

distributions

$$\Pr(T \mid D) = \begin{cases} \{pos : 0.99, neg : 0.01\} & \text{if } D = disease \\ \{pos : 0.001, neg : 0.999\} & \text{if } D = nodisease \end{cases}$$

Often we are interested in inferring the other conditional probability (i.e. probability of having disease, given test result $\Pr(D \mid T)$).

To derive this, we will typically use Bayes reasoning, for which we need a set of operations on distributions.

Operations on Distributions

- Constructing a joint distribution
- Marginalization
- Conditioning

So can think Baesianly

Constructing a Joint Distribution

Given

- a distribution $\Pr(V)$ and
- a conditional distribution $\Pr(W \mid V)$

Produce a **joint distribution** by computing, for every pair of values v in the domain of V and w in the domain of W, a product of the two factors.

need to do for every pair of actual values

$$\Pr(V = v, W = w) = \Pr(V = v) \Pr(W = w \mid V = v)$$

similarly

$$\Pr(V = v, W = w) = \Pr(W = w) \Pr(V = v \mid W = w)$$

So Bayes is just computation

$$P(V=v \mid W=w) = \frac{P(V=v)P(W=w \mid V=v)}{P(W=w)}$$

Check Yourself

- $\Pr(C = T) = 0.15$ $\Pr(C = F) = 0.85$
- $\Pr(A = T \mid C = T) = 0.3333$ $\Pr(A = F \mid C = T) = 0.6666$
- $\Pr(A = T \mid C = F) = 0.0588$ $\Pr(A = F \mid C = F) = 0.9412$

		C: T	C: F	
A	T	0.05	0.05	.1
	F	0.1	0.8	.9
		.15	.85	

Marginalization

Given

- a joint distribution
- index of one of the random variables

Produce a **marginal distribution** over the chosen random variable by summing over all assignments to the other random variables in the joint distribution.

$$\Pr(A = a) = \sum_c \Pr(A = a, C = c)$$

$$\Pr(C = c) = \sum_a \Pr(A = a, C = c)$$

Check Yourself

		C: T	C: F	
A	T	0.05	0.05	0.1
	F	0.1	0.8	0.9
		0.15	0.85	

Here are the marginal distributions.

NOT INDEPENDENT so

Notice that $\Pr(A = T, C = T) \neq \Pr(A = T)\Pr(C = T)$ for example.

are connected somehow

Conditioning

Given

- a joint distribution
- index of one of the random variables
- a value for that random value

Produce a **conditional distribution** over the **other** random variable(s) conditioned on event that the specified random variable has the specified value.

For example

$$\Pr(C \mid A = T) = \{\frac{\Pr(A = T, C = T)}{\Pr(A = T)}, \frac{\Pr(A = T, C = F)}{Pr(A = T)}\}$$

This is a distribution on C.

Part of Bayes

Conditional Dentistry

		C: T	C: F	
A	T	0.05	0.05	0.1
	F	0.1	0.8	0.9
		0.15	0.85	

- What is $\Pr(C = T \mid A = T)$? 0.05/0.1 = 0.5
- What is $\Pr(C = F \mid A = T)$? 0.05/0.1 = 0.5

Note that this is just normalizing the row for $A = T$ (dividing by $\Pr(A = T)$ so that the entries sum to 1).

Total Probability

Given

- a distribution $\Pr(V)$ and
- a conditional distribution $\Pr(W \mid V)$

Produce a **distribution** over $\Pr(W)$

$$\Pr(W) = \text{MarginalizeOut}(\text{Joint}(\Pr(V), \Pr(W \mid V)), V)$$

That is, for each w in domain D_W

$$\Pr(W = w) = \sum_v \Pr(W = w \mid V = v)\Pr(V = v)$$

gets us back to before

Oral AIDS Test: OraSure (OMT)

http://www.bmc.org/hiv-aids/services/testingandcounseling.htm

This test is performed by collecting a sample from your mouth and sending it to the lab for processing...

The OraSure test has 98.6% sensitivity and 97.7% specificity.

If the OraSure test comes back positive, your counselor must draw your blood for confirmation.

C = Disease A = Test

want both - tradeoff

Sensitivity: $P(A = T \mid C = T) = 0.986$ given have AIDs, prob test comes back pos

Specificity: $P(A = F \mid C = F) = 0.977$ given no AIDs, prob test comes back -

Note point of Bayes: we may know $P(A = T \mid C = T)$, but we really want to know $P(C = T \mid A = T)$!!!

that actually have disease when test comes back +

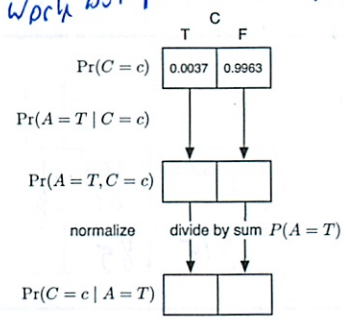
Oral AIDS Test: OraSure (OMT)

Prevalence of AIDS in US Population 2006

1.1 million case in 298 million people

$\Pr(C = T) = 0.0037$

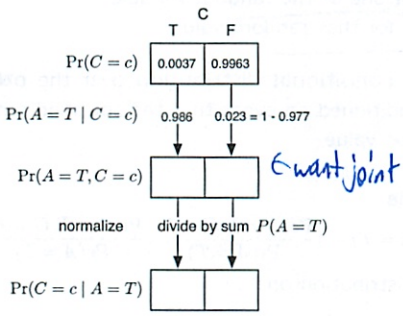
can work out prior probability of disease



Oral AIDS Test: OraSure (OMT)

Sensitivity: $P(A = T \mid C = T) = 0.986$

Specificity: $P(A = F \mid C = F) = 0.977$

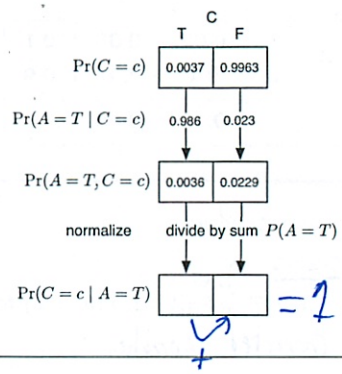


Oral AIDS Test: OraSure (OMT)

Compute Joint Distribution:

$$\Pr(A = T, C = T) = \Pr(A = T \mid C = T)\Pr(C = T)$$

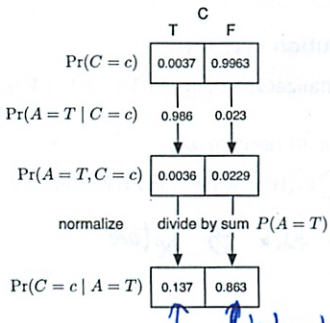
$$\Pr(A = T, C = F) = \Pr(A = T \mid C = F)\Pr(C = F)$$



Oral AIDS Test: OraSure (OMT)

Conditioning (by normalizing)

$$\Pr(A = T) = \Pr(A = T, C = T) + \Pr(A = T, C = F)$$



test not all that usefull if comes back (+, only have 14% chance of actually having AIDS

Remember our bet from last time?

- Total number of legos in the bag $= 4$
- Random variable R: number of red legos in the bag.
- Domain $D_R = \{0, 1, 2, 3, 4\}$
- Assume *uniform prior* on R (all values equally likely):
- Random variable L_0: color of first lego we draw out of the bag
- *Observation model:*

$$\Pr(L_0 = \text{red} \mid R = n) = \frac{n}{4}$$

$$\Pr(L_0 = \text{white} \mid R = n) = 1 - \Pr(L_0 = \text{red} \mid R = n)$$

- We want to know:

$$\Pr(R = n \mid L_0 = \text{observed color}, L_1 = \text{observed color})$$

function on what we know

Bayes!!

What do we know after two red trials?

