Probabilities and Language Models

Kevin Duh

Lecture Notes for Natural Language Processing, Fall 2019

1 Probability Basics

Natural language processing involves ambiguity resolution. Probability and statistics are effective frameworks to tackle this. Here, we will define some basic concepts in probability required for understanding language models and their evaluation.

Definitions

- Event space $\mathcal{E}$: the set of all possible outcomes
- Event $E \subseteq \mathcal{E}$: a subset of the event space
- Probability $P(E)$: a non-negative number assigned to an event, indicating it's probability of occurrence

	Event Space $\mathcal{E}$	Event E	Probability $P(E)$
The output of a coin flip	{Head, Tail}	{Head} {Tail}	$P(\{Head\}) = 0.5$ $P(\{Tail\}) = 0.5$

The output of a dice roll	{1, 2, 3, 4, 5, 6}	{1}
		{1,3,5}
			$P(\{1\}) = 1/6$
			$P(\{1,3,5\}) = 0.5$

The word after "Hi, my name..."	{is, was, ...} all vocabulary	{is} {was}	$P(\{is\}) = 0.7$ $P(\{was\}) = 0.1$
A random sentence from Shakespeare	all sentences		P({"To be or not to be"}) = 1E-5 P({"That's all, folks!"}) = 1E-9

Table 1: Example of events and their probability

Axioms of Probability

1. $\underline{P(E) \geq 0 \text{ for all events } E}$. Each subset of event space is assigned some probability of occurrence (may be zero).
2. $\underline{P(\mathcal{E}) = 1}$.

3. If two E_1 and E_2 are disjoint, then $P(E_1 or E_2) = P(E_1) + P(E_2)$. Dice roll example: $P(\{1,3,5\}) = P(\{1\})+P(\{3\})+P(\{5\}) = 1/6+1/6+1/6 = 0.5$. Extends to $P(\cup_{i=1}^{n} E_i) = \sum_{i=1}^{n} P(E_i)$ for mutually-exclusive E_i's.

Random Variable A random variable X is a variable whose possible values are the outcomes of some random trial, e.g. a coin flip. This can be discrete (e.g. Head or Tail) or continuous (e.g. any real number between 0 and ∞).

A probability mass function $p(x)$ is a function that assigns probability when a discrete random variable X is equal to some value x.

$$p(x) \overset{\text{def}}{=} P(X = x) \tag{1}$$

Dice roll example: Let x be 1. $p(1) = P(X = 1) = 1/6$; this is the probability we assign to the event $X = 1$. We also call $p(x)$ the probability distribution. For continuous random variables, we will be using a probability density function $f(x)$ to characterize the distribution; it defines probability on intervals: $P(a \leq X \leq b) = \int_a^b f(x)dx$.

Later in the course, we will be exploring many ways to "learn" a probability distribution from data. This distribution might be a multinomial distribution (which can be viewed as a table, with $p(x)$ values for each x), or a some log-linear parameterization, or some output of a softmax layer after a neural network. In all cases, we'll be using notation $x \sim Distribution(\theta)$ which means x is a sample drawn from the distribution with parameters θ.

Joint Probability Sometimes we care about the probability of two or more variables, and they might interact. This can be modeled by a joint probability of e.g. two variables X and Y:

$$p(x, y) \overset{\text{def}}{=} P(X = x, Y = y) \tag{2}$$

Say we flip two coins, where X represents the output of the first coin and Y represents that of the second coin. Our event space is:

$$\mathcal{E} = \{(Head, Head), (Head, Tail), (Tail, Tail), (Tail, Head)\} \tag{3}$$

and the probabilities are each $1/2 \times 1/2 = 1/4$, e.g. $p((Head, Tail)) = 1/4$. In this case, the two events do not influence each other, so the probability of both occurring is simply the product of their probabilities: $p(x, y) = p(x)p(y)$. We say that X and Y are **independent**.

Now consider a weather example, where we have two variables: X for rain and Y for cloud. Each event has two possible outcomes: {True, False}, so there are four possible outcomes in the joint event space. Suppose we observe, out of 12 days, the outcomes shown in Table 2.

We can compute the marginal probability from the joint probability by summing out the variable that is not considered: $p(x) = \sum_y p(x, y)$ and $p(y) = \sum_x p(x, y)$. In the weather example: P(Rain=True)

= P(Rain=True and Cloud=True) + P(Rain=True and Cloud=False) = 3/12 + 0/12 = 3/12.

