Ling 5801: Lecture Notes 13
Probability Models

Contents

13.1 Full joint models and sparse data problems

We can estimate probablities for new atomic events based on frequencies of these events in training (assume a probability space $\langle O, 2^O, \mathsf{P}\rangle$ and frequency space $\langle O, 2^O, \mathsf{F}\rangle$ with $O = X_1 \times X_2 \times \cdots \times X_V$):

$$\mathsf{P}(x_1, x_2, \ldots, x_V) \stackrel{\text{def}}{=} \frac{\mathsf{F}(x_1, x_2, \ldots, x_V)}{\mathsf{F}(O)}$$

This is called a **full joint distribution**.

For examlpe, if we have a space of regions, words and sounds: $O = \mathit{Reg} \times \mathrm{Wrd} \times \mathrm{Acou}$, where:

$\mathit{Reg} = \{\mathrm{ohio}, \mathrm{phil}, \ldots\}$	(speaker region)
$\mathit{Wrd} = \{\mathrm{neck}, \mathrm{knack}, \ldots\}$	(speaker's intended word)
$\mathit{Acou} = \{[\mathrm{n\varepsilon k}], [\mathrm{n\ae k}], \ldots\}$	(listener's observed phone)

For example, Ohians never pronounce /ɛ/ as [æ], but Philadelphians often do:

$\mathsf{P}_{\theta_{Pron}}(\mathit{Reg} = \mathrm{ohio}, \mathit{Wrd} = \mathrm{knack}, \mathit{Acou} = [\mathrm{n\ae k}]) = .2$
$\mathsf{P}_{\theta_{Pron}}(\mathit{Reg} = \mathrm{ohio}, \mathit{Wrd} = \mathrm{knack}, \mathit{Acou} = [\mathrm{n\varepsilon k}]) = 0$
$\mathsf{P}_{\theta_{Pron}}(\mathit{Reg} = \mathrm{ohio}, \mathit{Wrd} = \mathrm{neck}, \mathit{Acou} = [\mathrm{n\ae k}]) = 0$ (never)
$\mathsf{P}_{\theta_{Pron}}(\mathit{Reg} = \mathrm{ohio}, \mathit{Wrd} = \mathrm{neck}, \mathit{Acou} = [\mathrm{n\varepsilon k}]) = .3$
$\mathsf{P}_{\theta_{Pron}}(\mathit{Reg} = \mathrm{phil}, \mathit{Wrd} = \mathrm{knack}, \mathit{Acou} = [\mathrm{n\ae k}]) = .2$
$\mathsf{P}_{\theta_{Pron}}(\mathit{Reg} = \mathrm{phil}, \mathit{Wrd} = \mathrm{knack}, \mathit{Acou} = [\mathrm{n\varepsilon k}]) = 0$
$\mathsf{P}_{\theta_{Pron}}(\mathit{Reg} = \mathrm{phil}, \mathit{Wrd} = \mathrm{neck}, \mathit{Acou} = [\mathrm{n\ae k}]) = .2$ (often)
$\mathsf{P}_{\theta_{Pron}}(\mathit{Reg} = \mathrm{phil}, \mathit{Wrd} = \mathrm{neck}, \mathit{Acou} = [\mathrm{n\varepsilon k}]) = .1$

We can write this as probability table, where rows sum to one:

$\mathsf{P}_{\theta_{Pron}}(Reg, Wrd, Acou) =$

ohio knack [næk]	ohio knack [nɛk]	ohio neck [næk]	ohio neck [nɛk]	phil knack [næk]	phil knack [nɛk]	phil neck [næk]	phil neck [nɛk]
.2	0	0	.3	.2	0	.2	.1

Each probability in the model is called a ***parameter***.

There's one for each combination of values: in this case $2 \cdot 2 \cdot 2 = 8$.

But if there are too many outcomes for the training data, full joints can make arbitrary predictions.

For example, consider this (junk/not-junk) email classifier algorithm:

1. Memorize a set of training emails (random variables are for first word, second word, etc.).
2. Compare each test email to the training set (estimate probability of atomic event).
   (a) If you find an exact match, report the class from the matching training example.
   (b) If you find no exact match, call it whatever is most common in training data (not-junk).

This will very rarely find an exact match, and end up classifying virtually nothing as junk.

This is called a **sparse data problem**. It shows the importance of *generalizing* from data.

13.2 Marginals and conditionals

One way to make better use of scarse data is to consolidate training instances across conditions.

We can do this by marginalizing or summing out certain variables that don't matter as much.

