Chapter 2

Preliminaries

2.1 Probability

Below is a very brief review of basic probability theory. The notation used for probabilities in NLP is a little sloppy, but hopefully this is good enough. For a proper treatment, see the textbook by Bertsekas and Tsitsiklis (2008).

Random variables A random variable is a variable with a different random value in each "experiment". For example, if our experiments are coin flips, we could define a random variable $C \in \{\text{heads}, \text{tails}\}$ for the result of the flip. Or, if our experiments are the words of a speech, we could define a random variable $W \in \{\text{a}, \text{aa}, \text{ab}, \ldots\}$ for the words spoken. If X is a random variable with values in $\mathcal{X}$, we call $P(X)$ the distribution of X. If $x \in \mathcal{X}$, we write $P(X = x)$ for the probability that X has value x. We must have

$$\sum_{x \in \mathcal{X}} P(X = x) = 1.$$

For example, if $P(W)$ is a distribution over English words, we might have

$$\begin{aligned} P(W = \text{the}) &= 0.1 \\ P(W = \text{syzygy}) &= 10^{-10} \\ &\vdots \end{aligned}$$

Joint and marginal probabilities Things get more interesting when we deal with more than one random variable. For example, suppose our experiments are words spoken during a debate, and W is again the words spoken, while $S \in \{\text{Clinton}, \text{Trump}, \ldots\}$ is the person speaking. We can talk about the *joint distribution* of S and W, written $P(S, W)$, which should satisfy

$$\sum_{s,w} P(S = s, W = w) = 1.$$

Let's make up some numbers:

$$\begin{aligned} P(S = \text{Trump}, W = \text{bigly}) &= 0.2 \\ P(S = \text{Trump}, W = \text{huge}) &= 0.4 \\ P(S = \text{Clinton}, W = \text{people}) &= 0.3 \\ P(S = \text{Clinton}, W = \text{think}) &= 0.1. \end{aligned}$$

We also have to have

$$\begin{aligned} P(S = s) &= \sum_w P(S = s, W = w) \\ P(W = w) &= \sum_s P(S = s, W = w), \end{aligned}$$

known as marginal distributions.

Using our made-up numbers, we have

$$\begin{aligned} P(S = \text{Trump}) &= 0.2 + 0.4 = 0.6 \\ P(S = \text{Clinton}) &= 0.3 + 0.1 = 0.4 \end{aligned}$$

and

$$\begin{aligned} P(W = \text{bigly}) &= 0.2 \\ P(W = \text{huge}) &= 0.4 \\ P(W = \text{people}) &= 0.3 \\ P(W = \text{think}) &= 0.1. \end{aligned}$$

It's extremely common to write $P(w)$ as shorthand for $P(W = w)$. This leads to some sloppiness, because the symbol P is now "overloaded" to mean several things and you're supposed to know which one. To be precise, we should distinguish the distributions (using $P(S = s)$ or $P_S(s)$). But in NLP, we deal with some fairly complicated structures, and it becomes messy to keep this up. In practice, it's rarely a problem to use the sloppier notation.

Conditional probabilities We also define the *conditional distributions*

$$\begin{aligned} P(s \mid w) &= \frac{P(s, w)}{P(w)} \\ P(w \mid s) &= \frac{P(s, w)}{P(s)}. \end{aligned}$$

Note that

$$\begin{aligned} \sum_s P(s \mid w) &= 1 \\ \sum_w P(w \mid s) &= 1. \end{aligned}$$

You should know this already, but it should be second nature, and in particular, be sure never to get $p(s \mid w)$ and $p(w \mid s)$ confused! Using our made-up numbers:

$$P(\text{Trump} \mid \text{bigly}) = 0.2/0.2 = 1$$
$$P(\text{bigly} \mid \text{Trump}) = 0.2/0.6 \approx 0.333.$$