Joint Probability	Cloud=True	Cloud=False	Marginal p(Rain)
Rain=True	3/12	0/12	3/12
Rain=False	1/12	8/12	9/12
Marginal p(Cloud)	4/12	8/12	

Table 2: Weather example: Joint and Marginal probabilities for Rain and Cloud

Definition of Independence Two variables are independent if and only if (iff) their joint probability is a product of the marginal probabilities. We see this is not the case for the weather example, e.g. P(Rain=True) $\times$ P(Cloud=True) $= 3/12 \times 4/12 \neq$ P(Rain=True and Cloud=True). Naturally, rain can't happen without clouds.

Conditional Probability Continuing from the weather example, we can ask "What's the probability of rain, given that we already know it is cloudy?" This is a conditional probability, which we would write as P(Rain=True $|$ Cloud=True). Formally,

$$p(x \mid y) \overset{\text{def}}{=} \frac{p(x,y)}{p(y)} \tag{4}$$

One can interpret this definition as restricting the sample space to situations where the conditioning factor occurs.[1] So P(Rain=True $|$ Cloud=True) = P(Rain=True and Cloud=True) $\div$ P(Cloud=True) $= 3/12 \div 4/12 = 3/4$.

Note that if two variables are independent, the $p(x \mid y) = \frac{p(x)p(y)}{p(y)} = p(x)$. This means knowledge of y does not tell us anything about x.

Chain Rule The conditional probability definition (Eq. 4) can be extended to more variables. First, start with $p(x \mid y, z) \overset{\text{def}}{=} \frac{p(x,y,z)}{p(y,z)}$. Also, $p(y \mid z) \overset{\text{def}}{=} \frac{p(y,z)}{p(z)}$. Putting the two together, we get:

$$p(x, y, z) = p(x \mid y, z)p(y \mid z)p(z) \tag{5}$$

In general, the chain rule says that a joint probability can be decomposed into a series of conditional probabilities like the form above.

[1]Take as another example the discussion of random sentence generation by PCFG in previous lectures. We have a grammar that generates many types of sentences, $s_1, s_2, s_3, \ldots$. At the same time, the sentence s_1 may have been generated by two different trees t_a and t_b. What if we want to know the probability of a specific tree t_a given s_1? We would calculate the conditional probability $p(t_a|s_1) = \frac{p(t_a,s_1)}{p(s_1)} = \frac{p(t_a,s_1)}{p(t_a,s_1)+p(t_b,s_1)}$. The division by $p(s_1)$ re-normalizes the probabilities to focus on cases where s_1 occurred. $p(t_a, s_1)$ and $p(t_b, s_1)$ can be obtained from our PCFG by multiplying the rule probabilities.

Bayes Rule Bayes Rule says:

$$p(x \mid y) = \frac{p(y \mid x)p(x)}{p(y)} \tag{6}$$

This can be derived from our definition of conditional probability. First, from Eq. 4, we get $p(x, y) = p(x|y)p(y)$ and $p(y, x) = p(y|x)p(x)$. Next, we know $p(x, y) = p(y, x)$, so $p(x|y)p(y) = p(y|x)p(x)$. Finally, assuming non-zero probabilities, we can divide through by $p(y)$ to get: $p(x|y) = p(y|x)p(x)/p(y)$.

Bayes Rule can be viewed as a way to link $p(x \mid y)$ and $p(y \mid x)$. In this case, we'll call $p(x)$ the prior, $p(y \mid x)$ the likelihood, and $p(x \mid y)$ the posterior. Now, can you compute P(Cloud=True | Rain=True) using Bayes Rule?

2 Entropy and Perplexity

Definition of Information Let E be some event with occurs with probability $P(E)$. If I told you that E occurred, then I have given you

$$I(E) = -\log_2 P(E) \tag{7}$$

bits of information. Why does this make sense to define information in terms of probabilities? Suppose you know that it almost never rains but I tell you that it just rained. That's is a valuable information. But if it always rains and I tell you it just rained, that's little information for you. One can think of information as the amount of surprise when observing E, or alternatively, the amount of bits needed to communicate about E.

More examples. How much information is in...

- the outcome of a fair coin flip? $-\log_2 1/2 = 1$ bit. I can use bit 0 to tell you that the outcome it's Head and bit 1 to tell you the outcome is Tail.
- the outcome of two fair coin flips? $-\log_2 1/4 = 2$ bits. Note that information is additive.
- the outcome of a dice roll? $-\log_2 1/6 = 2.59$ bits.
- drawing a random word from a 10000 vocab? $-\log_2 1/10000 = 13.29$ bits.

Entropy Now, suppose we have a distribution $p(x)$ with K possible values $x_1, x_2, \ldots, x_K$ in its event space. What is the **average amount of information** in observing the samples from this distribution?

$$H(p) = \sum_{k=1}^{K} P(X = x_k)I(x_k) = \sum_{k=1}^{K} p(x_k)I(x_k) = -\sum_{k=1}^{K} p(x_k)\log_2 p(x_k) \tag{8}$$

This is called entropy in information theory, and it is a function fully characterized by the distribution $p(x)$. One can think of entropy as the average amount of surprise of a distribution. Note that entropy is by definition non-negative. What kind of distribution has higher entropy: one with uniform probabilities, or one with skewed probabilities?

