

Python Course: Lecture 14

February 6, 2006

1 Smoothing

- Say you built a birdwatching model from the following observations.

Bird	Count	Probability
robin	2	0.2
sparrow	5	0.5
bluejay	3	0.3
Total	10	1.0

- What is the probability of seeing an emu?

$$p(\text{emu}) = \frac{C(\text{emu})}{10} = \frac{0}{10} = 0 \quad (1)$$

- This model says it's impossible to see an emu. But suppose there's an Australian circus in town, and their emu wrangler is a drunk, and one night he forgets to lock the cages, and ...
- A better model would say that $p(\text{emu})$ is small but non-zero.
- Likewise for probabilities of seeing a bald eagle, a California condor, a roc etc.
- Add an "unseen" category to our table of observed birds. Emus, eagles, rocs, etc. will all be treated as members of this category. By definition the unseen category has a count of zero.

Bird	Count	Probability
robin	2	0.2
sparrow	5	0.5
bluejay	3	0.3
unseen	0	0.0
Total	10	1.0

- So how do we get a non-zero probability for members of the unseen category? One way would be to lie and for the purposes of calculating the probability pretend we saw at least one of the bird in question. For example

$$p(\text{emu}) = \frac{C(\text{emu}) + 1}{10 + 1} = \frac{1}{11} = 0.09 \quad (2)$$

Here we add one to the denominator so that the normalized probability over all the bird types including the unseen type still comes out to 1.0.

- The denominator will have to go to 11 for all the birds. For example

$$p(\text{robin}) = \frac{C(\text{robin}) + 1}{10 + 1} = \frac{3}{11} = 0.27 \quad (3)$$

- The net effect is that probability of the unseen birds goes up while the probability of the seen birds goes down by the same amount.
- We say that probability mass has been redistributed among the different bird types.
- This general process is called *smoothing*. There are any number of mathematically sophisticated smoothing techniques, but in this class we will focus on a couple of basic ones to give you an idea of how it works.
- In practice, we rarely just add one count to a single unseen type, but we often do something similar called Laplace smoothing, described below.

2 Maximum Likelihood, Laplace, and Lidstone

Here are the mathematical definitions of a couple basic smoothing techniques.

2.1 Definitions

- Let X be a set of observed types, including the unseen type. For example, in the birds case

$$X = \{\text{robin}, \text{sparrow}, \text{bluejay}, \text{unseen}\} \quad (4)$$

- Let $C(x)$ be the number of instances of $x \in X$ observed in the training data. By definition $C(\text{unseen}) = 0$.
- Let N be the total number of training instances.

$$N = \sum_{x \in X} C(x) \quad (5)$$

- Let V be the number of distinct categories we bin those instances into (aka the number of types, aka the vocabulary size).

$$V = |X| \quad (6)$$

2.2 Maximum Likelihood Distribution

- The unsmoothed probability estimate is called the *maximum likelihood* distribution.

$$p_{MLE}(x) = \frac{C(x)}{N} \quad (7)$$

It is called “maximum likelihood” because it is the distribution that assigns the highest probability to the training data.

2.3 Laplace Smoothing

- In *Laplace* smoothing you add 1 to each of the counts and renormalize.

$$p_{LAP}(x) = \frac{C(x) + 1}{N + V} \quad (8)$$

- Laplace distributions are properly normalized

$$\sum_{x \in X} p_{LAP}(x) = \sum_{x \in X} \frac{C(x) + 1}{N + V} \quad (9)$$

$$= \frac{1}{N + V} \sum_{x \in X} (C(x) + 1) \quad (10)$$

$$= \frac{1}{N + V} \left(\sum_{x \in X} C(x) + \sum_{x \in X} 1 \right) \quad (11)$$

$$= \frac{1}{N + V} (N + V) \quad (12)$$

$$= 1 \quad (13)$$

Step (12) follows from the definitions in (5) and (6).

2.4 Lidstone Smoothing

- *Lidstone* smoothing is a generalization of Laplace smoothing where you add δ to each of the counts and renormalize.

$$p_{LID}(x; \delta) = \frac{C(x) + \delta}{N + \delta V} \quad (14)$$

Typically δ is a small number between 0 and 1.¹

¹By convention in probability functions, distribution parameters like δ are separated from the basic arguments like x with a semicolon instead of a comma. This is just a typographical convention and has no mathematical significance. $p_{LID}(x; \delta)$ and $p_{LID}(x, \delta)$ are different ways of writing the same thing.

- Lidstone distributions are properly normalized

$$\sum_{x \in X} p_{LID}(x; \delta) = \sum_{x \in X} \frac{C(x) + \delta}{N + \delta V} \quad (15)$$

$$= \frac{1}{N + \delta V} \sum_{x \in X} (C(x) + \delta) \quad (16)$$

$$= \frac{1}{N + \delta V} \left(\sum_{x \in X} C(x) + \sum_{x \in X} \delta \right) \quad (17)$$

$$= \frac{1}{N + \delta V} (N + \delta V) \quad (18)$$

$$= 1 \quad (19)$$

Step (18) follows from the definitions in (5) and (6).

- Laplace is a special case of Lidstone where $\delta = 1$.

3 Smoothing Example with Birds

- Here are probability values for a few different distributions using the birdwatching data from above. The Lidstone distribution uses the smoothing parameter $\delta = 0.01$.

