CSE250A Fall '12: Discussion Week 4

Aditya Menon (akmenon@ucsd.edu)

October 26, 2012

1 Schedule for today

- Maximum likelihood.
- Unigram and bigram models.

2 Maximum likelihood

We begin with a simple example. Suppose we have a Bayesian network with only one node, call it X. Say $X \in \{0, 1\}$, and $\Pr[X = 1] = p$ is unknown. So, X represents the outcome from flipping a biased coin. How could we go about estimating p, or equivalently, the bias of the coin?

Maximum likelihood estimation is a particular procedure for doing so. We assume that we can collect *independent* samples $\{x^{(1)}, \ldots, x^{(T)}\}$ from the network. These are just coin flips, of course. We now want to estimate p. The maximum likelihood solution is

$$\begin{aligned}
\hat{p} &= \mathrm{argmax}_p \Pr[x^{(1)}, \ldots x^{(T)}; p] \\
&= \mathrm{argmax}_p \log \Pr[x^{(1)}, \ldots x^{(T)}; p] \\
&= \mathrm{argmax}_p \log \prod_{t=1}^{T} \Pr[X = x^{(t)}; p] \\
&= \mathrm{argmax}_p \sum_{t=1}^{T} \log \Pr[X = x^{(t)}; p] \\
&= \mathrm{argmax}_p \sum_{t=1}^{T} x^{(t)} \log p + (1 - x^{(t)}) \log(1 - p).
\end{aligned}$$

It turns out that this is $\hat{p} = \sum_{t=1}^{T} x^{(t)}/T$, that is, the fraction of times the coin landed hands. This is what we expect intuitively!

The same idea holds for a general Bayesian network involving some random variables $X_1, \ldots, X_N$. We assume that we know the structure of the network, but don't know the values to put into the CPTs. Remember that these are weights

$$\Pr[X_i = x_i | \mathrm{Pa}(X_i) = \pi]$$

where $\mathrm{Pa}(X)$ denotes the parents of X, x_i all possible values of X_i, π all possible values of the parents of X_i. Let Θ refer to all such parameters in all CPTs in the network. To slightly abuse notation, let $\Theta(i, x_i, \pi) = \Pr[X_i = x_i | \mathrm{Pa}(X_i) = \pi]$.

Suppose we are given some independent random samples from the network, call them $\{x^{(1)}, \ldots, x^{(T)}\}$. Each $x^{(t)} = \{x_1^{(t)}, \ldots, x_N^{(t)}\}$. Note that

$$\begin{aligned}\Pr[(X_1, \ldots, X_N) = x^{(t)}; \Theta] &= \prod_{i=1}^{N} \Pr[X_i = x_i^{(t)} | \text{Pa}(X_i) = \pi_i^{(t)}; \Theta] \\ &= \prod_{i=1}^{N} \Theta(i, x_i^{(t)}, \pi_i^{(t)})\end{aligned}$$

This is from the canonical decomposition of a Bayesian network.

The principle of maximum likelihood says that we should choose

$$\hat{\Theta} = \text{argmax}_\theta \Pr[x^{(1)}, \ldots, x^{(T)} | \Theta].$$

Since the samples from the network are independent, this is equivalent to maximizing

$$\begin{aligned}\mathcal{L}(\Theta) &= \log \Pr[x^{(1)}, \ldots, x^{(T)} | \Theta] \\ &= \log \prod_{t=1}^{T} \Pr[x^{(t)} | \Theta] \\ &= \sum_{t=1}^{T} \log \Pr[x^{(t)} | \Theta] \\ &= \sum_{t=1}^{T} \sum_{i=1}^{N} \log \Pr[X_i = x_i^{(t)} | \text{Pa}(X_i) = \pi_i^{(t)}] \\ &= \sum_{t=1}^{T} \sum_{i=1}^{N} \log \Theta(i, x_i^{(t)}, \pi_i^{(t)}).\end{aligned}$$

The function $\mathcal{L}(\Theta)$ is called the *likelihood* function. Sometimes, this optimization can be done in closed form, like the coin flips example. But often it can't. Then, we resort to numerical optimization techniques like *gradient descent*.

3 Unigram models

Let's consider the problem of learning a probabilistic model for sentences through maximum likelihood. Such a model will let us generate random sentences that, depending on how good the model is, may seem remarkably coherent!

We'll represent a sentence of length N by $W_1, W_2, \ldots, W_N$. We will think of each W_i as being a random variable. This is because we want to think of a word's inclusion in a sentence as being governed by some probability. The possible values for the random variable W_i are the strings in some *vocabulary* $\mathcal{V}$, for example, all words in the English language.

For every sentence of length N, the unigram model of text posits that

- There are no relationships between the random variables W_i and W_j for $i \neq j$. That is, the Bayesian network is completely disconnected.
- The CPT for every node i, specifying $\Pr[W_i]$, is identical.

This is not a very sensible model: it says that a sentence is just a random collection of words, and that the words don't have to flow together. In particular, note that

$$\Pr[W_1 = w_1, W_2 = w_2] = \Pr[W_1 = w_2, W_2 = w_1].$$

This is known as the *bag of words* assumption: a sentence is formed by just tossing together some words into a "bag", where they all get jumbled up. (This property is also known as *exchangeability* of the random variables.)

3.1 Learning unigram models

The CPT for a given i specifies

$$\Pr[W_i = w] = \theta_w$$

for every word $w \in \mathcal{V}$. Therefore, the number of free parameters in a unigram model is $|\mathcal{V}| - 1$.

