CS769 Spring 2010 Advanced Natural Language Processing

Language Modeling

Lecturer: Xiaojin Zhu *jerryzhu@cs.wisc.edu*

In this lecture we will define the probability of a whole sentence or document, using the so-called n-gram approximation. We will also see why the MLE can be bad at times, and ways to smooth it.

1 The Need for Finding More Likely Text

Consider these problems:

- Text messaging, chat, and IM. Internet slang. What is YKWIM? How can we decode acronyms that are not invented yet (a dictionary is not the solution)?
- Say you're designing a speech recognizer. Why do you prefer

  s_1 =It's hard to recognize speech

  over

  s_2 =It's hard to wreck a nice beach?

  The two sentences have nearly the same acoustic signal. We must have some intrinsic preference over sentences.
- Similarly, if you are building an Optical Character Recognition (OCR) system, there are plenty of ambiguity in input (e.g., 1 vs. l, 0 vs. O). But we know that certain letter sequences are more likely than others.

Language modeling tries to capture the notion that some text is more likely than others. It does so by estimating the probability $P(s)$ of any text s. Given input a (be it acronyms, acoustic sound, or digitized pixels), the recognition process can be defined as finding the text s that maximizes, among all texts, the conditional probability

$$s^* = \arg\max_{s \in \mathcal{S}} P(s|a). \tag{1}$$

By Bayes rule, we have

$$P(s|a) = \frac{P(a|s)P(s)}{P(a)}. \tag{2}$$

Note $P(a)$ is a constant w.r.t. the maximization over s, so that the problem reduces to

$$s^* = \arg\max_{s \in \mathcal{S}} P(a|s)P(s). \tag{3}$$

The term $P(a|s)$ describes the probability of output a that can be generated from the text s. For acronym decoding, it could be as simple as a deterministic 0/1 function.

In the following we assume words are the basic units, i.e. $s = w_1 \dots w_n \equiv w_{1:n}$. Estimating $P(s)$ directly by counting is not feasible, if n is large. We will use the *chain rule*: [1]

$$P(s) = P(w_{1:n}|\theta) = P(w_1|\theta)P(w_2|w_1,\theta)P(w_3|w_{1:2},\theta)\dots P(w_n|w_{1:n-1},\theta). \tag{4}$$

[1] Technically this is not a complete definition of $P(s)$ if s varies in length: there should be a distribution over sentence length $P(n)$, and $P(s) = P(n)P(w_{1:n})$.

This does not buy us anything yet – the conditional probabilities are impossible to estimate if the history is long. We will look at n-gram language modeling, which approximates the chain rule by shortening the histories.

2 Unigram Language Model

A *unigram* (1-gram) language model makes the strong *independence assumption* that words are generated independently from a multinomial distribution θ (of dimension V, the size of the vocabulary). That is,

$$P(w_{1:n}|\theta) = \prod_{i=1}^{n} P(w_i|\theta) = \prod_{w=1}^{V} \theta_w^{c_w}, \tag{5}$$

where c_w is the count of word w in s. This is the multinomial distribution over c, except that we do not have the combinatorial coefficient, because we know the particular sequence. Note there is no history or conditional probability at all. The unigram approximation is

$$P(w_i|w_{1:i-1}, \theta) \approx P(w_i|\theta).$$

You might be alarmed that this seems to be a particularly bad model of language, as it ignores word orders! You are right – more on this later. But it is *useful*.

Where do we get θ? We estimate it from a corpus. We need a vocabulary of V word types, and counts $c_{1:V}$ of each word type in the corpus. The MLE is

$$\theta_w^{ML} = \frac{c_w}{\sum_{w=1}^{V} c_w} = \frac{c_w}{|C|}, \tag{6}$$

where $|C| = \sum_{w=1}^{V} c_w$ is the length of the corpus. In this case the MLE is simply the frequency estimate, as we've seen before.

There is a tiny problem with the MLE: if a word type w (e.g., aardvark) is in the vocabulary but not in the corpus (hence $c_w = 0$), the MLE is $\theta_w = 0$. Any new sentence with aardvark in it will thus be considered impossible since it have zero probability. This does not make sense. No problem — this can be avoided by restricting the vocabulary to the words in the corpus – for now. For document classification, people found that unigram is often sufficient. Trigrams have been the standard for speech recognition.

3 N-gram Language Models

A better approximation to the chain rule is to keep *some* history. In particular, for an n-gram language model the approximation is

$$P(w_i|w_{1:i-1}) \approx P(w_i|w_{i-n+1:i-1}). \tag{7}$$

The conditioning part $w_{i-n+1:i-1}$ is called 'history', which has $n-1$ previous words. Compared to the chain rule (4), the n-gram language model dictates that

$$P(w_{1:n}|\theta) = \prod_{i=1}^{n} P(w_i|w_{i-n+1:i-1}, \theta). \tag{8}$$

Unigram is a special case when $n = 1$. Common names include bigram ($n = 2$) and trigram ($n = 3$). N-grams traditionally do not run across sentences.