– what happened last week

	Number of Red Legos 0	1	2	3	4	
Pr(R = r)	.2	.2	.2	.2	.2	
Pr(L_0= red | R = r)	0	.25	.5	.75	1	
Pr(L_0= red, R = r)	0	.05	.1	.15	.2	Sum = .5
normalize						
Pr(R = r | L_0= red)	0	.1	.2	.3	.4	
Pr(L_1= red | R=r,L_0= red) = Pr(L_1= red | R=r)	0	.25	.5	.75	1	
Pr(L_1= red ,R=r | L_0= red)	0	.025	.1	.225	.4	Sum = .75
normalize						
Pr(R = r | L_0= red,L_1=red)	0	.0333	.1333	.3	.5333	

mislead → what it meant

What do we know after one red and one white trial?

	Number of Red Legos 0	1	2	3	4	
Pr(R = r)	.2	.2	.2	.2	.2	
Pr(L_0= red | R = r)	0	.25	.5	.75	1	
Pr(L_0= red, R = r)	0	.05	.1	.15	.2	Sum = .5
normalize						
Pr(R = r | L_0= red)	0	.1	.2	.3	.4	
Pr(L_1=white | R=r,L_0= red) = Pr(L_1= white | R=r)	1	.75	.5	.25	0	
Pr(L_1= white,R=r | L_0=red)	0	.075	.1	.075	0	Sum = .25
normalize						
Pr(R=r | L_0=red,L_1=white)	0	.3	.4	.3	0	

same order of history does not matter

What do we know after one white and one red trial?

	Number of Red Legos 0	1	2	3	4	
Pr(R = r)	.2	.2	.2	.2	.2	
Pr(L_0= white | R = r)	1	.75	.5	.25	0	
Pr(L_0= white, R = r)	.2	.15	.1	.05	0	Sum = .5
normalize						
Pr(R = r | L_0= white)	.4	.3	.2	.1	0	
Pr(L_1=red | R=r,L_0=white) = Pr(L_1= red | R=r)	0	.25	.5	.75	1	
Pr(L_1=red,R=r | L_0=white)	0	.075	.1	.075	0	Sum = .25
normalize						
Pr(R=r | L_0=white,L_1=red)	0	.3	.4	.3	0	

What do we know after two white trials?

	Number of Red Legos 0	1	2	3	4	
Pr(R = r)	.2	.2	.2	.2	.2	
Pr(L_0= white | R = r)	1	.75	.5	.25	0	
Pr(L_0= white, R = r)	.2	.15	.1	.05	0	Sum = .5
normalize						
Pr(R = r | L_0= white)	.4	.3	.2	.1	0	
Pr(L_1=white | R=r,L_0=white) = Pr(L_1= red | R=r)	1	.75	.5	.25	0	
Pr(L_1=red,R=r | L_0=white)	.4	.225	.1	.025	0	Sum = .75
normalize						
Pr(R=r | L_0=white,L_1=white)	.5333	.3	.1333	.0333	0	

So how safe was our bet?

no observation →

	Number of Red Legos 0	1	2	3	4
Pr(R = r)	.2	.2	.2	.2	.2
Pr(R = r | L_0= red,L_1=red)	0	.0333	.1333	.3	.5333
Pr(R=r | L_0=red,L_1=white)	0	.3	.4	.3	0
Pr(R=r | L_0=white,L_1=red)	0	.3	.4	.3	0
Pr(R=r | L_0=white,L_1=white)	.5333	.3	.1333	.0333	0

← summary

$$\Pr(L_2=\text{red}\mid L_0,L_1) = \Pr(L_2=\text{red}\mid R=0)\Pr(R=0\mid L_0,L_1) + \ldots + \Pr(L_2=\text{red}\mid R=4)\Pr(R=4\mid L_0,L_1)$$

In our case, this was 0*0 + .0333*.25 + .1333*.5 + .3*.75 + .5333*1 = .8333

↑ was reasonable to do the bet w/ no info – would be 50-50

Hidden Markov Models

state of system fixed – always 3 red were in bag

System with a state that changes over time, probabilistically.

- Discrete time steps $0, 1, \ldots, t$
- Random variables for states at each time: $S_0, S_1, S_2, \ldots$
- Random variables for observations: $O_0, O_1, O_2, \ldots$

State at time t determines the probability distribution:

- over the observation at time t
- over the state at time $t+1$

(State did not change
only belief changes

not always true – our robot is always moving

Hidden Markov Models

System with a state that changes over time, probabilistically.

- Discrete time steps $0, 1, \ldots, t$
- Random variables for states at each time: $S_0, S_1, S_2, \ldots$
- Random variables for observations: $O_0, O_1, O_2, \ldots$
- **Initial state distribution:**

$$\Pr(S_0 = s)$$

- **State transition model:**

$$\Pr(S_{t+1} = r \mid S_t = s)$$

- **Observation model:**

$$\Pr(O_t = o \mid S_t = s)$$ that I get "the" observation

Inference problem: given actual sequence of observations $o_0, \ldots, o_t$, compute

$$\Pr(S_{t+1} = s \mid O_0 = o_0, \ldots, O_t = o_t)$$

hidden = don't know current state

Are my leftovers edible?

- $D_{S_t} = \{\text{tasty}, \text{smelly}, \text{furry}\}$
- $\Pr(S_0 = \text{tasty}) = 1; \Pr(S_0 = \text{smelly}) = \Pr(S_0 = \text{furry}) = 0$ ← 1st state
- State transition model:

		S_{t+1}		
		T	S	F
	T	0.8	0.2	0.0
S_t	S	0.1	0.7	0.2
	F	0.0	0.0	1.0

- No observations
- What is $\Pr(S_4 = s)$?

.8 .7

T .2 → S .2 → F ↺ 1

this is normal markoff model (last week 6.041)

State transition update

all of the tails

$$\Pr(S_{t+1} = s) = \sum_{r \in D_S} \Pr(S_{t+1} = s \mid S_t = r) \Pr(S_t = r)$$

My way

Tasty $(.8)^4$

Smelly $(.8)^3 \cdot .2 + (.8)^2 \cdot .2 \cdot .7 + (.8) \cdot .2 \cdot (.7)^2$

Furry $(.8)^2 \cdot .2 \cdot .2 + .8 \cdot .2 \cdot .7 \cdot .2 + .8 \cdot .2 \cdot .2 \cdot 1$

this way

- separate sheet

Copy Machine

Initial state distribution

	S: good	bad
$P(S_0 = s)$	0.9	0.1

initial

binary state

State transition model

	S: good	bad
$P(S_{t+1} = s \mid S_t = good)$	0.7	0.3
$P(S_{t+1} = s \mid S_t = bad)$	0.1	0.9

taking observation

Observation model

	O: perfect	smudge	black
$P(O_t = o \mid S_t = good)$	0.8	0.1	0.1
$P(O_t = o \mid S_t = bad)$	0.1	0.7	0.2

in real life - have observation

have state transitions, need to build conditions

* can't see what state machine is in

A perfect copy!

Step 1: Bayes Evidence:

Build joint distribution:

$$\Pr(S_0, O_0) = \Pr(O_0 \mid S_0) \Pr(S_0)$$ Observation · initial

		O: perfect	smudged	black
S	good	0.72	0.09	0.09
	bad	0.01	0.07	0.02

Condition on actual observation $O_0 = perfect$

$$\Pr(S_0 \mid O_0 = perfect) = \{\frac{\Pr(S_0 = good, O_0 = perfect)}{\Pr(O_0 = perfect)}, \ldots\}$$

$$\Pr(S_0 \mid O_0 = perfect) = \{good : 0.986, bad : 0.014\}$$

A perfect copy!

