CSCI 5582
Artificial Intelligence

Lecture 17

Jim Martin

Today 10/31

- HMM Training (EM)
- Break
- Machine Learning

Urns and Balls

- Π Urn 1: 0.9; Urn 2: 0.1
- A

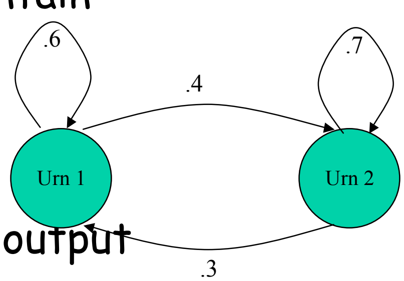
	Urn 1	Urn 2
Urn 1	0.6	0.4
Urn 2	0.3	0.7

- B

	Urn 1	Urn 2
Red	0.7	0.4
Blue	0.3	0.6

Urns and Balls

- Let's assume the input (observables) is Blue Blue Red (BBR)
- Since both urns contain red and blue balls any path through this machine could produce this output



Urns and Balls

Blue Blue Red

1 1 1	(0.9*0.3)*(0.6*0.3)*(0.6*0.7)=0.0204
1 1 2	(0.9*0.3)*(0.6*0.3)*(0.4*0.4)=0.0077
1 2 1	(0.9*0.3)*(0.4*0.6)*(0.3*0.7)=0.0136
1 2 2	(0.9*0.3)*(0.4*0.6)*(0.7*0.4)=0.0181

2 1 1	(0.1*0.6)*(0.3*0.7)*(0.6*0.7)=0.0052
2 1 2	(0.1*0.6)*(0.3*0.7)*(0.4*0.4)=0.0020
2 2 1	(0.1*0.6)*(0.7*0.6)*(0.3*0.7)=0.0052
2 2 2	(0.1*0.6)*(0.7*0.6)*(0.7*0.4)=0.0070

Urns and Balls

- Baum-Welch Re-estimation (EM for HMMs)
  - What if I told you I lied about the numbers in the model (π,A,B).
  - Can I get better numbers just from the input sequence?

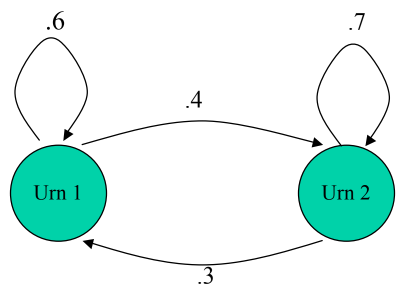
Urns and Balls

- Yup
  - Just count up and prorate the number of times a given transition was traversed while processing the inputs.
  - Use that number to re-estimate the transition probability

Urns and Balls

- But… we don't know the path the input took, we're only guessing
  - So prorate the counts from all the possible paths based on the path probabilities the model gives you
- But you said the numbers were wrong
  - Doesn't matter; use the original numbers then replace the old ones with the new ones.

Urn Example



Let's re-estimate the Urn1->Urn2 transition and the Urn1->Urn1 transition (using Blue Blue Red as training data).

Urns and Balls

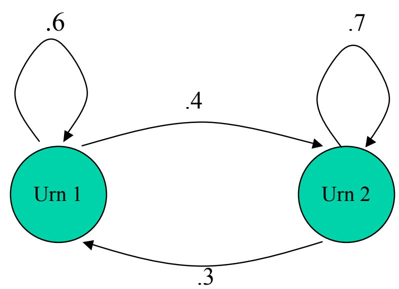
Blue Blue Red

1 1 1	(0.9*0.3)*(0.6*0.3)*(0.6*0.7)=0.0204
1 1 2	(0.9*0.3)*(0.6*0.3)*(0.4*0.4)=0.0077
1 2 1	(0.9*0.3)*(0.4*0.6)*(0.3*0.7)=0.0136
1 2 2	(0.9*0.3)*(0.4*0.6)*(0.7*0.4)=0.0181
2 1 1	(0.1*0.6)*(0.3*0.7)*(0.6*0.7)=0.0052
2 1 2	(0.1*0.6)*(0.3*0.7)*(0.4*0.4)=0.0020
2 2 1	(0.1*0.6)*(0.7*0.6)*(0.3*0.7)=0.0052
2 2 2	(0.1*0.6)*(0.7*0.6)*(0.7*0.4)=0.0070