Bird	Count	MLE	LAP	LID _{0.01}
robin	2	0.2000	0.2143	0.2002
sparrow	5	0.5000	0.4286	0.4990
bluejay	3	0.3000	0.2857	0.2998
unseen	0	0.0000	0.0714	0.0010
Total	10	1.0	1.0	1.0

- And to demonstrate the language's utility, here is the interactive Python session I used to calculate these numbers.

```
>>> b = {'robin':2, 'sparrow':5, 'bluejay':3, 'unseen':0}
>>> N = sum(b.values())
>>> V = len(b.keys())
>>> mle = lambda x: b.get(x,0)/float(N)
>>> print "\n".join(["%s\t%.4f" % (bird, mle(bird)) for bird in b.keys()])
unseen 0.0000
bluejay 0.3000
robin 0.2000
sparrow 0.5000
>>> sum([mle(bird) for bird in b.keys()]) # Check normalization
1.0
```

```

>>> lap = lambda x: (b.get(x,0) + 1)/(float(N) + V)
>>> print "\n".join(["%s\t%.4f" % (bird, lap(bird)) for bird in b.keys()])
unseen 0.0714
bluejay 0.2857
robin 0.2143
sparrow 0.4286
>>> sum([lap(bird) for bird in b.keys()]) # Check normalization
1.0
>>> delta = 0.01
>>> lid = lambda x: (b.get(x,0) + delta)/(float(N) + delta*V)
>>> print "\n".join(["%s\t%.4f" % (bird, lid(bird)) for bird in b.keys()])
unseen 0.0010
bluejay 0.2998
robin 0.2002
sparrow 0.4990
>>> sum([lid(bird) for bird in b.keys()]) # Check normalization
1.0

```

4 Smoothing in Language Applications

- A fundamental feature of natural language is its ability to generate utterances that we understand even though they are completely novel.
- Noam Chomsky gave the famous example “Colorless green ideas sleep furiously”. Though this is a weird sentence that you have probably never encountered before², you have no problem recognizing it as valid English and understanding it.
- For this reason, smoothing techniques are essential in natural language applications.
- If we wanted use Lidstone smoothing in an N-gram model the role of the unseen bird is played by any type that does not occur in the training corpus. N would be then number of tokens in the training corpus and V ³ would be the number of types (including the unseen type).
- For example, say we have a (tiny) training corpus consisting of the phrase “to be or not to be”. From this we could generate the following unsmoothed and smoothed unigram models.

$$N = 6 \tag{20}$$

$$V = 5 \tag{21}$$

²Unless you’re a linguist, in which case you’re heard it a million times already. Regardless, it’s easy to come up with some other completely novel utterance, e.g. “their emu wrangler is a drunk”.

³Again, aka the vocabulary size.

Type	Count	MLE	LID _{0.01}
be	2	0.3333	0.3328
not	1	0.1667	0.1669
or	1	0.1667	0.1669
to	2	0.3333	0.3328
unseen	0	0.0000	0.0017
Total	6	1.0000	1.0000

So according to the Lidstone smoothed unigram model

$$p_{LID}(\text{be}) = 0.3328 \quad (22)$$

and

$$p_{LID}(\text{habedashery}) = p_{LID}(\text{tomahawk}) = p_{LID}(\text{from}) = 0.0017 \quad (23)$$

- Again, here's the Python session.

```
>>> corpus = 'to be or not to be'
>>> freq = {}
>>> for ty in corpus.split(): freq[ty] = freq.get(ty, 0) + 1
...
>>> freq
{'not': 1, 'to': 2, 'or': 1, 'be': 2}
>>> types = freq.keys()
>>> types.sort()
>>> types.append('unseen')
>>> N = sum(freq.values())
>>> N
6
>>> V = len(types)
>>> V
5
>>> mle = lambda x:freq.get(x, 0)/float(N)
>>> print "\n".join(["%s\t%.4f" % (ty, mle(ty)) for ty in types])
be      0.3333
not     0.1667
or      0.1667
to      0.3333
unseen  0.0000
>>> sum([mle(ty) for ty in types]) # Check normalization
1.0
>>> delta = 0.01
>>> lid = lambda x: (freq.get(x,0)+delta)/(float(N)+delta*V)
```

```

>>> print "\n".join(["%s\t%.4f" % (ty, lid(ty)) for ty in types])
be      0.3328
not     0.1672
or      0.1672
to      0.3328
unseen  0.0017
>>> sum([lid(ty) for ty in types]) # Check normalization
0.9999999999999999

```

Here Lidstone normalization has a roundoff error, but this is still sufficiently accurate for our purposes.

- The maximum likelihood conditional distribution of context length m is the following

$$p_{MLE}(w_n|w_{n-m}^{n-1}) = \frac{C(w_{n-m}^n)}{C(w_{n-m}^{n-1})} \quad (24)$$

where w_i^j is the consecutive sequence of the words between w_i and w_j inclusive. So for example, $p(\text{Spain}|\text{rain in})$ is equal to the number of times “rain in Spain” occurs in the corpus divided by the number of times “rain in” occurs.

$$p(\text{Spain}|\text{rain in}) = \frac{C(\text{rain in Spain})}{C(\text{rain in})} \quad (25)$$

- The Lidstone smoothed distribution of a conditional N-gram is the following.

$$p_{LID}(w_n|w_{n-m}^{n-1}; \delta) = \frac{C(w_{n-m}^n) + \delta}{C(w_{n-m}^{n-1}) + \delta V} \quad (26)$$