If we get a perfect copy, dist of state of machine →

	S: good	bad
$\Pr(S_0)$	0.9	0.1
$\Pr(O_0 = perfect \mid S)$	0.8	0.1
$\Pr(O_0 = perfect \text{ and } S_0)$	0.72	0.01
	divide by 0.73	
$\Pr(S_0 \mid O_0 = perfect)$	.986	.014

Time passes

Step 2: Total Probability:

Build joint distribution:

$$\Pr(S_0, S_1 \mid O_0 = \textit{perfect}) = \Pr(S_1|S_0)\Pr(S_0 \mid O_0 = \textit{perfect})$$

		S_1	
		good	*bad*
S_0	*good*	0.690	0.296
	bad	0.001	0.012

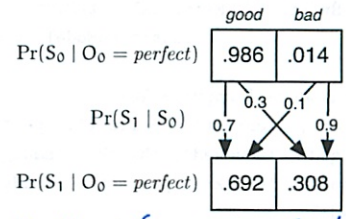
prob that states change

Marginalize out S_0:

$$\Pr(S_1 \mid O_0 = \textit{perfect}) = \sum_s \Pr(S_0 = s, S_1 \mid O_0)$$

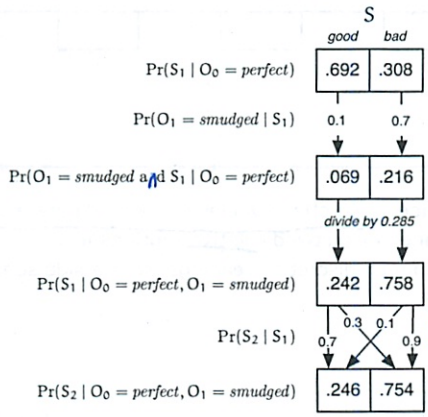
$$\Pr(S_1 \mid O_0 = \textit{perfect}) = \{\textit{good} : 0.691, \textit{bad} : 0.308\}$$

Time passes



Now prob of states in next time step (given 1st copy was perfect)

A smudged copy, another day



get a smudged day!

prob of system gotten perfect & transition 0 → 1 smudged!

never really did this in 6.041

Stochastic State Machine

There are no actions in a Hidden Markov Model. input

A Stochastic State Machine is like an HMM with actions. The state transition model now involves an input, that is, an action.

$$\Pr(S_{t+1} \mid S_t, I_t)$$

The initial state distribution and observation model are like an HMM.

It's the probabilistic generalization of a State Machine.

Python Model

```
initialStateDistribution = dist.DDist({'good': 0.9, 'bad': 0.1})
def observationModel(s):
    if s == 'good':
        return dist.DDist({'perfect' : 0.8,
                           'smudged' : 0.1, 'black' : 0.1})
    else:
        return dist.DDist({'perfect' : 0.1,
                           'smudged' : 0.7, 'black' : 0.2})
def transitionModel(i):
    def transitionGivenI(oldState):
        if oldState == 'good':
            return dist.DDist({'good' : 0.7, 'bad' : 0.3})
        else:
            return dist.DDist({'good' : 0.1, 'bad' : 0.9})
    return transitionGivenI
```

given the machinery

Python SSM

```
class StochasticSM(sm.SM):
    def __init__(self, startDistribution, transitionDistribution,
                 observationDistribution):
        self.startDistribution = startDistribution
        self.transitionDistribution = transitionDistribution
        self.observationDistribution = observationDistribution

    def startState(self):
        return self.startDistribution.draw()

    def getNextValues(self, state, inp):
        return (self.transitionDistribution(inp)(state).draw(),
                self.observationDistribution(state).draw())
```

give all 3 models

store

Using probability distributions inside a SM

Copy Machine Machine

```
copyMachine = ssm.StochasticSM(initialStateDistribution,
                               transitionModel, observationModel)

>>> copyMachine.transduce(['copy']* 20)
['perfect', 'smudged', 'perfect', 'perfect', 'perfect', 'perfect',
  'perfect', 'smudged', 'smudged', 'black', 'smudged', 'black',
  'perfect', 'perfect', 'black', 'perfect', 'smudged', 'smudged',
  'black', 'smudged']
```

can simulate copy machine
probabilistic - if run again will be different

Python State Estimation

```
class StateEstimator(sm.SM):
    def __init__(self, model):
        self.model = model
        self.startState = model.startDistribution
        self.obsD = self.model.observationDistribution
        self.transD = self.model.transitionDistribution

    def getNextValues(self, state, inp):
        (o, i) = inp
        sGo = dist.bayesEvidence(state, self.obsD, o)
        dSPrime = dist.totalProbability(sGo, self.transD(i))
        return (dSPrime, dSPrime)
```

↑ state

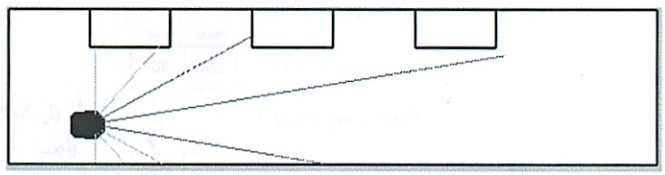
Note that we're making the state estimator be a state machine as well.

Where am I?

- **Mapping**: Assume you know where the robot is. Build a map of objects in the world based on sensory information at different robot poses.
- **Localization**: Assume you know where the objects in the world are (the map). Determine the robot's pose.
- **SLAM:** You know neither the location of the robot or of the obstacles. Do *simultaneous localization and mapping*.

if knew where robot is - could build map of world
- but often you don't know either were you are or what world looks like
- So SLAM: estimate both

One-dimensional localizer

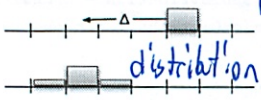


- **State space:** discretized values of x coordinate along hallway
- **Input space:** discretized relative motions in x
- **Output space:** discretize readings from a side sonar sensor

Transition model

$$\Pr(S_{t+1} = s' | S_t = s) = \Pr(S_{t+1} = s' \mid S^*(s) = s + \Delta)$$
$$= \mathrm{Tri}(s'; s + \Delta, \mathrm{hw}_s)$$

- Infer a **nominal action** based on the robot's odometry.



when robot moved
distribution of possibilities where it actually is

- Let x_t be the robot's observed x coordinate at time t.
- The nominal action that the robot took at time t is

$$\Delta = \mathrm{round}((x_t - x_{t-1})/w)$$

where w is width of a state bin.
- Assume nominal distance is all that matters; new state distribution is a triangular distribution, with half-width hw_s, centered on the nominal displacement.

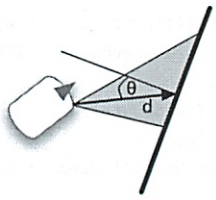
Observation model

prob that get that distance, given current location

$$\Pr(O = d | S = s) = \Pr(O = d | D^*(S) = d^*)$$
$$= \mathrm{Tri}(d; d^*, \mathrm{hw}_d)$$

- s is a robot state and d is a sonar reading
- Calculate **Nominal sonar distance:** $d^*(s)$

- is some noise - so approx



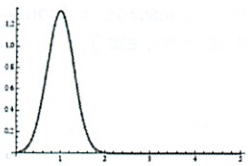
- Assume nominal distance is all that matters; observation distribution is a triangular distribution, with half-width hw_d, centered on the nominal distance.

"but isn't this discrete"

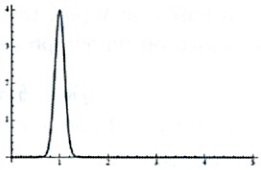
Continuous error distribution

Gaussian:

$$g(d, d^*, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{(d-d^*)^2/2\sigma^2}$$



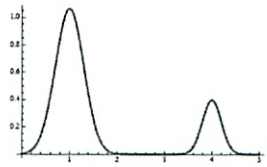
$\sigma = 0.3$



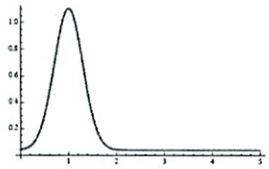
$\sigma = 0.1$

Mixture distributions

Mixture of Gaussians:



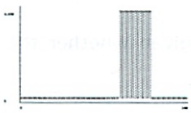
Mixture of uniform and Gaussian:



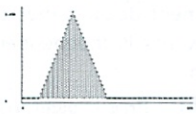
yeah here is discrete

Discrete error distributions

Primitive distributions:



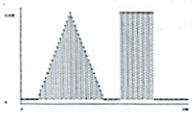
Sq(x; lo, hi)

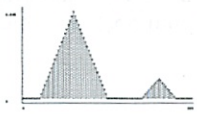


Tri(x; center, hw, lo, hi)

Mixture distributions:

$$\Pr_{mix}(x) = p\Pr(D_1 = x) + (1-p)\Pr(D_2 = x)$$



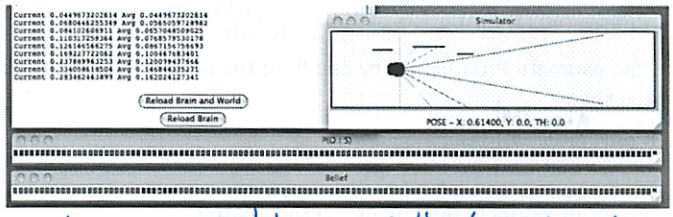


Must still sum to 1.