1. We can define **marginal** distributions from the Kolmogorov axioms:

$$\mathsf{P}(y, y', \dots) = \sum_{x, x', \dots \in X \times X' \times \dots} \mathsf{P}(x, x', \dots, y, y', \dots)$$

   For example:

$$\mathsf{P}_{\theta_{Pron}}(Wrd = w, Acou = a) = \sum_{r \in Reg} \mathsf{P}_{\theta_{Pron}}(Reg = r, Wrd = w, Acou = a)$$

$\mathsf{P}_{\theta_{Pron}}(Wrd, Acou) =$

knack [næk]	knack [nɛk]	neck [næk]	neck [nɛk]
.4	0	.2	.4

   Another example:

$$\mathsf{P}_{\theta_{Pron}}(Acou = a) = \sum_{r \in Reg, w \in Wrd} \mathsf{P}_{\theta_{Pron}}(Reg = r, Wrd = w, Acou = a)$$

$\mathsf{P}_{\theta_{Pron}}(Acou) =$

[næk]	[nɛk]
.6	.4

2. We can then define **conditional** distributions based on these marginals:

$$\mathsf{P}(x, x', \ldots \mid y, y', \ldots) = \frac{\mathsf{P}(x, x', \ldots, y, y', \ldots)}{\mathsf{P}(y, y', \ldots)}$$

For example:

$$\mathsf{P}_{\theta_{Pron}}(Wrd = w \mid Acou = a) = \frac{\mathsf{P}_{\theta_{Pron}}(Wrd = w, Acou = a)}{\mathsf{P}_{\theta_{Pron}}(Acou = a)}$$
$$= \frac{\sum_{r \in Reg} \mathsf{P}_{\theta_{Pron}}(Reg = r, Wrd = w, Acou = a)}{\sum_{r \in Reg, w \in Wrd} \mathsf{P}_{\theta_{Pron}}(Reg = r, Wrd = w, Acou = a)}$$

Table has one row for each combination of conditioned-on variable values:

$\mathsf{P}_{\theta_{Pron}}(Wrd \mid Acou) =$

Acou	knack	neck
[næk]	.667	.333
[nɛk]	0	1

So, by our model, we can calculate that [næk] is probably from knack in the general case:

$$\mathsf{P}_{\theta_{Pron}}(Wrd = \text{knack} \mid Acou = [\text{næk}]) = \frac{.2 + .2}{.2 + 0 + .2 + .2} = .667$$

but if the speaker is from Philadelphia, the odds are even:

$$\mathsf{P}_{\theta_{Pron}}(Wrd = \text{knack} \mid Acou = [\text{næk}], Reg = \text{phil}) = \frac{.2}{.2 + .2} = .5$$

Practice

Write a Python program to calculate $\mathsf{P}(Wrd \mid Acou)$ in a dictionary `WgivA[w,a]` given a full joint distribution $\mathsf{P}(Reg, Wrd, Acou)$ in a dictionary `RWA[r,w,a]`.

13.3 Factored models

We can consolidate data by first factoring our model, using the definition of conditional probability (you can validate this by cross-cancelling the numerators and denominators):

$$\mathsf{P}(x_1, x_2, \ldots, x_{V-1}, x_V) = \mathsf{P}(x_1) \cdot \underbrace{\frac{\mathsf{P}(x_1, x_2)}{\mathsf{P}(x_1)}} \cdot \underbrace{\frac{\mathsf{P}(x_1, x_2, x_3)}{\mathsf{P}(x_1, x_2)}} \cdots \underbrace{\frac{\mathsf{P}(x_1, x_2, \ldots, x_{V-1}, x_V)}{\mathsf{P}(x_1, x_2, \ldots, x_{V-1})}}$$
$$= \mathsf{P}(x_1) \cdot \mathsf{P}(x_2 \mid x_1) \cdot \mathsf{P}(x_3 \mid x_1, x_2) \cdots \mathsf{P}(x_V \mid x_1, x_2, \ldots, x_{V-1})$$

This is called a **chain rule decomposition**.