Cross-Entropy Let's say we don't know the true distribution (denoted $p^*(x)$) that generated our data. We have some model (denoted $p(x)$) that attempts to approximate this distribution. We want to know how good the model is. Analogous to our definition of entropy, we can answer this by asking how surprised our model is on average when tested on data generated by $p^*(x)$.

Let's say draw K random samples from $p^*(x)$. To be more precise in our notation, we will say that we have K random variables indexed by k, and each variable X_k could take some value in the event space $\{x_1, x_2, \dots, x_M\}$. For example, suppose $K = 5$ and we first drew x_2 three times, followed by x_5 once, and then x_3 once. This is the sequence of events: $X_1 = x_2$, $X_2 = x_2$, $X_3 = x_2$,$X_4 = x_5$, $X_5 = x_3$, and their probabilities are: $P^*(X_1 = x_2)$, $P^*(X_2 = x_2)$, $P^*(X_3 = x_2)$,$P^*(X_4 = x_5)$, $P^*(X_5 = x_3)$. Note that if $P^*(X = x_2) \overset{\text{def}}{=} p*(x_2)$ is high, then we should draw it many times. In other words, we can use the sequences of draws $X_1 = x_2$, $X_2 = x_2$, $X_3 = x_2$,$X_4 = x_5$, $X_5 = x_3$ to approximate the true distribution $p(x_m)$. In the limit as $K \to \infty$, we can get the true distribution.

Now we can define cross-entropy:

$$H(p^*, p) = \sum_{m=1}^{M} P^*(X = x_m) I(x_m) \approx \frac{1}{K} \sum_{k=1}^{K} I(x_m) = -\frac{1}{K} \sum_{k=1}^{K} \log_2 P(X_k = x_m) \tag{9}$$

Note the first approximation follows because we are approximating the true distribution with the K test samples. In the context of language models, this means preparing some test sentences and measuring our language model probabilities on them.

Perplexity Perplexity is simply defined as $2^{cross-entropy}$. In other words, given some test set of $x_1, x_2, \dots, x_K$ samples, the perplexity (PPL) of a model p is calculated by:

$$PPL = 2^{-\frac{1}{K}\sum_{k=1}^{K} \log_2 P(X_k = x_m)} \tag{10}$$

A nice interpretation of perplexity is that it measures the "branching factor." Cross-entropy measures uncertainty in terms of bits, but when exponentiated it measures the size of an equally weighted distribution with uniform probability. Consider an unfair dice with entropy of 2.32 bits rather than $-\log_2 1/6 = 2.59$ bits; its perplexity is $2^{2.32} \approx 5$, which means it has the same amount of information as a 5-sided dice with equal probabilities ($-\log_2 1/5 = 2.32$).

3 Language Models

We are interested in computing the probability of a sentence. This is useful for many applications in NLP. Suppose a sentence is represented by $\vec{w} = w_1 w_2 \cdots w_n$ (a sequence of n words); then by chain rule, we can calculate the probability of the sentence as the product of conditional word prediction probabilities:

$$\begin{aligned} p(\vec{w}) &= p(w_n \mid w_{n-1}, w_{n-2}, \ldots, w_1) \times p(w_{n-1} \mid w_{n-2}, \ldots, w_1) \\ &\times p(w_{n-2} \mid w_{n-3}, \ldots, w_1) \times p(w_{n-3} \mid w_{n-4}, \ldots, w_1) \\ &\times p(w_{n-4} \mid w_{n-5}, \ldots, w_1) \times \ldots \times p(w_2 \mid w_1) \times p(w_1) \end{aligned}$$

Recall in Table 1, we have the model that tries to predict the next word given some prefix. This is such a model.

How do we estimate these conditional probabilities? One way is use a training set of raw sentences and count the frequency of sub-sequences:

$$p(w_2 \mid w_1) = \frac{Count(\text{“}w_1 w_2\text{”})}{Count(\text{“}w_1\text{”})} \tag{11}$$

This is known as the **maximum likelihood estimate (MLE)** because it is a setting a probabilities that maximizes the likelihood on the training data.

However, there are some problems: for long sub-sequences, we may not have sufficient data to trust this probability estimate. One solution is to simplify the conditional probabilities.

N-gram N-gram language models make independence assumptions about the conditional probabilities, such as $p(w_3 \mid w_2, w_1) \approx p(w_3 \mid w_2)$. This is a **Markov independence assumption** that says the probability of a word only depends on its closest previous words.