Localization = State Estimation

Define Stochastic State Machine

Apply standard state estimation algorithm.

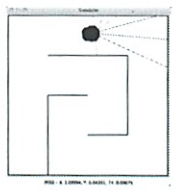


use it to get robot to estimate where is
get approx estimation -
using Bayes to reverse - given observation -
probabilities of each state

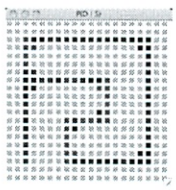
Extending to multiple dimensions

- State is x, y, θ
- Have to handle coordinate transforms instead of simple Δx
- Use all 8 sonars

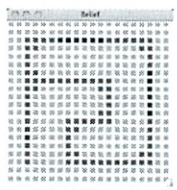
Full localizer: without and with angles



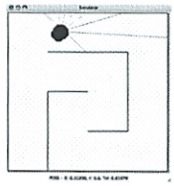
Simulator



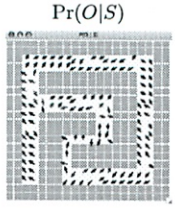
$\Pr(O|S)$



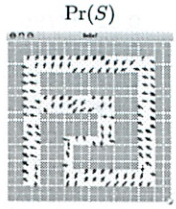
$\Pr(S)$



Simulator



$\Pr(O|S)$



$\Pr(S)$

plan where it should move

Applications of Bayesian Estimation

- spam filtering
- speech recognition
- medical diagnosis
- tracking aircraft on radar
- robot localization
- ...

Spam filtering

Given an email message, m, we want to know whether it is **Spam** or **Ham**.

Make this decision based on $W(m)$, the words in message m (could also use features based on typography, email address, etc.).

If

not spam

$$\Pr(\text{Spam}(m) = T \mid W(m)) > \Pr(\text{Spam}(m) = F \mid W(m))$$

then dump it in the trash.

Learning

We'd like to estimate $\Pr(\text{Spam}(m) \mid W(m))$ from past experience with email. But we'll never see the same W twice!

(won't send same email twice)

Use Bayes' rule:

$$\Pr(\text{Spam}(m) \mid W(m)) = \frac{\Pr(W(m) \mid \text{Spam}(m))\Pr(\text{Spam}(m))}{\Pr(W(m))}$$

We can estimate $\Pr(\text{Spam}(m))$ by counting the proportion of our mail that is spam.

given that

$\Pr(W(m) \mid \text{Spam}(m))$ seems harder...

Assume:

- Order of words in document doesn't matter
- Presence of individual words is independent given whether the message is spam or ham

Spamistic words

Given our assumptions: ✓ stat, not very true

- Order of words in document doesn't matter
- Presence of individual words is independent given whether the message is spam or ham

$$\Pr(W(m) \mid \text{Spam}(m)) = \prod_{w \in W(m)} \Pr(w \mid \text{Spam}(m))$$

And now, we can count examples in our training data:

$$\Pr(w \mid \text{Spam}(m)) = \frac{\text{num spam messages with } w}{\text{total num spam messages}}$$

$$\Pr(w \mid \text{Ham}(m)) = \frac{\text{num non-spam messages with } w}{\text{total num non-spam messages}}$$

Pr(spam given word) for an example message

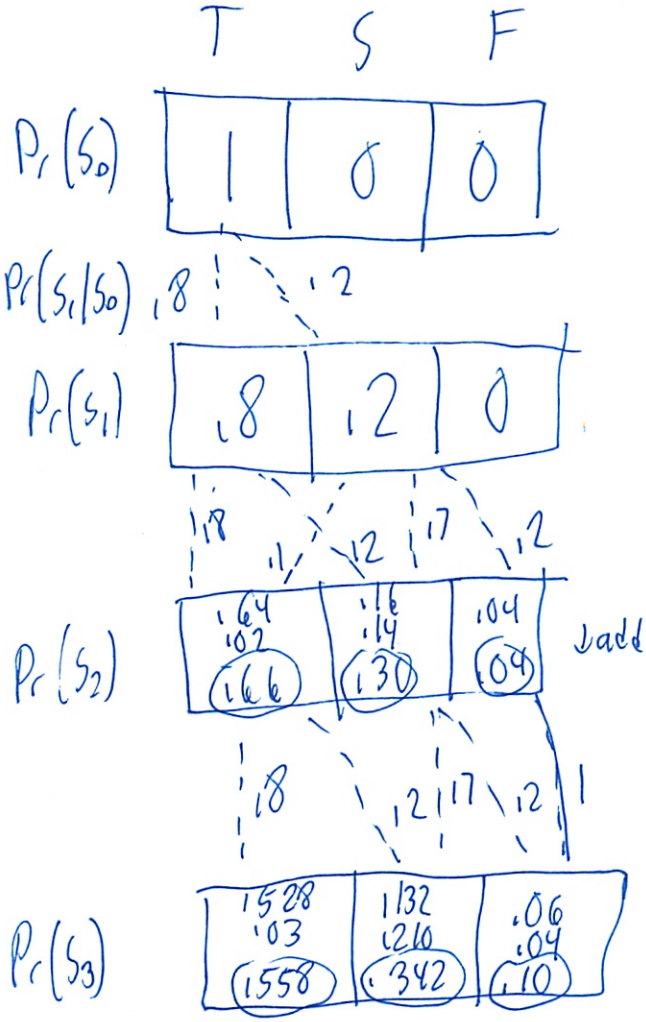
madam	0.99
promotion	0.99
republic	0.99
enter	0.9075001
quality	0.8921298
investment	0.8568143
valuable	0.82347786
very	0.14758544
organization	0.12454646
supported	0.09019077
people's	0.09019077
sorry	0.08221981
standardization	0.07347802
shortest	0.047225013
mandatory	0.047225013

This Week

Software lab: Transition and observation models

Design lab: Writing code to update state estimation models for your robot.