Now, in our example:

$$\mathsf{P}_{\theta_{Pron}}(Reg, Wrd, Acou) = \frac{\mathsf{P}_{\theta_{Pron}}(Reg)}{1} \cdot \frac{\mathsf{P}_{\theta_{Pron}}(Reg, Wrd)}{\mathsf{P}_{\theta_{Pron}}(Reg)} \cdot \frac{\mathsf{P}_{\theta_{Pron}}(Reg, Wrd, Acou)}{\mathsf{P}_{\theta_{Pron}}(Reg, Wrd)}$$
$$= \mathsf{P}_{\theta_{Reg}}(Reg) \cdot \mathsf{P}_{\theta_{Wrd}}(Wrd \mid Reg) \cdot \mathsf{P}_{\theta_{Acou}}(Acou \mid Reg, Wrd)$$

Now the number of parameters in any variable's conditional probability distribution:

$\mathsf{P}_{\theta_{X_i}}(X_i \mid X_1, ..., X_{i-1})$

is the product of the cardinalities of its modeled and conditioned-on variables:

$|\theta_{X_i}| = |X_i| \cdot |X_1| \cdot ... \cdot |X_{i-1}|$

In our example:

$|\theta_{Acou}| = |Acou| \cdot |Reg| \cdot |Wrd| = 2 \cdot 2 \cdot 2 = 8$ parameters

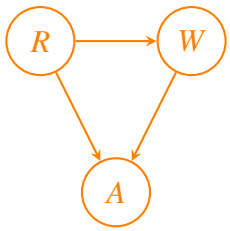
So far this is no better than the full joint distribution. . . but that's ok for now.

13.4 Graphical representation ('Bayes net')

Terms $\mathsf{P}_{\theta_{X_i}}(X_i \mid C_{X_i})$ from chain can be represented graphically —

- **circles** for random variables X_i
- **arrows** for conditional dependencies (from conditioned-on C_{X_i} to modeled variables)

For example, probability space $\langle R \times W \times A, 2^{R \times W \times A}, \mathsf{P} \rangle$ would be drawn:



Graphical models are like sudoku: some values given, some unknown, values interdepend

In fact, they can be considered a *generalization* of sudoku puzzles!

13.5 Conditional probability model class

Extend '`model.py`' to implement conditional model: dictionary of dictionary

(You may find this useful for certain problem set questions!)

```
import re
# define distribution to map value tuples to probs (or frequencies or scores)
class Model(dict):
    # init with model id
    def __init__(self,i=''):
        self.id = i
    # read model
    def read(self,s):
        m = re.search('^ *'+self.id+' +: +(.*?) += +(.*) *',s)
```

```
        if m is not None:
            v = tuple(re.split(' +',m.group(1)))
            if len(v)==1: v = v[0]
            self[v] = float(m.group(2))
    # write model
    def write(self):
        for v in sorted(self):
            s = self.id
            s = s + ' :'
            if type(v) is tuple:
                for f in v:
                    s = s + ' ' + f
            else: s = s + ' ' + v
            print( s + ' = ' + str(self[v]) )

# define model to map condition tuples to distributions
class CondModel(dict):
    # populate with default values when queried on missing keys
    def __missing__(self,k):
        self[k]=Model()
        return self[k]
    # define get without promiscuity
    def get(self,k):
        return dict.get(self,k,Model())
    # init with model id
    def __init__(self,i):
        self.id = i
    # read model
    def read(self,s):
        m = re.search('^ *'+self.id+' +(.*?) +: +(.*?) += +(.*) *',s)
        if m is not None:
            c = tuple(re.split(' +',m.group(1)))
            if len(c)==1: c = c[0]
            v = tuple(re.split(' +',m.group(2)))
            if len(v)==1: v = v[0]
            self[c][v] = float(m.group(3))
    # write model
    def write(self):
        for c in sorted(self):
            for v in sorted(self[c]):
                s = self.id
                if type(c) is tuple:
                    for f in c:
                        s = s + ' ' + f
                else: s = s + ' ' + c
                s = s + ' :'
                if type(v) is tuple:
```

```
                for f in v:
                    s = s + ' ' + f
            else: s = s + ' ' + v
            print( s + ' = ' + str(self[c][v]) )
```

Example, e.g. type into a program file '`myprog.py`':

```
import re
import sys
import model

# read in params beginning with 'R' (prior model: no value before colon)
# read in params beginning with 'W' (prior model: no value before colon)
# read in params beginning with 'A' (cond model: value before colon)
R = model.Model('R')
W = model.Model('W')
A = model.CondModel('A')
for line in sys.stdin:
    R.read(line)
    W.read(line)
    A.read(line)

# use a prior model
for r in R:
  print ( 'prob of '+r+' is '+str(R[r]) )
# use a conditional model
for r,w in A:
    for a in A[r,w]:
        print ( 'prob of '+a+' given '+r+' and '+w+' is '+str(A[r,w][a]) )
# calc prob of W='knack' given R='phil' and A='[naek]'
probAnyW = 0.0
for w in W:
  probAnyW = probAnyW + R['phil'] * W[w] * A['phil',w]['[naek]']
print ( R['phil'] * W['knack'] * A['phil','knack']['[naek]'] / probAnyW )
```

Reads files in the following format:

```
R : ohio = .5
R : phil = .5
W : knack = .4
W : neck  = .6
A ohio knack : [naek] = 1.0
A ohio neck  : [nek]  = 1.0
A phil knack : [naek] = 1.0
A phil neck  : [nek]  = .333333
A phil neck  : [naek] = .666667
⋮
```

then describes the information in the file and then prints the probability of knack:

```
0.499999875
```

13.6 Estimating θ from fully-specified ('annotated') data

Every θ_{X_i} is a conditional probability table $\mathsf{P}_{\theta_{X_i}}(X_i \mid C_{X_i})$.