Suppose we have a *corpus* of sentences, $\{s^{(1)}, \ldots, s^{(T)}\}$, possibly of varying lengths, and we would like to model them with unigrams. We can ask: what is the "best" choice of CPT to explain the data? Equivalently, we want to find a way to choose the weights $\Theta = \{\theta_w : w \in \mathcal{V}\}$. Maximum likelihood says that we should choose the weights that maximize the probability of the sentences under a unigram model:

$$\hat{\Theta} = \mathrm{argmax}_\theta \Pr[s^{(1)}, \ldots, s^{(T)}; \Theta].$$

The log-likelihood simplifies to

$$\begin{aligned}
\log \Pr[s^{(1)}, \ldots, s^{(T)}; \Theta] &= \log \prod_{t=1}^{T} \Pr[s^{(t)}; \Theta] \\
&= \log \prod_{t=1}^{T} \prod_{i=1}^{N_i} \Pr[W_i = w_i^{(t)}; \Theta] \\
&= \sum_{t=1}^{T} \sum_{i=1}^{N_i} \log \Theta(w_i^{(t)}).
\end{aligned}$$

It turns out that, as expected, the optimal solution to the above is to just find the frequency of occurence of each word.

3.2 An example

Let's say that $s^{(1)}$ ="I want to know", $s^{(2)}$ ="I and I", $s^{(3)}$ ="I know what I want to", $s^{(4)}$ ="You and I". The vocabulary is { I, want, to, know, and, what, You }. The counts of the words are:

Word	Count
I	6
want	2
to	2
know	2
and	2
what	1
You	1

The CPT probabilities are derived by normalizing by the total number of words, namely, 16. What's the probability of the sentence "I want to" under this learned model?

$$\begin{aligned}
\Pr[\text{"I want to"}] &= \Pr[W_1 = \text{"I"}, W_2 = \text{"want"}, W_3 = \text{"to"}] \\
&= \Pr[\text{I}] \cdot \Pr[\text{want}] \cdot \Pr[\text{to}] \\
&= (6/16) \cdot (2/16) \cdot (2/16) \\
&\approx 0.0058.
\end{aligned}$$

Note that the probability of the sentence "to want I" under this learned model is identical, i.e. shuffling the words around doesn;t matter.

We see that the model penalizes long sentences at an exponential rate. To do better, we can try to exploit the structure of the sentence. Bigram models are a way to do this.

4 Bigram models

In a bigram model, we assume that

- There is a Markovian relationship between the random variables, i.e. $W_1 \to W_2 \to \ldots \to W_N$.
- The CPT for every node i, specifying $\Pr[W_i | W_{i-1}]$, is identical. (Generally we augment the network with dummy start and end nodes W_0, W_{N+1}.)

This is a better model. Once we generate a particular word in the sentence, that influences what the next word is going to be. (It does affect the word after too, but only *implicitly*.) Note that now, in general,

$$\Pr[W_1 = w_1, W_2 = w_2] \neq \Pr[W_1 = w_2, W_2 = w_1].$$

4.1 Learning bigram models

The CPT for a given i specifies

$$\Pr[W_i = w | W_{i-1} = w'] = \theta_{ww'}$$

for every pair of words $w, w' \in \mathcal{V}$. So, the total number of parameters to estimate in a bigram model is $|\mathcal{V}|^2 - 1$.

We can apply the principle of maximum likelihood to learn these weights also. For a given corpus, the log-likelihood is

$$\begin{aligned}
\log \Pr[s^{(1)}, \ldots, s^{(T)}; \Theta] &= \log \prod_{t=1}^{T} \Pr[s^{(t)}; \Theta] \\
&= \log \prod_{t=1}^{T} \prod_{i=1}^{N_i+1} \Pr[W_i = w_i^{(t)} | W_{i=1} = w_{i-1}^{(t)}; \Theta] \\
&= \sum_{t=1}^{T} \sum_{i=1}^{N_i+1} \log \Theta(w_i^{(t)}, w_{i-1}^{(t)}).
\end{aligned}$$

It turns out that the optimal parameters involving counting the fraction of times that word w follows word w'. For our example from before, the table of such counts looks like the following.

	START	I	want	to	know	and	what	You	END
START	0	3	0	0	0	0	0	1	0
I	0	0	2	0	1	1	0	0	2
want	0	0	0	2	0	0	0	0	0
to	0	0	0	0	1	0	0	0	1
know	0	0	0	0	0	0	0	0	0
and	0	2	0	0	0	0	0	0	0
what	0	1	0	0	0	0	0	0	0
You	0	0	0	0	0	1	0	0	0
END	0	0	0	0	0	0	0	0	0

To get the CPT probabilities, we divide each row by the sum of its elements.

4.2 An example

What's the probability of the sentence "I want to" under this learned model?

$$\begin{aligned}
\Pr[\text{"I want to"}] &= \Pr[\text{START} \to \text{I}] \cdot \Pr[\text{I} \to \text{want}] \cdot \Pr[\text{want} \to \text{to}] \cdot \Pr[\text{to} \to \text{END}] \\
&= (3/4) \cdot (2/6) \cdot (2/2) \cdot (1/2) \\
&= 0.125.
\end{aligned}$$

Note that the probability of the sentence "to want I" under this learned model is 0, i.e. shuffling the words around matters.