Urns and Balls

- That's
  - (.0077*1)+(.0136*1)+(.0181*1)+(.0020*1)
  - = .0414
- Of course, that's not a probability, it needs to be divided by the probability of leaving Urn 1 total.
- There's only one other way out of Urn 1… go from Urn 1 to Urn 1

Urn Example



Let's re-estimate the Urn1->Urn1 transition

Urns and Balls

Blue Blue Red

1 1 1	(0.9*0.3)*(0.6*0.3)*(0.6*0.7)=0.0204
1 1 2	(0.9*0.3)*(0.6*0.3)*(0.4*0.4)=0.0077
1 2 1	(0.9*0.3)*(0.4*0.6)*(0.3*0.7)=0.0136
1 2 2	(0.9*0.3)*(0.4*0.6)*(0.7*0.4)=0.0181

2 1 1	(0.1*0.6)*(0.3*0.7)*(0.6*0.7)=0.0052
2 1 2	(0.1*0.6)*(0.3*0.7)*(0.4*0.4)=0.0020
2 2 1	(0.1*0.6)*(0.7*0.6)*(0.3*0.7)=0.0052
2 2 2	(0.1*0.6)*(0.7*0.6)*(0.7*0.4)=0.0070

Urns and Balls

- That's just
  - (2*.0204)+(1*.0077)+(1*.0052) = .0537
- Again not what we need but we're closer… we just need to normalize using those two numbers.

Urns and Balls

- The 1->2 transition probability is .0414/(.0414+.0537) = 0.435
- The 1->1 transition probability is .0537/(.0414+.0537) = 0.565
- So in re-estimation the 1->2 transition went from .4 to .435 and the 1->1 transition went from .6 to .565

Urns and Balls

- As with Problems 1 and 2, you wouldn't actually compute it this way. The Forward-Backward algorithm re-estimates these numbers in the same dynamic programming way that Viterbi and Forward do.

Speech

- And… in speech recognition applications you don't actually guess randomly and then train.
- You get initial numbers from real data: bigrams from a corpus, and phonetic outputs from a dictionary, etc.
- Training involves a couple of iterations of Baum-Welch to tune those numbers.

Break

- Start reading Chapter 18 for next time (Learning)
- Quiz 2
  - I'll go over it as soon as the CAETE students get in done
- Quiz 3
  - We're behind schedule. So quiz 3 will be delayed. I'll update the schedule soon.

Where we are

- Agents can
  - Search
  - Represent stuff
  - Reason logically
  - Reason probabilistically
- Left to do
  - Learn
  - Communicate

Connections

- As we'll see there's a strong connection between
  - Search
  - Representation
  - Uncertainty
- You should view the ML discussion as a natural extension of these previous topics

Connections

- More specifically
  - The representation you choose defines the space you search
  - How you search the space and how much of the space you search introduces uncertainty
  - That uncertainty is captured with probabilities

Kinds of Learning

- Supervised
- Semi-Supervised
- Unsupervised

What's to Be Learned?

- Lots of stuff
  - Search heuristics
  - Game evaluation functions
  - Probability tables
  - Declarative knowledge (logic sentences)
  - Classifiers
  - Category structures
  - Grammars

Supervised Learning: Induction

- General case:
  - Given a set of pairs (x, f(x)) discover the function f.
- Classifier case:
  - Given a set of pairs (x, y) where y is a label, discover a function that correctly assigns the correct labels to the x.

Supervised Learning: Induction

- Simpler Classifier Case:
  - Given a set of pairs (x, y) where x is an object and y is either a + if x is the right kind of thing or a – if it isn't. Discover a function that assigns the labels correctly.