Simply count instances of $x_1, ..., x_i \in C_{X_i} \times X_i$ and divide by count of $x_1, ..., x_{i-1} \in C_{X_i}$.

This is called **relative frequency estimation**.

13.7 Inducing θ from not-fully-specified ('unannotated') data

You'll study this in comp ling 2!

13.8 Independence assumptions

If we then think some variables don't influence others much, we can remove them from conditions:

$$\mathsf{P}_{\theta_X}(x \mid y, y', \ldots, z, z', \ldots) \stackrel{\text{def}}{=} \mathsf{P}(x \mid y, y', \ldots)$$

These re-definitions are called **independence assumptions**.

For example, our model of pronunciation variation may become:

$$\langle Reg \times Wrd \times Acou, 2^{Reg \times Wrd \times Acou}, \mathsf{P} \rangle$$
$$\mathsf{P}_{\theta_{Reg}}(Reg) \stackrel{\text{def}}{=} \mathsf{P}(Reg)$$
$$\mathsf{P}_{\theta_{Wrd}}(Wrd \mid Reg) \stackrel{\text{def}}{=} \mathsf{P}(Wrd)$$
$$\mathsf{P}_{\theta_{Acou}}(Acou \mid Reg, Wrd) \stackrel{\text{def}}{=} \mathsf{P}(Acou \mid Reg, Wrd)$$

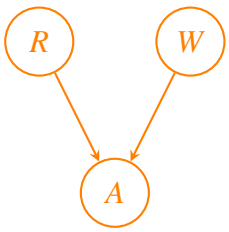
The joint probability is:

$$\mathsf{P}_{\theta_{Pron}}(Reg, Wrd, Acou) = \mathsf{P}_{\theta_{Reg}}(Reg) \cdot \mathsf{P}_{\theta_{Wrd}}(Wrd \mid Reg) \cdot \mathsf{P}_{\theta_{Acou}}(Acou \mid Reg, Wrd)$$

$$\mathsf{P}_{\theta_{Pron}}(Reg, Wrd, Acou) \stackrel{\text{def}}{=} \mathsf{P}_{\theta_{Reg}}(Reg) \cdot \mathsf{P}_{\theta_{Wrd}}(Wrd) \cdot \mathsf{P}_{\theta_{Acou}}(Acou \mid Reg, Wrd)$$

because of independence assumption: $\mathsf{P}_{\theta_{Pron}}(Wrd \mid Reg) \stackrel{\text{def}}{=} \mathsf{P}_{\theta_{Pron}}(Wrd)$

Graphically:



13.9 Another example: speech components

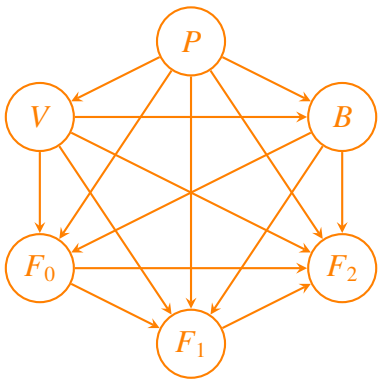
Here's an example of speech components: Phone, Voice, Back, Formant frequencies (binned):

$$\langle P \times V \times B \times F_0 \times F_1 \times F_2,\ 2^{P\times V\times B\times F_0\times F_1\times F_2},\ \mathsf{P}\rangle$$
$$\text{where } P = \{/\text{i}/, /\text{u}/\}, V = \{+,-\}, B = \{+,-\}, F_0 = \mathbb{I}_0^{99}, F_1 = \mathbb{I}_0^{99}, F_2 = \mathbb{I}_0^{99}$$

1. Chain rule decomposition (no independence assumptions):

$$\begin{aligned}\mathsf{P}_{\theta_{Sp}}(P,V,B,F_0,F_1,F_2) = {}& \mathsf{P}_{\theta_P}(P)\cdot\mathsf{P}_{\theta_V}(V\,|\,P)\cdot\mathsf{P}_{\theta_B}(B\,|\,P,V)\cdot\\ & \mathsf{P}_{\theta_{F_0}}(F_0\,|\,P,V,B)\cdot\mathsf{P}_{\theta_{F_1}}(F_1\,|\,P,V,B,F_0)\cdot\mathsf{P}_{\theta_{F_2}}(F_2\,|\,P,V,B,F_0,F_1)\end{aligned}$$

   Graphed, it produces this evil hexagram:



   Here $|\theta_{F_2}| = 100\cdot 2\cdot 2\cdot 2\cdot 100\cdot 100 = 8,000,000$ parameters!