not building detailed model of spam
- just looking at probabilities



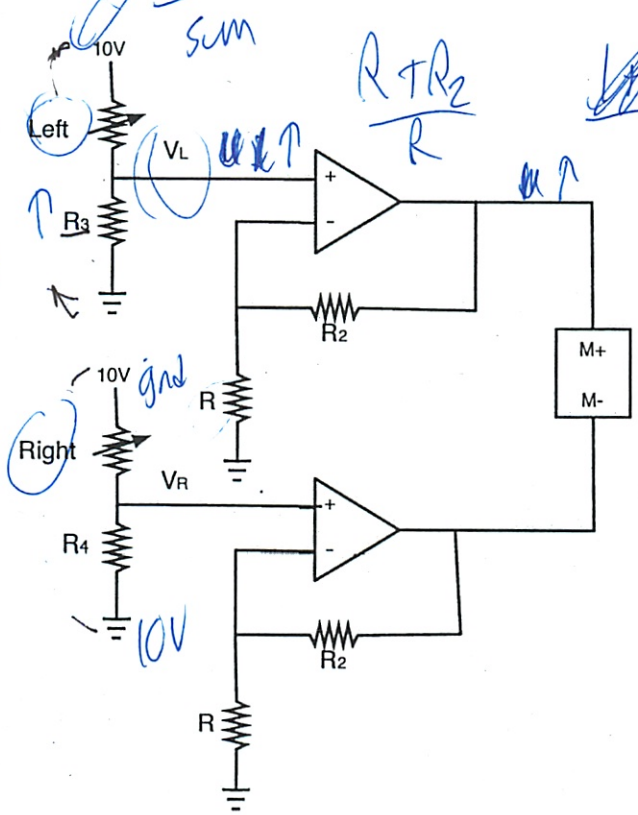
interesting to see the different ways

I still like my way

Interview 2

11/16

Reference design



bottom

sum

$\frac{R+R_2}{R}$

$\frac{1+9}{1} = \frac{10}{1}$

$\frac{1}{5}$ $\frac{4}{5}$

$10 - V_L$

gnd

10V

rough

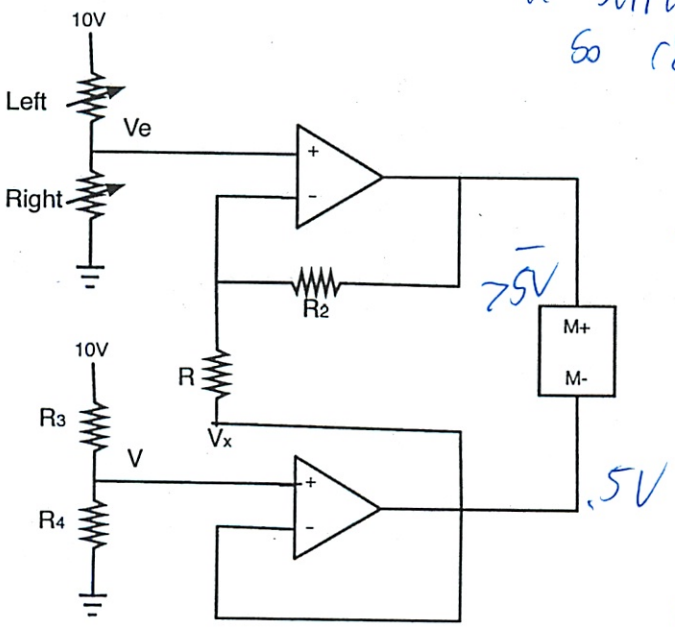
When $R3 = 2R4$

will settle

so right light brighter

Alternative design

adder



$>5V$

$+ \downarrow 10$

$<5V$

$5 = 5$ not move

.5V

will never saturate

Software Lab 11: Hallway World

1 Hallway World

We are going to work with an abstract simulation of a robot moving up and down a one-dimensional hallway made up of a fixed number of rooms.

- The robot starts out knowing how many rooms there are, but not knowing what room it is in.
- The robot knows a probability distribution over what observations it will make in each room.
- The robot can attempt to move in each direction in the hallway; it will not always move completely reliably. If it is up against the left end of the hallway and tries to move to the left, it will stay in the leftmost room; similarly for moving right when it is in the rightmost room.

The goal of the robot is to determine what room it is in. We will approach this problem using state estimation.

Step 1. Start idle with `-n`, and load the file `sl11Work.py` and run it. (We will not be using Soar. This is a stand-alone piece of software). Now, type

```
p = makePerfect()
p.run(10)
```

At this point, p is an instance of a little application that both simulates the robot moving in the world and shows the estimated belief state.

Each square corresponds to a room in the world. Each room has a "true" color, which determines the distribution over the observations that the robot will make in that room. The room's true color is shown in the outer rim of each square. The robot cannot "see" the room's true color; the robot's observations are some "corrupted" color (whose distribution generally depends on the true color).

The robot's *belief state* is a probability distribution over which room it is in. In the window, the current belief state is displayed in the colors of the inner squares in each block, with brighter red values closer to zero and brighter blue values closer to one. The color black is assigned to the probability associated with the uniform distribution (in this case 0.2). Note that these "colors" are just a way of encoding the probabilities visually; they have nothing to do with the room's actual or perceived color. The probability assigned to each room is also printed in each square.

In fact, p is a combination of a *stochastic state machine* that simulates the behavior of the robot-world system and a *state machine* that does state estimation, based on the robot's actions and observations, to compute a new belief state on each time step.

The way this complex machine works is the following:

- When p is initialized, at the beginning of each call to `p.run`:
  1. The state estimator *initializes the belief state* to the starting belief state which, in this case, is the uniform distribution over the rooms.
  2. The simulator selects an *initial starting location* for the robot at random from the starting distribution. Note that *the state estimator does not know this true location;* it is just used inside the simulation. It is also not displayed in any way in the window or the printed output.
- On every step of p:
  1. The simulator *generates an observation,* drawing from the distribution of observations that is associated with the room that the robot is currently really located in. This observation is a color name, like 'white' or 'green'.
  2. The simulator *prompts the user for an action* that the robot should take. The action must be an integer between -4 and 4, inclusive.
  3. The simulated *robot moves* a number of rooms that depends on the specified action; but if the robot's motion model is noisy (we will discuss what that means in detail, later), then it won't necessarily be the exact number of rooms commanded. In addition, it will not move past either end of the hallway. An action of 0 will cause the robot to try to stay in its current location.
  4. The state estimator does an *observation update* of its belief state, based on its old belief state and the observation. The update depends on the observation model, which specifies the probability distribution over observations for each state. This belief state is printed out.
  5. The state estimator does a *transition update* of its belief state, based on the belief state that resulted from the observation update and the specified action. The update depends on the transition model, which specifies the probability distribution over next states, given the previous state and action.
  6. The *squares are redrawn* with colors and numbers that reflect the probabilities in the new belief state.
- If you enter `quit` as an action, the whole machine terminates. It will stop after 10 steps unless you call `run` with a larger numeric argument.

  If you want to run the machine again, it's best to create a new instance; it's okay if you call `run` again on the old instance, but the initial belief display will be incorrect until it does one update.

Check Yourself 1. Move the robot around in the perfect simulator. Be sure you understand what the colors representing the belief state mean and that the numbers being printed out in the Python shell make sense. Feel free to ask a staff member for clarification.

Step 2. The world you just created has perfect motion and perfect sensing. You can create and run one with noisy motion and sensing as follows:

```
n = makeNoisy()
n.run(20)
```

Check Yourself 2. Move the robot around in the noisy simulator. Be sure you understand what the colors mean, and have a basic idea of what might be going on. Feel free to ask a staff member for clarification.

1.1 The observation model

Section 8.6 of the readings may help with understanding the next sections.

The observation model is a conditional probability distribution specifying a what color the robot sees given what room it is in: $P(O_t = o_t | S_t = s_t)$. In our case o_t ranges over

```
('black', 'white', 'red', 'green', 'blue', 'purple', 'orange', 'darkGreen',
'gold', 'chocolate', 'PapayaWhip', 'MidnightBlue', 'HotPink', 'chartreuse')
```

and s_t ranges over all the possible rooms the robot could be in.