Error Analysis: Simple Case

Correct

Chosen	+	-
+	Correct	False Positive
-	False Negative	Correct

Learning as Search

- Everything is search…
  - A hypothesis is a guess at a function that can be used to account for the inputs.
  - A hypothesis space is the space of all possible candidate hypotheses.
  - Learning is a search through the hypothesis space for a good hypothesis.

Hypothesis Space

- The hypothesis space is defined by the representation used to capture the function that you are trying to learn.
- The size of this space is the key to the whole enterprise.

Kinds of Classifiers

- Tables
- Nearest neighbors
- Probabilistic methods
- Decision trees
- Decision lists
- Neural networks
- Genetic algorithms
- Kernel methods

What Are These Objects

- By object, we mean a logical representation.
  - Normally, simpler representations are used that consist of fixed lists of feature-value pairs
  - This assumption places a severe restriction on the kind of stuff that can be learned
- A set of such objects paired with answers, constitutes a training set.

The Simple Approach

- Take the training data, put it in a table along with the right answers.
- When you see one of them again retrieve the answer.

Neighbor-Based Approaches

- Build the table, as in the table-based approach.
- Provide a distance metric that allows you compute the distance between any pair of objects.
- When you encounter something not seen before, return as an answer the label on the nearest neighbor.

Naïve-Bayes Approach

- Argmax P(Label | Object)
- P(Label | Object) =

$$\frac{P(\text{Object} \mid \text{Label})*P(\text{Label})}{P(\text{Object})}$$

- Where Object is a feature vector.

Naïve Bayes

- Ignore the denominator because of the argmax.
- P(Label) is just the prior for each class. I.e.. The proportion of each class in the training set
- P(Object|Label) = ???
  - The number of times this object was seen in the training data with this label divided by the number of things with that label.

Nope

- Too sparse, you probably won't see enough examples to get numbers that work.
- Answer
  - Assume the parts of the object are independent given the label, so P(Object|Label) becomes

$$\prod P(Feature = Value \mid Label)$$

Naïve Bayes

- So the final equation is to argmax over all labels

$$P(label)\prod_i P(F_i = Value \mid label)$$

Training Data

#	F1 (In/Out)	F2 (Meat/Veg)	F3 (Red/Green /Blue)	Label
1	In	Veg	Red	Yes
2	Out	Meat	Green	Yes
3	In	Veg	Red	Yes
4	In	Meat	Red	Yes
5	In	Veg	Red	Yes
6	Out	Meat	Green	Yes
7	Out	Meat	Red	No
8	Out	Veg	Green	No

Example

- P(Yes) = ¾, P(No)=1/4

- P(F1=In|Yes)= 4/6
- P(F1=Out|Yes)=2/6
- P(F2=Meat|Yes)=3/6
- P(F2=Veg|Yes)=3/6
- P(F3=Red|Yes)=4/6
- P(F3=Green|Yes)=2/6

- P(F1=In|No)= 0
- P(F1=Out|No)=1
- P(F2=Meat|No)=1/2
- P(F2=Veg|No)=1/2
- P(F3=Red|No)=1/2
- P(F3=Green|No)=1/2

Example

- In, Meat, Green
  - First note that you've never seen this before
  - So you can't use stats on In, Meat, Green since you'll get a zero for both yes and no.

Example: In, Meat, Green

- P(Yes|In, Meat,Green)=
  P(In|Yes)P(Meat|Yes)P(Green|Yes)P(Yes)

- P(No|In, Meat, Green)=
  P(In|No)P(Meat|No)P(Green|No)P(No)

Remember we're dumping the denominator since it can't matter

Naïve Bayes

- This technique is always worth trying first.
  - Its easy
  - Sometimes it works well enough
  - When it doesn't, it gives you a baseline to compare more complex methods to

Naïve Bayes

- This equation should ring some bells…

$$P(label)\prod_i P(F_i = Value \mid label)$$