A **trigram language model** defines

$$p(\vec{w}) \overset{\text{def}}{=} \prod_{i=1}^{n+1} p(w_i \mid w_{i-2}, w_{i-1}) \tag{12}$$

A **bigram language model** defines

$$p(\vec{w}) \overset{\text{def}}{=} \prod_{i=1}^{n+1} p(w_i \mid w_{i-1}) \tag{13}$$

A **unigram language model** defines

$$p(\vec{w}) \overset{\text{def}}{=} \prod_{i=1}^{n+1} p(w_i) \tag{14}$$

Usually, we would add beginning-of-sentence (`BOS`) and end-of-sentence (`EOS`) tokens[2] The index i runs to $n+1$ to account of `EOS`. For the beginning of the sentence, we will pad with `BOS` like this for bigrams $p(w_1 \mid \texttt{BOS})$ and trigrams $p(w_1 \mid \texttt{BOS}, \texttt{BOS})$.

Training and Test set At this time, we need to clarify the important distinction between training data and test data. Training data are sentences used to estimate probabilities for the language model; we extract counts from the training data to estimate probabilities like Eq. 11.

Importantly, the test set is a dataset that does not overlap with the training set. Using the model trained on the training set, we measure its cross-entropy (or perplexity) on the test set. This division is important to ensure a fair evaluation.

Sometimes, the test set may contain words that are not observed in the training data. This is known as **out-of-vocabuarly** words or unknown words, and are mapped to the `<unk>` token. There are two ways to deal with this in a n-gram language model. For a closed-vocabulary model, we assume the list of vocabulary is fixed at training time and do not generate probabilities for `<unk>`. For a open-vocabulary model, we may heuristically convert some tokens in the training set to `<unk>` and estimate its probability as if its like any other word token.

When talking about datasets, researchers often report statistics in terms of **tokens** and **types**. The number of tokens in a sentence dataset (a.k.a. **corpus**) is the total word count; the number of types measures the number of distinct words, i.e. the vocabulary size. Language model performance in terms of perplexity depends on many factors, including the size of the training corpus, its OOV rate and genre/style similarities with respect to the test corpus.

N-gram Smoothing Even with the Markov assumption, our n-gram may still have unreliable probabilities, especially for infrequent or unknown n-grams. For example, suppose we wish to estimate $p(w_i \mid "is", "name", "my")$. There are many possible person-names that can have high probability, but it is unlikely that we have all of them in this particular sequence in the training data.

The solution is to reserve probability mass for these zero-count n-grams, or to add probability mass to infrequent n-grams. Since probabilities need to normalize, we will be moving probability mass from high-count n-grams. This is called smoothing, and there is a rich class of techniques for doing so.

As an example, we will explain **Add-one smoothing** for unigrams. In standard MLE, the unigram is estimated as $p(w_i) = c_i/N$ where $c_i = Count(w_i)$ and N is the total number of tokens in the training set. Add-one smoothing will instead estimate:

[2]Sometimes BOS and EOS are used interchangeably with `<s>` and `</s>` to model words at sentence boundaries. Other times one refers to an explicit sentence boundary that is also treated as a word token while the other is used to implement stopping in the language model.

$$p_{add-one}(w_i) = \frac{c_i + 1}{N + V} \tag{15}$$

V is the vocabulary size (i.e. number of types) and is needed for normalization. This addition has the effect of increasing the probability of infrequent unigrams at the expense of the frequent ones.

N-gram Interpolation and Back-off In practice, smoothing is used in conjunction with interpolation or backoff techniques. Suppose we are building a trigram model and on't have sufficient data to estimate a particular conditional probability $p(w_i \mid w_{i-2}, w_{i-1})$. We can approximate it by a simpler model $p(w_i \mid w_{i-1})$, which ought to have more data.

In interpolation, we combine these models:

$$p_{interp}(w_i \mid w_{i-2}, w_{i-1}) = \lambda_1 p(w_i \mid w_{i-2}, w_{i-1}) + \lambda_2 p(w_i \mid w_{i-1}) \tag{16}$$

In backoff, we dynamically switch models:

$$p_{backoff}(w_i \mid w_{i-2}, w_{i-1}) = \begin{cases} p(w_i \mid w_{i-2}, w_{i-1}) & \text{if } Count(w_i, w_{i-2}, w_{i-1}) > \tau \\ \alpha(w_{i-2}, w_{i-1}) \cdot p(w_i \mid w_{i-1}) & \text{otherwise} \end{cases}$$

Note that λ and α need to be set so that the probability normalizes correctly. You are encouraged to read the textbook to learn about other smoothing methods.