If there are m possible observations and n possible locations, then it will, in general, require $m \cdot n$ numbers to specify the observation model. In this problem, we make an assumption that makes the model much more compact: the robot's observation only depends on the *actual color* of the room it's in. That is, all rooms that are actually white have the same distribution over possible observations and all rooms that are actually green have the same distribution over possible observations (which will generally be different from the observation distribution for rooms that are actually white). Given this assumption, we only need to specify $P(\mathit{observedColor} \mid \mathit{actualColor})$, and then we can find the probability of observing each color in any room, as long as we know the actual color of that room:

$$P(O_t = \mathit{observedColor} \mid S_t = s_t) = P(O_t = \mathit{observedColor} \mid \mathit{ActualColor} = \mathit{actualColor}(s_t)) \ .$$

The conditional probability distribution

$$P(O_t = \mathit{observedColor} \mid \mathit{ActualColor}) \ ,.$$

which specifies a distribution on the observed color given the actual color of the robot's room, is called the *observation noise distribution*. We can specify an observation noise distribution in Python as a procedure that takes an actual color as input and returns a distribution on observed colors. Here is a very simple example that always observes the true color.

```
def perfectObsNoiseModel(actualColor):
    return dist.DDist({actualColor: 1.0})
```

Now, given an observation noise distribution `obsNoise`, such as `perfectObsNoiseModel`, we can construct the entire observation model (a conditional probability distribution on observed colors given robot location) as shown below:

```
def makeObservationModel(hallwayColors, obsNoise):
    return lambda loc: obsNoise(hallwayColors[loc])
```

Here, `hallwayColors` is a list specifying the true color of each location in the hallway, `loc` is an integer representing the location of the robot, and `obsNoise` is a conditional distribution of observed color given actual color. This procedure returns a conditional probability distribution, which is a procedure that takes a location as input and returns a distribution over observed colors.

The example world we have been using is specified with

```
standardHallway = ['white', 'white', 'green', 'white', 'white']
```

Given these procedures, we can specify the observation model for perfect observations with:

```
perfectObsModel = makeObservationModel(standardHallway, perfectObsNoiseModel)
```

Step 3.

Wk.11.1.1 Do this tutor problem on defining observation models.

Check Yourself 3. Just to be sure you understand these models, consider a world with two rooms: room R_0 is actually green and room R_1 is actually white.

- With a perfect sensor, what is the probability distribution over observations for each room?

	green	white
Pr(obs | R0)	1	0

	green	white	
	0	1	Pr(obs | R1)

- With `whiteEqGreenObsDist`, what is the probability distribution over observations for each room?

	green	white
Pr(obs | R0)	.5	.5

	green	white	
	.5	.5	Pr(obs | R1)

- With `whiteVsGreenObsDist`, what is the probability distribution over observations for each room?

	green	white
Pr(obs | R0)	0	1

	green	white	
	1	0	Pr(obs | R1)

Step 4. Now consider a bigger test world, called `testHallway`. The rooms on each end have true color 'chocolate'. The others are either green or white.

Try out your `whiteEqGreenObsDist` and `whiteVsGreenObsDist` and see what happens to the belief state. Paste your definitions of those procedures from the tutor into your `sl11Work.py` file. Use the perfect motion models, as shown below. **Be sure that you use something like `whiteEqGreenObsDist` as the third argument below.** (The variables `actions`, `standardDynamics`, and `perfectTransNoiseModel` are already defined for you in `sl11Work.py`).

```
w = makeSim(testHallway, actions,
            <your observedColor given actualColor model>,
            standardDynamics, perfectTransNoiseModel)
w.run(50)
```

Checkoff 1. How do `whiteEqGreenObsDist` and `whiteVsGreenObsDist` compare to:

- A perfect sensor model
- A sensor that always reads 'black' no matter what room it is in

Explain your answers to a staff member.

1.2 The state-transition model

The state-transition model is a conditional probability distribution over the state at time $t+1$, given the state at time t and the selected action a. That is, $P(S_{t+1} = s_{t+1} | S_t = s_t, A_t = a_t)$. The next state of the system depends both on where it was before and the action that was taken. If you were to write this model out as a matrix, it would be very big: if n is the number of states of the world and m is the number of actions, then it would have size mn^2.

Often, the transition model can be described more sparsely or systematically. In this particular world, the robot can try to move some number of rooms to the right or left, or to stay in its current location. We'll assume that the kinds of errors the robot makes when it tries to move in a given direction don't depend on where the robot actually is (except if it is at the edge of the world), and we'll further assume that the transition probabilities to most states are zero (there's no chance of the robot teleporting to the other end of the hallway, for example). This will allow us to describe the transition model more compactly.

We will start on defining the transition model by first defining a *dynamics* procedure, such as `standardDynamics` below. The dynamics is a procedure which returns the nominal new location resulting from taking action `act` in location `loc`, in a hallway with `hallwayLength` locations. This will be useful in other problems. The possible actions that the robot can take are: $-4, \ldots, -1, 0, 1, \ldots, 4$. So, we can just add the robot's action to its current location to get its nominal new location, except we have to be sure it doesn't move off of the edges of the world, so we use `util.clip` to keep the value from going below 0 or above `hallwayLength - 1`.

```
def standardDynamics(loc, act, hallwayLength):
    return util.clip(loc + act, 0, hallwayLength-1)
```

Next, we define a *noise model*, which is independent of location. It returns a distribution over the possible resulting locations given the nominal location that results from an action under the given dynamics. The simplest noise model assumes that transitions are perfect, so that the resulting location will be the nominal location.

```
def perfectTransNoiseModel(nominalLoc, hallwayLength):
    return dist.DDist({nominalLoc : 1.0})
```

Ultimately, we need to make a full transition model, which is a conditional probability distribution of the form $\Pr(S_{t+1} \mid S_t, A_t)$. Because it is conditioned on two variables, we will represent

it using nested procedures (representing conditional distributions). So, we'll think of it as something like $\Pr(S_{t+1} \mid S_t \mid A_t)$, or, as a procedure that takes an a_t and returns a procedure that takes an s_t and returns a distribution over S_{t+1}.

Here is the basic form of the transition model that takes two procedures, a dynamics procedure and a noise model, and an integer indicating the length of the hallway.

```
def makeTransitionModel(dynamics, noiseDist, hallwayLength):
    return lambda act: lambda loc: noiseDist(dynamics(loc, act, hallwayLength),
                                             hallwayLength)
```

A perfect transition model for our standard hallway under the standard dynamics is constructed as follows:

```
perfectTransModel = makeTransitionModel(standardDynamics, perfectTransNoiseModel, 5)
```

Step 5.

Wk.11.1.3	Do this tutor problem on defining transition models.

Step 6. Now consider a test world that has only white rooms, and has noisy transitions and observations. We will initialize the state estimator with an initial belief state that assigns probability 1 to location 7 (and, of course, probability 0 to all other locations). The variable `sterile` specifies a hallway made up of 16 white rooms. You can create this world and state estimator with:

```
w = makeNoisyKnownInitLoc(7, sterile)
w.run(50)
```

Experiment with selecting action 0 several times in a row. What happens? What happens when you drive the robot around?

Checkoff 2.	Demonstrate the world with noisy dynamics to a staff member and explain why it does what it does.

Step 7.

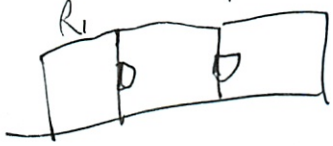
Wk.11.1.5-6	Do these two tutor problems on implementing joint distributions and operations on conditional distributions.

SWLab 11

11/16

1. Hallway World

- One-d hallway
- fixed # of rooms
- knows how many rooms but not which is in

TAs 

← basically a string of interconnecting rooms

- just trying to ~~de~~ descritize 1D space

- knows observation prob dist.
- will not always move reliabilitiy
- can't go more left + right than end of room chain
- sll work.py
  - Collored rooms "true" color
  - uses collored rooms lib
- ~~room~~ robot observes some "corrupted" color
- robot's belief state is probability dist of what rooms is in

inside (red → closer to 0
black → uniform
blue → closer to 1) have nothing to do w/ room percieved/actual

outside color – actual color of room

- p is ~~the~~ combo stochastic SM that simulates behavior of robot-world system, + SM that does state estimation based on robots actions + observation

(2)

- So first SM is like copy machine example
- Second is standard robot SM

P - initalized

1. State estimator initalizes belief state
2. Simulator select a start location at random
   - not shown at all

On every step of P

1. Simulator generates an observation at random - according to observation probability distribution
2. Simulator prompts for action (-4, 4)
3. ~~n~~ Robot moves that # of rooms
   - if motion noisy won't always move that # of rooms
   - remember can't leave hallway $-4 < x < 4$
4. State estimator does an observation update on belief state
   - based on old ~~observation~~ belief state + observation
   - depends on observation model
   - belief state printed
5. State estimator does a transition update of belief state
   - previous belief state (from observation) + action
6. Squares redrawn w/ probabilities

(3)

Can call quit

If quit recreate machine

(Y) → Try it out

- oh so blue = most likly position

  1 = exact position

  States # from start

  - not signed, like I was thinking

- this world has perfect sensing

- can also do noisy

  n = makeNoisy

  n.run(20)

---

note: the move probability includes going wrong direction

I was just thinking it was - if say go 4, it may go 3

but it may go -1

wrong direction!