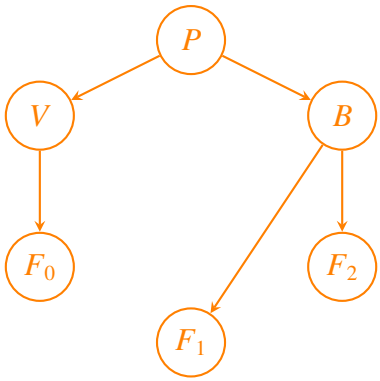
2. With independence assumptions:

$$\begin{aligned}\mathsf{P}_{\theta_P}(P) &\stackrel{\text{def}}{=} \mathsf{P}(P)\\ \mathsf{P}_{\theta_V}(V\,|\,P) &\stackrel{\text{def}}{=} \mathsf{P}(V\,|\,P)\\ \mathsf{P}_{\theta_B}(B\,|\,P,V) &\stackrel{\text{def}}{=} \mathsf{P}(B\,|\,P)\\ \mathsf{P}_{\theta_{F_0}}(F_0\,|\,P,V,B) &\stackrel{\text{def}}{=} \mathsf{P}(F_0\,|\,V)\\ \mathsf{P}_{\theta_{F_1}}(F_1\,|\,P,V,B,F_0) &\stackrel{\text{def}}{=} \mathsf{P}(F_1\,|\,B)\\ \mathsf{P}_{\theta_{F_2}}(F_2\,|\,P,V,B,F_0,F_1) &\stackrel{\text{def}}{=} \mathsf{P}(F_2\,|\,B)\end{aligned}$$

   and joint distribution:

$$\begin{aligned}\mathsf{P}_{\theta_{Sp}}(P,V,B,F_0,F_1,F_2) \stackrel{\text{def}}{=}{}& \mathsf{P}_{\theta_P}(P)\cdot\mathsf{P}_{\theta_V}(V\,|\,P)\cdot\mathsf{P}_{\theta_B}(B\,|\,P)\cdot\\ & \mathsf{P}_{\theta_{F_0}}(F_0\,|\,V)\cdot\mathsf{P}_{\theta_{F_1}}(F_1\,|\,B)\cdot\mathsf{P}_{\theta_{F_2}}(F_2\,|\,B)\end{aligned}$$

   it looks less busy:



Now $|\theta_{F_2}| = 100 \cdot 2 = 200$ parameters!

Practice

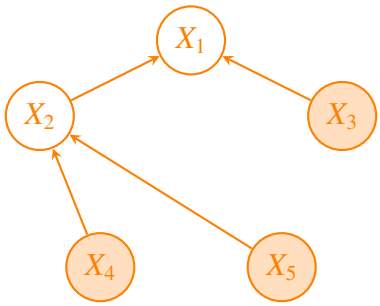
1. Draw a model of pet communication given random variables for vocalization, hunger, location, and tail being stepped on. The pet vocalizes when it's hungry, or when its tail is stepped on. Its tail gets stepped on only when it is in the kitchen.
2. Write the factored equation for $\mathsf{P}(Hunger, TailStep \mid Voc, Loc)$
3. Write a program to do this calculation in `modHTgivVL[v, l][h, t]` given models `modL[l]`, `modT[l][t]`, `modH[h]`, `modV[t, h][v]`

13.10 Generative vs discriminative models

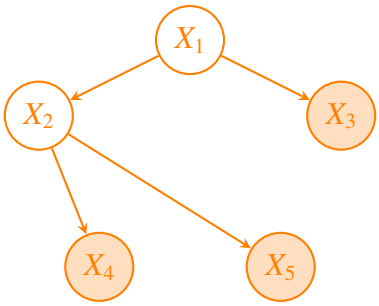
Random variables that are observed at test time are typically shaded in graphical representations.

These are called **observations**; the others are called **hidden** variables.

When these observed variables are conditioned on, the model is called **discriminative**:



When observed variables are *not* conditioned on, the model is called **generative**:


X_1
X_2
X_3
X_4
X_5