6.01: Introduction to EECS I

Bayesian estimation, etc.

Week 10 *November 10, 2009*

What about the bet?

- Total number of legos in the bag $= 4$
- Random variable R: number of red legos in the bag.
- Domain $D_R = ?$
- 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:*

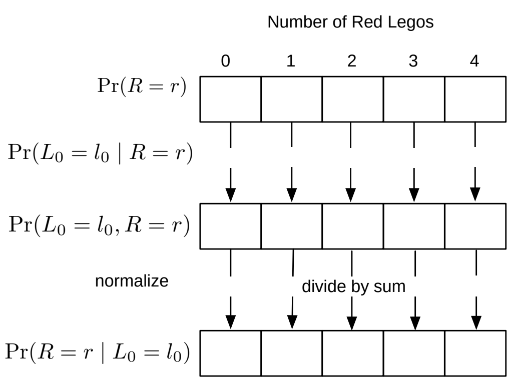
$$\Pr(L_0 = \text{red} \mid R = n) = ?$$
$$\Pr(L_0 = \text{white} \mid R = n) = 1 - \Pr(L_0 = \text{red} \mid R = n)$$

- We want to know:

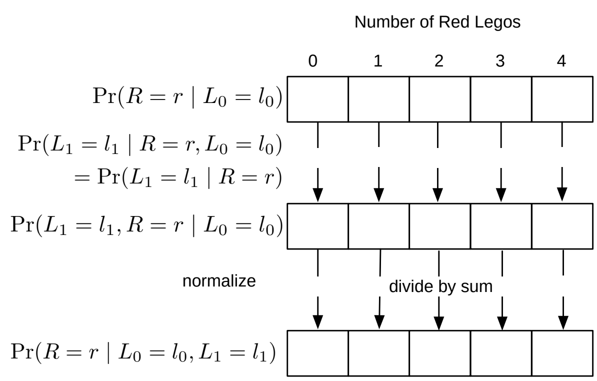
$$\Pr(R = n \mid L_0 = \text{whatever color we observed})$$

Bayes!!

What do we know after first Lego draw?



What do we know after second Lego draw?



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

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

Copy Machine

Initial state distribution

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

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

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

A perfect copy!

Step 1: Bayes Evidence:

Build joint distribution:

$\Pr(S_0, O_0) = \Pr(O_0 \mid S_0) \Pr(S_0)$

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

Condition on actual observation $O_0 = \textit{perfect}$

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

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

A perfect copy!

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

Time passes

Step 2: Total Probability:

Build joint distribution:

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

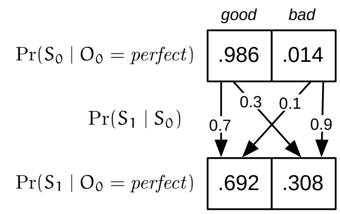
		S_1 good	S_1 bad
S_0	*good*	0.690	0.296
	bad	0.001	0.012

Marginalize out S_0:

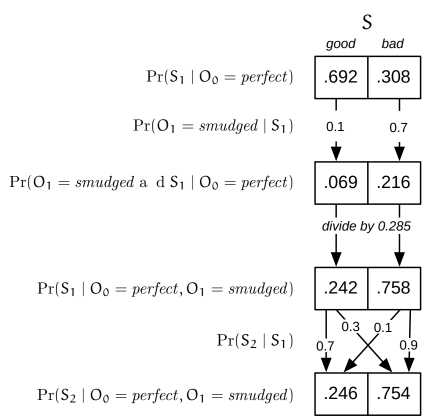
$$\Pr(S_1 \mid O_0 = \textit{perfect}) = \sum_s \Pr(S_0 = s, S_1 \mid O_0)$$

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

Time passes



A smudged copy, another day



Stochastic State Machine

There are no actions in a Hidden Markov Model.