1.1 Observation model

- section 8.6 will help

- $P(O_t = o \mid S_t = s)$

(4)

m possible observations
n possible locations
↳ so requires $m \cdot n$ # to specify the observation model

but we are assuming robots' observation on depends on actual <u>color</u> of room, so only specify P(observed | actual)
(That makes sense)

Can find prob of observing each color in any room if know actual color of room

$$P(O_t = \text{observed} \mid S_t = s_t) = P(O_t = \text{observed} \mid \text{Actual} = \text{actual}(s_t))$$

$P(O_t = \text{observed} \mid \text{Actual})$ = observation noise distribution

Perfect noise model

```
def perfectObsNoiseModel(actual):
    return dist.DDist({actual: 1.0})
```

↑ oh cool the variable name just in there

Now entire observation model

```
def makeObservationModel(hallwayColors, obsNoise):
    return lambda loc: obsNoise(hallwayColors[loc])
```

↑ list of true colors

(5)

Can then specify the observation model

perfectObsModel = makeObservationModel(standardHallway, perfectObsNoiseModel)

Do tutor ~~[illegible]~~ problem 11.1.1

- looking at defining observation models
- dist modual (look for help)
- makeObsModel + stdHallway + possibleColors defined
- Part 1 White==Green
  - robot color blind b/w white + green
  - so want to fill in white Eq Green Obs Dist
  - So this is what makeObs calls
  - So if actual != white, green
    → actualColor 1.0

    actual = white or green
    white .5, green .5
  - error - need quotes around 'white' 'green'

(X)

- (✓)

(6)

Part 2 White ↔ Green

- white always looks green
- green " " white
- all other ~~colors~~ colors ~~at~~ ~~are~~ perfectly observed

```
if actual == green:
    ddist (white : 1)
elif " == white
    ddist (green : 1)
elif
    ddist (actual : 1)
```

(✓)

Part 3 Noisy

.8 pr of observing actual color
.2 of any other color
- evenly distributed among other possible colors

? can I hard code possible colors
- will assume yes

(X) typo
(✓) fixed

(7)

Part 4: Observation Model

- so far have encoded error model ind. of location
- but we should consider location
- want full observation model for st. hallway
- w/ noisy Obs model
- So

  - noisyObsModel(hallway Colors, obsNoise):
    return lambda loc: obsNoise(hallway Colors[loc])
  - w/ proper variables

(X) only 1 arg given

- takes current state

not supposed to do that here (actually defined in problem)

Prof: no diff b/w noise obs + noise model

this is the algorthmic/abstract way to generate model
I hard coded it in part 3

noisyObsModel = makeObservationModel(St. Hallway, noisyObs)

CY3 - filled in

(8)

New bigger test world test Hallway

chocolate _____ Chocolate

green or white

- paste into SI Work py

Checkoff

TA: Actually it knows perfectly when green/white reversed because that is model does

- the prof model is <u>not</u> the model of camera
- but it works on the <u>output</u> of the camera and flips it for our purposes ✓

---

lab over

office hrs 11/17

1.2 State transition model

- Conditional prob distribution over state at time $t+1$ given state at time t and selected action a

$$P(S_{t+1} = s_{t+1} \mid S_t = s_t, A_t = a_t)$$

where is currently

action

9

If tried to write out as matrix $(\#actions)(\#states)^2$

- but we are simplifying
  - robots error does not depend on current pos
  - except the end of the world
  - transition prob to ~~both~~ most stations $= 0$

Need to define a dynamics procedure

- proc that returns new location
- paramaters : act, loc
- use clip to prevent going out of world

Noise model

- ind. of current location
- "perfect" = no noise

```
def perfectTransNoiseModel(nominalLoc, hallwayLength):
    return dist.DDist({nominalLoc: 1.0})
```

- will be building a more complex one

(10)

Combine it all into a transition model

```
def makeTransitionModel(dynamics, noiseDist, hallwayLength):
    return lambda act, lambda loc: noiseDist(dynamics(
        loc, act, hallwayLength), hallwayLength)
```

```
PerfectTransModel = makeTransitionModel(stdDynamics, perfect
    TransNoiseModel, 5)
```

Now Tutor problem 11.1.3

- defining transition models
- use dist model again

```
- def incrDictEntry(d, k, v):
    if d.has_key(k):
        d[k] += v
    else
        d[k] = v
```

Part 1: Living on a donut

↳ a circular world

room 0 connected to hallwayLenght -1

(11)

So what is a dynamic

- ok → the std. one is the clip
- clip (v, vMin, VMax)
- So we want to do!

return loc + act

if loc + act > hallway Lenght − 1

loc + act − (hallway Lenght − 1)

if loc + act < 0

hallwayLenght − 1 − (loc + act)

but that is only 1 x around ring

need to loop till have a valid position

~~While~~ ← like other order

loc + act = target

while 0 > target:

target = ~~target~~ hallway Lenght − 1 − target

while target > hallway lenght − 1

target = target − (hallway lengh − 1)

return target

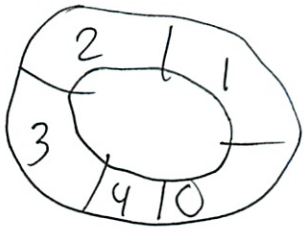
(12)

(X)

- was close actually

when it fit 1 time arand it worked

when it was 5 on a hallway lenght 5



returns 1
should be 0

5 - _ = 1

now doing 5-1
should do 5

So target = ~~hallway lenght~~ ~~-1~~
target - hallway lenght
↑ no -1

New works except 1

0, -3, 5

should be 2

- but my script going to 8

Oh sign error

Oh do target - hallway lenght for both

(13)

No!!! ∞ loop

$-3 + 5 = 2$

target + hallway lenght

(✓) works now

Part 2: Slipping to the left

~~With~~ left slip noise model

w/ p .1 will go 1 extra spot left

.9 lands right

w/o going out of world

↑ need to check!

```
def leftSlipTrans(nominalLoc, hallwayLenght)
```

↑ loc it should be

what are we returning? ~~location, assuming~~ prob

```
if nominalLoc == 0:
    return DDist{nominalLoc: 1.0}
else
    return DDist{nominalLoc: .9, (nominalLoc-1): .1}
```

(✓)

(14)

Part 3 Noisy Transitions

one to the left .1
one to the right .1
nominal sq .8
Can't fall off world

~~if nominalLoc == 0 or nominalLoc == (hallwaylenght-1)~~
no can still fall one to the right
(remember my issue w/ -4)
-prob does not depend on location
- " " " how much you move

```
if nominalLoc == 0:
    return DDist {nominalLoc: .9, nominalLoc+1: .1}
if nominalLoc == (hallway Lenght -1):
    return DDist {nominalLoc: .9, nominalLoc-1: .1}
else
    return DDist {nominalLoc: .8, nominalLoc+1: .1,
                  nominalLoc-1: .1}
```

They did a much nicer one w/ clip

0

(15)

Part 4 Transition Models

noisyTransModel =

↑ put it all together (like before + like perfectTransModel in handout)

(✓) straight off the handout

Back to handout Step 6

- test world
  - white rooms
  - noisy transitions + observations
  - start in location 7

W = makeNoisyKnownInitLoc(7, sterile)

↑ start ↑ 16 white rooms

Select 0 a bunch of times

- each time becomes more uncertain where it is

Checkoff 2 (✓)