It is worth noting that the number of parameters in θ grows rapidly as $O(V^n)$. Another way to look at it is that for a given history h, $P(w|h)$ is a multinomial of size V, but there are V^{n-1} such possible histories, and θ consists of all these multinomials. For $V = 10,000$ which is typical, in theory there are 10,000-1 unigram parameters, 10^8 bigram parameters (compared to US population 3×10^8), and 10^{12} trigrams (about

150 trigrams per person in the world). In practice, the number of n-grams is bounded by corpus length. In 2006, Google released a "Web 1T 5-gram" corpus on 6 DVDs, with counts from 10^{12} tokens. `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2006T13` In the Google 1T 5-gram corpus:

```
Number of tokens:    1,024,908,267,229
Number of sentences:    95,119,665,584
Number of unigrams:         13,588,391
Number of bigrams:         314,843,401
Number of trigrams:        977,069,902
Number of fourgrams:     1,313,818,354
Number of fivegrams:     1,176,470,663
```

It should be clear that the data sparseness problem is much more severe with a larger n, because the histories "fragment" the corpus. There are two issues:

1. Some history might not appear in corpus at all.
2. For a history that does appear, not all word types will follow it.

The MLE is in big trouble now. A more satisfying solution is known as smoothing in the language modeling community, which is related to shrinkage in statistics, or, in certain cases, arises from an estimate other than MLE as we show next.

4 Smoothing

There are a large number of smoothing methods for language modeling, e.g., Good-Turing, Jelinek-Mercer interpolated, Katz, Whitten-Bell, Absolute discounting, and Kneser-Ney.

4.1 The Maximum A Posteriori (MAP) Estimate and Additive Smoothing

We view θ as a random variable too. Let's go back to unigram and think of the corpus as a count vector $c_{1:V}$. The MAP estimate finds the most likely parameter, given the corpus. This is to find the maximum in the posterior probability

$$\theta^{MAP} = \arg\max_{\theta} p(\theta|c_{1:V}).$$

Applying Bayes rule,

$$\begin{aligned}
\theta^{MAP} &= \arg\max_{\theta} p(\theta|c_{1:V}) && (9)\\
&= \arg\max_{\theta} \frac{p(c_{1:V}|\theta)p(\theta)}{p(c_{1:V})} && (10)\\
&= \arg\max_{\theta} p(c_{1:V}|\theta)p(\theta). && (11)
\end{aligned}$$

The last line follows because $p(c_{1:V})$ is a constant in optimization. We see the familiar likelihood function $p(c_{1:V}|\theta)$ which the MLE maximizes. We also see something new, the *prior distribution* $P(\theta)$, which needs to be set according to our prior knowledge about θ. The MAP estimate is the same as the MLE if the prior is uniform, but otherwise is different.

The Dirichlet distribution is a particularly convenient choice, since it is the *conjugate prior* of multinomial. A Dirichlet prior $P(\theta|\alpha)$ has parameters $\alpha_{1:V}$:

$$p(\theta|\alpha) = \frac{\Gamma(\sum \alpha_i)}{\prod \Gamma(\alpha_i)} \prod \theta_i^{\alpha_i - 1}. \quad (12)$$

It can be shown that the MAP estimate is

$$\theta_w^{MAP} = \frac{c_w + \alpha_w - 1}{|C| + \sum_{i=1}^{V}(\alpha_i - 1)}. \tag{13}$$

The hyperparameters thus act as "pseudo counts", i.e. $(\alpha_w - 1)$ can be viewed as the count of word w we collected before seeing the corpus. Clearly this avoids the zero probability problem. A particular choice of hyperparameters $\alpha_i = 2$ results in the so-called "add-one smoothing":

$$\theta_w^{MAP} = \frac{c_w + 1}{|C| + V}, \tag{14}$$

which is also known as Laplace smoothing, with which Laplace allegedly computed the probability that the Sun will rise again tomorrow. A simple variation leads to "add-ϵ smoothing". These smoothing methods are not the best in terms of performance, but are often used because of their simplicity.

4.2 Interpolation

One can take the MLE trigram LM $P_3(w|w_{i-2}w_{i-1})$, the MLE bigram LM $P_2(w|w_{i-1})$, the MLE unigram LM $P_1(w)$, and the uniform (zeroth order) LM $P_0(w) = 1/V$ where V is the vocabulary size, and create an intepolated LM:

$$P_{int}(w|w_{i-2}w_{i-1}) = \lambda_3 P_3(w|w_{i-2}w_{i-1}) + \lambda_2 P_2(w|w_{i-1}) + \lambda_1 P_1(w) + \lambda_0 P_0(w) \tag{15}$$

where the interpolation weights $\sum_n \lambda_n = 1$, $\lambda_n \geq 0$. If the bigram $w_{i-2}w_{i-1}$ never appear in the training corpus, one needs to ignore the MLE trigram LM (as it is not well-defined) by setting $\lambda_3 = 0$, and so on. The optimal λ's can be learned using the Expectation-Maximization algorithm, which will be discussed later in class.