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
```

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())
```

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']
```

Python State Estimation

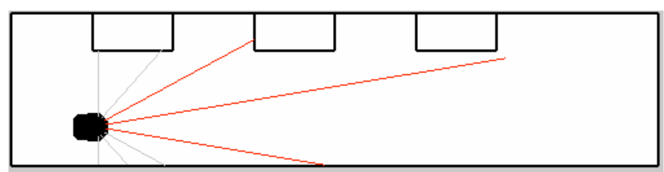
```
class StateEstimator(sm.SM):
    def __init__(self, model):
        self.model = model
        self.startState = model.startDistribution
        self.obsD = self.model.observationDistribution
        self.transD = self.model.transitionDistribution

    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)
```

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*.

One-dimensional localizer

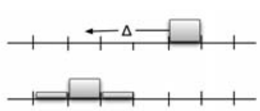


- **State space:** discretized values of x coordinate along hallway
- **Input space:** discretized relative motions in x
- **Output space:** discretize readings from a side sonar sensor

Transition model

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

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



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

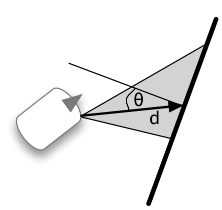
  $$\Delta = \mathsf{round}((x_t - x_{t-1})/w)$$

  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

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

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

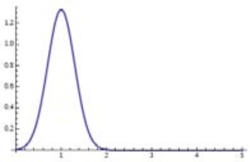


- Assume nominal distance is all that matters; observation distribution is a triangular distribution, with half-width hw_d, centered on the nominal distance.

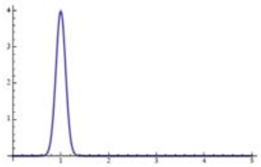
Continuous error distribution

Gaussian:

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



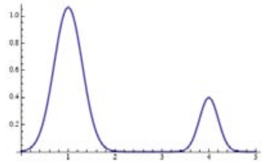
$\sigma = 0.3$



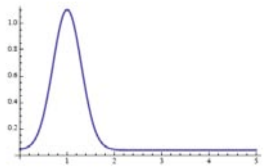
$\sigma = 0.1$

Mixture distributions

Mixture of Gaussians:

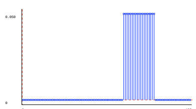


Mixture of uniform and Gaussian:

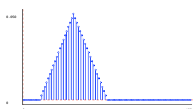


Discrete error distributions

Primitive distributions:



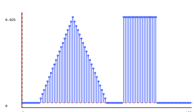
Sq(x; lo, hi)

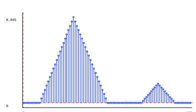


Tri(x; center, hw, lo, hi)

Mixture distributions:

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



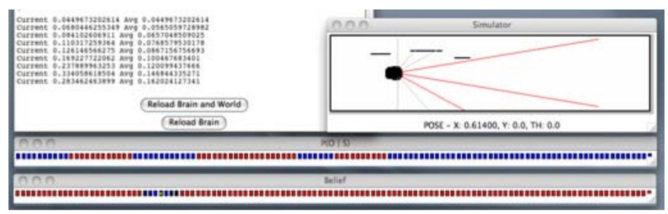


Must still sum to 1.

Localization = State Estimation

Define Stochastic State Machine

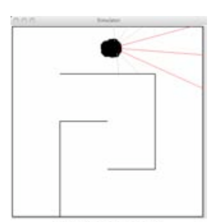
Apply standard state estimation algorithm.



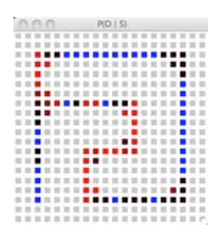
Extending to multiple dimensions

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

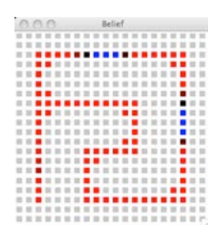
Full localizer



Simulator

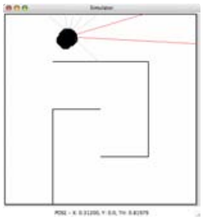


$\Pr(O|S)$

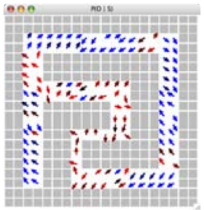


$\Pr(S)$

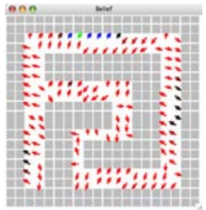
Full localizer: with angles



Simulator



$\Pr(O|S)$



$\Pr(S)$

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 is 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

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

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!

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.

$\Pr(W(m) \mid \text{Spam}(m))$ 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, Assume:

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

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

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

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

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

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