(16)

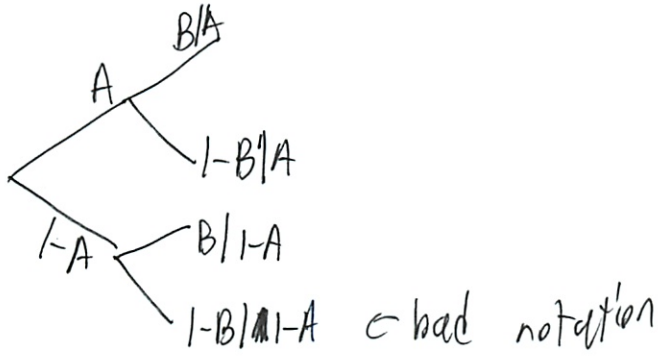
Tutor 11.1.5 Implementing Joint Distributions

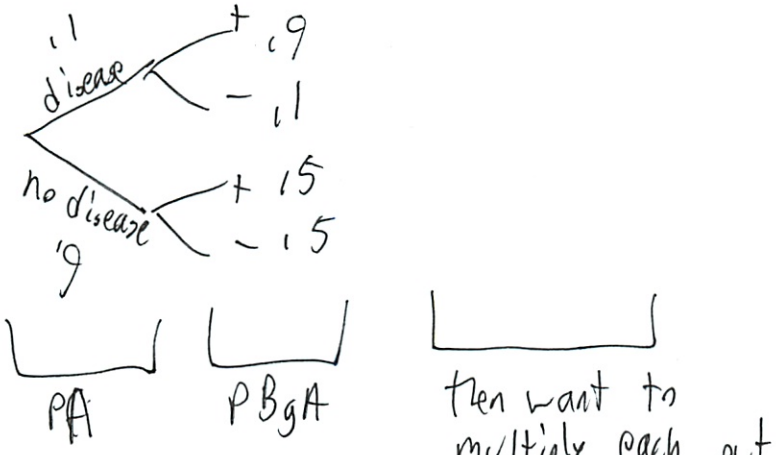
JDist function

have PA P(A)

PBgA P(B|A)

So iterate through each value





So for each thing in A

- ? how w/ DDist disclose
  - support ok

then for each sub of that

- but its weird → just returns Dist

(17)

Need to use that somehow

- again how to iterate over
- Oh is a d dist
- Nice

~~no~~ now need to multiply out an save

how append to DDist

- just do a normal dict for now

list indexes can't be tuples

(✓) got it working - and really elegant I think

<u>Part 2</u> <u>Marginalization</u>

marginalizeOut(self, index) to DDist class

1 or 0

2nd 1st variable

remove Elt already defined

and a Dict element entry

So if index = 0 - which is disease, no disease

then add up ....

(18)

	Dis	no Dis	
+	.09	.45	.54
−	.01	.45	.46

So if I add up all the +/− tests
? but this is dis/no Dis ??
I add up the dise, no dis

get an item
iterate through
Sort
Oh – remember need to remake item each time!
don't need to iterate over disease/no disease!
Cool working in python
(X) errors in tutor
(✓) works now – this is fun
– really challenging!

(19)

Part 3 Conditioning

conditionOnVar (self, index, value)

0 or 1 again — variable assumed to be there

- so lets do this w/ previous ans

$$P(B \mid A=a) = \frac{P(A=a \cap B)}{P(A=a)}$$ ← for each B

↑ value is specified

~~or $P(B|A) = \frac{P(A \cap B)}{P(B)}$ ← marginalize~~

for each value of cause

	Des	No Des	
+	.09	.45	.54
−	.01	.45	.46
	.1	.9	

example $P(1 = posTest) = \frac{P(posTest \cap Disease)}{P(PosTest)}$ .54

ah I kinda got 1,0 now

(20)

We need to do it for the flip(index)

take x[flip(index)]

^ie disease

find (postest, disease)

Oh that is just x in every case!

self.d[(x[flip(index)], x[index])] = ~~x.p.~~ self.prob(x)

Opps!

that 0,1 index thing is weird

- but what would you call it?

why does this just run once?

- oh its the marginalize out running!

- duh

Oh when / by marginalize out need proper one

- and a flip of te index to meet their def

Nice working

✓ cool

that took a long time - but was fun

theirs was nicer of course

(21)

actually my code may be nicer
- I think so

11.1.6 Implementing Operations on Conditional Dist.

Part 1 Implement Bayes Evidence

bayesEvidence(PBgA, PA, b)
can write in 1 line

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

but b given

$$P(A|B=b) = \frac{P(B|A)\,P(A)}{P(B=b)}$$

- so need to iterate over as ;)
- note - not part of DDist class
- so PBgA is a function

PBgA(b) = DDist(~~posTest~~ disease, ~~negTest~~ noDisease)

b ↑ posTest

P(A) = DDist (disease, noDisease)

(22)

↑ Can get that from marginalizing?
— well its given

$P(B) = DDist\ (posTest, negTest)$
↑ given

so $P(b)$. prob(value)

but need to do two seperate
since can't • DDists, right?

$P(B)$ never given though??

Ok multiplied together
— now need to divide by $P(B)$

wait isn't $P(B)$ the sum of the two
— yeah! duh

$P(B|A)\ P(A) = P(B)$

— I just made them!

— is it sum, or the p of chosen action
— sum, since must $= 1$

Nice working
(why do I always forget the prob things — need to really get)

(23)

And so much for only 1 line!

(x) errors in tutor

had DDiet not DDist

(x) still wrong

- runs fine in Python
- oh I define posTest in PBgA
- its not defined in there
- ~~so wtf they want~~
- but they are calling PTgD
  test — but they have given postest
- so I implemented it wrong

  its not PBgA(b)

  its PBgA(disease) = {pos test, neg test}

  PTgD

  P(A) = P(Disease) = {disease, no disease}

  P(B = postest) ← need

  but where get this?

want $P(\text{Disease} \mid \text{test} = +) = \dfrac{P(\text{test} \mid \text{disease})\, P(\text{disease})}{P(\text{test} = +)}$

{dis, no dis} {+, −} (dis, no dis)

(24)

So $\dfrac{P(\text{test} = + \mid \text{disease}) \cdot P(\text{disease})}{P(\text{test} = +)}$

.9 .1

.54 ← can find w/ marginalization

= .1667 ✓ works out

but got .54 from marginalizing before

So first need what are the P(A)s

PA.support()

Then take that for each ~~all~~ .prob(b) ✓

~~Get that~~

Now ∘ the p of disease / no disease ✓

~~Go~~

Now the marginalize part

- but don't have PB
- oh now I can add ~~.9·.1 + .1·.9~~ .5·.9 + .9·.1

~~.45~~ .45 + .09

Sum + Normalize

(25)

Oh wow the only thing I actually changed was

Old → PBgA (b), prob(x) · PA, prob(x) and x in PBgA

new → PBgA (x), prob(b) · PA, prob(x) x in PA

✓ Works now

So what did I do wrong 1st time?

Ans the same when ~~[illegible]~~ just transfered the ~~[illegible]~~ conditional the other away around (was that even legal?)

↑ no abs not!

Think it was iterating over right thing

- getting confused w/ letters
- had to write it out

Part 2 Total Probability

total Probability (PBgA, PA)

So tP (PTgD, disease) = {pos test, neg test}

- is this just a subset of before

P(T | D) P(dis)

{+, -} {dis, nodis}

(26)

Yeah just subset I think

- no - since B=b not given

So sum the pos test under both dis and no dise

- So kinda like marginalized

<u>and</u> need to normalize

Yeah but my values are wrong

ahh - its not just simple add

I have $P(\text{test}_{\pm} \mid \text{disease}) + P(\text{test}_{\pm} \mid \text{no disease})$

Need $P(\text{test}_{\pm} \mid \text{disease})\, P(\text{disease})$

Works

(I must have built the same thing several times - just so confused

(J) Done

Done

Lab