4.3 Kneser-Ney

Kneser-Ney and its variants are the state-of-the-art smoothing techniques. A detailed discussion of its construction is outside the scope of this lecture, but let us point out two things:

1. It is among the smoothing methods that produce the best empirical results.
2. It has a nice Bayesian interpretation that ties to the so-called Pitman-Yor process.

Interested readers should refer to the reading list on the course webpage.

5 Other Types of Language Models

Sometimes one trains an LM on a corpus that is in a different domain than the future text. For example, the training corpus might be news articles, but the LM will be used in a speech recognizer for dictating personal emails. Given the LM (call it LM_0), and a little bit of text from the new domain, one wants to perform language model adaptation so that the new LM is better for the new domain. Here is one simple idea called cache LM. Train a new LM (call it LM_1) solely from the new text. This is going to be a poorly estimated LM since there is not enough new text. But one can interpolate it with the old LM to produce

$$P_{cache}(w|h) = \lambda P_0(w|h) + (1 - \lambda) P_1(w|h).$$

Many other types of LM have been proposed over the years, including class-based LM, tree LM, grammar-based LM, whole-sentence LM. However, a well-smoothed n-gram LM is surprisingly difficult to beat.

6 Evaluating Language Models

How do we know a language model (or a density estimator in general) is good or bad? The MLE is the 'best' (most likely) among all estimates on the training corpus $C = c_{1:V}$ by definition, and MAP seems to be even a bit inferior to MLE.

The answer lies in our interest in *future data*, not the corpus we trained on. Any model should give high probability to the kind of text we are interested in, but not seen in the training corpus. Therefore we assume we have a *test* corpus C', and we hope our θ gives high probability on any such C'.

Therefore test-set likelihood seems like a good quality measure for θ: $P(C'|\theta)$. But it is an extremely small number, and depends on the length of C'. A more useful measure is per-word test-set log likelihood:

$$\frac{1}{|C'|}\sum_{w=1}^{V} c'_w \log \theta_w. \tag{16}$$

If we call the actual distribution from which words are drawn $p(w)$, the above quantity is actually a stochastic approximation to

$$\sum_{w=1}^{V} p(w) \log \theta_w, \tag{17}$$

which becomes exact when $|C'| \to \infty$. We will come back to this in the next lecture.

A derived quantity, known as the *perplexity*, is widely used in language modeling:

$$PP(C';\theta) = 2^{-\frac{1}{|C'|}\sum_{w=1}^{V} c'_w \log_2 \theta_w} = P(C'|\theta)^{-\frac{1}{|C'|}}. \tag{18}$$

Note log base 2. Perplexity measures on average, how many 'equally likely' words we must choose from for each word position – the smaller the number, the more certain we are, and the better the model θ.

Beyond n-gram type language models, a variety of models have been proposed to try to "put language back into language model", including class-based LM, tree LM, grammar-based LM, Maximum entropy LM, whole sentence LM and so on. However, trigram language model is surprisingly hard to beat.

7 Sampling from a Distribution

We can generate sentences by sampling from the distribution $P(s)$ described by a language model. We write $s \sim P(s)$ if we select a sentence s at randomly according to the probability $P(s)$. For a unigram model, the sampling process is as simple as generating random numbers (word index) from the multinomial $P(w|\theta)$. For n-gram models, we sample from the conditional probability $P(w_i|w_{i-n+1:i-1}, \theta)$ where $w_{i-n+1:i-1}$ are the most recent $n-1$ words we just generated.

If this reminds you of a random walk on a Markov Chain, you are right. An n-gram LM is a $(n-1)$-th Markov model, θ is the transition matrix.

Shakespeare unigram (JM p.203):

- To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
- Every enter now severally so, let
- Hill he late speaks; or! a more to leg less first you enter
- Will rash been and by I the me loves gentle me not slavish page, the and hour; ill let
- Are where exeunt and sighs have rise excellency took of .. sleep knave we. near; vile like

Bigram:

- What means, sir. I confess she? then all sorts, he is trim, captain.
- Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
- What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?
- Enter Menenius, if it so many good direction found'st thou art a strong upon command of fear not a liberal largess given away, Falstaff! Exeunt
- Thou whoreson chops. Consumption catch your dearest friend, wekll, and I know where many mouths upon my undoing all but be, how soon, then; we'll execute upon my love's bonds and we do you will?

Trigram:

- Sweet prince, Falstaff shall die. Harry of Monmouth's grave.
- This shall forbid it should be branded, if renown made it empty.
- What is't that cried?
- Indeed the duke; and had a very good friend.
- Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.

4-gram:

- King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;
- Will you not tell me who I am?
- It cannot be but so.
- Indeed the short and the long. Marry, 'tis a noble Lepidus.
- They say all lovers swear more performance than they are wont to keep obliged faith unforfeited!