Expected values Finally, if a random variable has numeric values, we can talk about its average or expected value. For example, let $c_{\text{e}}(w)$ be the number of occurrences of the letter e in w. The *expectation* of c_{e} is

$$E[c_{\text{e}}] = \sum_w P(W = w) c_{\text{e}}(w),$$

and using our made-up numbers, this is

$$E[c_{\text{e}}] = 0.2 \cdot 0 + 0.4 \cdot 1 + 0.3 \cdot 2 + 0.1 \cdot 0 = 1.$$

2.2 Logarithms

You learned logarithms a long time ago, but you'll really use them a lot in this class. The following identities should be second nature:

$$\log \exp x = x \qquad \exp \log x = x$$
$$\log xy = \log x + \log y \qquad \exp(x + y) = \exp x \exp y$$
$$\log \prod_i x_i = \sum_i \log x_i \qquad \exp \sum_i x_i = \prod_i \exp x_i$$
$$\log x^n = n \log x \qquad \exp nx = (\exp x)^n$$
$$\log 1 = 0 \qquad \exp 0 = 1$$

We will always assume that log and exp have base e.

Log-probabilities Logarithms are used a lot to simplify expressions like this product of many probabilities:

$$p(x_1, \ldots, x_n) = \prod_i p(x_i).$$

It's extremely common to take the log of everything, changing the product into a sum:

$$\log p(x_1, \ldots, x_n) = \sum_i \log p(x_i).$$

There are a few of reasons for this. First, it used to be that additions are faster than multiplications, but we don't worry about this anymore (in fact, floating-point multiplication is sometimes faster). Second, it's often easier on paper to work with sums instead of products. (For example, taking derivatives is easier.)

Third, a product of many probabilities quickly becomes a very small number. An IEEE 754 double only goes down to 10^{-308}, and we often deal with probabilities much smaller than that. To avoid underflow, the typical solution is to use log-probabilities.

Computing with log-probabilities is easy. If we have two log-probabilities $\log p$ and $\log q$, instead of multiplying p and q, we add $\log p$ and $\log q$ (because $\log pq = \log p + \log q$). To compare p and q, just compare $\log p$ and $\log q$, which is equivalent.

The only tricky part is addition. To compute $\log(p+q)$ given $\log p$ and $\log q$, we can't do this:

$$\log(p+q) = \log(\exp \log p + \exp \log q)$$

because either of the exp's might cause an underflow. What should you do instead? The short answer is that in Python, you should use `numpy.logaddexp`. The long answer is: Assume that $p > q$; if not, swap them. Then, observe that:

$$\begin{aligned}
\log(p+q) &= \log p \left(1 + \frac{q}{p}\right) \\
&= \log p + \log \left(1 + \frac{q}{p}\right) \\
&= \log p + \log \left(1 + \exp \log \frac{q}{p}\right) \\
&= \log p + \log(1 + \exp(\log q - \log p)).
\end{aligned}$$

Now, the exp could still cause an underflow, but the underflow is harmless. (Why?) For an extra little boost in accuracy, you can use the `log1p` function, found in nearly all standard libraries, which computes $\log(1+x)$ but is accurate for small x.

Note that if p is a probability, $\log p$ is negative or zero. Sometimes we work with $-\log p$, which is positive or zero, but is confusingly called a *negative log-probability*.

Softmax Suppose that $x_1, x_2, \ldots, x_n$ are log-probabilities; then their exps should sum to one. But sometimes, it's more convenient to let them be unconstrained and make them sum to one by dividing by their sum. In neural networks, this is called a *softmax*. Let $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ be a vector of real numbers (positive or negative), and define softmax $\mathbf{x}$ to be the vector

$$[\text{softmax } \mathbf{x}]_i = \frac{\exp x_i}{\sum_{i'=1}^{n} \exp x_{i'}},$$

where $\exp \mathbf{x}$ means the elementwise exp of $\mathbf{x}$. It's also common to express this as

$$\log [\text{softmax } \mathbf{x}]_i = x_i - \log \sum_{i'=1}^{n} \exp x_{i'},$$

where the second term is computed using the trick above.

Bibliography

Bertsekas, Dimitri P. and John N. Tsitsiklis (2008). *Introduction to Probability*. 2nd. Athena Scientific.