Generative Models of Text Classification

3

From Random Experiments to Natural Language Processing – by Wilker Aziz. This book has not been published yet.

3.1 Data and task definition 3
Statistical task 3
NLP task 4
3.2 Tabular CPDs 5
3.3 Naive Bayes classifier 5
Parameter estimation 7
Terminology 8
Text analysis 8
Smoothing tabular cpds . . . 10

To complete this class you need to know the following:

1. From *statistics*:
   - the Categorical probability distribution, namely, its parameterisation and its probability mass function;
   - how to estimate the parameters of a Categorical distribution from data using maximum likelihood estimation.
2. From *probability theory*:
   - random variables, conditional probability, marginal probability, chain rule, conditional independence, Bayes rule.

Oftentimes we want to analyse *text* in terms of certain implicit properties of it. For example, we read a product review and try to analyse in terms of the sentiment the authors expressed towards the product. For that, we may have a categorisation of sentiments as shown in Table 3.1, and, for simplicity, we assume that the these ways of describing sentiment indeed capture different (disjoint) sentiments.

Table 3.1: One-to-one mapping

k	**sentiment**
1	clearly negative
2	somewhat negative
3	neutral
4	somewhat positive
5	clearly positive

Another example, in costumer service, we read an email request and try to route it to the department that is best qualified to act upon the request (*e.g.*, `shipment`, `payments`, `returns`). There are plenty of other examples including spam detection, document categorisation, *etc.*. For now, we will consider the case where the attribute is one of a finite set of possible categories.

3.1 Data and task definition

Our approach is data-driven, that is, we will observe a collection $\mathcal{D}$ of labelled documents, and learn a model that reproduces (aspects of) it.

An observation is a pair (x, y), where $x \in \mathcal{X}$ is a document, a sequence $x = \langle w_1, \ldots, w_l \rangle$ of $l = |x|$ words, each from a finite vocabulary $\mathcal{W} = \{1, \ldots, V\}$ of known words,[1] and y is a category in a countably finite set $\mathcal{Y} = \{1, \ldots, K\}$ of possible categories.[2] The set $\mathcal{X} = \mathcal{W}^*$ is the set of all possible sequences of known words, no matter their length (*i.e.*, the length n is anything from 0 to any natural number).

1: Assume a one-to-one correspondence between words and integers.

2: Assume a one-to-one correspondence between categories and integers.

$\mathcal{W}^*$ is not the set of all documents in a given dataset, it is a much larger set that includes any document we may ever see—as long as the words in it come from the vocabulary $\mathcal{W}$—and sequences that do not even look like plausible grammatical documents. Unlike the set of words and the set of categories, the set of all possible sequences is countably infinite (*i.e.*, it is a set of discrete elements, but there are infinitely elements in it). We do not represent this set explicitly, we just know it exists, conceptually.

Statistical task

In statistical terms, we want a mechanism that can be used to map any document $x \in \mathcal{X}$ to the probability that a certain attribute value y describes it, and we want to be able to do so for all possible values of the attribute. For example, if we read the short review `you get what you pay for` we may want to express the probability with which it conveys one of the 5 sentiment levels of Table 3.1.

A bit more formally, we want to map text $x \in \mathcal{X}$ to a conditional probability distribution $P_{Y|X=x}$ over a finite set $\mathcal{Y}$ of categories (such as those in Table 3.1). Table 3.2 illustrates two examples of conditional probability distributions (cpds) in a sentiment classification context where x is a short product review and y is one of 5 sentiment levels. These distributions assign probability to certain outcomes (*e.g.*, sentiment) given others (*e.g.*, review text), they represent and quantify our uncertainty about the mapping from one domain to another (*e.g.*, from text to one's sentiment). Statistics will help us estimate those objects, probability theory will help us design them in a compact way, and being able to do so for any document we may ever encounter is a task for an NLP researcher.

$P_{Y|X=x}$ is the probability distribution of the random variable Y conditioned on a given assignment $X = x$ of the random variable X. $P_{Y|X}(Y = y|X = x)$ denotes the probability value of an assignment $Y = y$ given an assignment $X = x$, we will often use $P_{Y|X}(y|x)$ for brevity. $P_{Y|X}$, without arguments, refers to the collection of all conditional distributions of the form $P_{Y|X=x}$, that is, for all values of X.

$X = x$	$Y = 1$	$Y = 2$	$Y = 3$	$Y = 4$	$Y = 5$
amazing characters, surprising ending	0.	0.	0.1	0.2	0.7
no depth, kinda obvious	0.5	0.3	0.2	0.	0.

Table 3.2: Each row of the table concerns a given review, each column of the table concerns a sentiment level. Each cell is the probability $P_{Y|X}(y|x)$ of a sentiment level y given the review text x—their numerical values are made up for this example. Note that every cell is a valid probability value (a number between 0 and 1) and the cells in a row add to 1.0.

NLP task

In NLP applications, the practitioner generally wants to map text x to a *single* category (*e.g.*, to decide whether or not a book should be recommended to a user, or to which department an email should be routed, or whether an email should be moved to the spam box). When we label a document with a single class we are choosing to ignore any uncertainty inherent to the mapping from document to classes. Consider a review online (e.g., on a platform such as Bol or Amazon), a person writes a review and assigns a star rating (e.g., 2 stars), but if we ask ten other people to guess the score from that piece of text, we would probably obtain a number of different guesses. Perhaps the same person would assign a different score to their own review text a day later. These thought experiments show that uncertainty in mapping from document to sentiments is a very plausible thing, yet, in many applications we are 'forced' to make a **decision** in favour of a single class.

Statistical task: map any piece of text x to a conditional probability distribution $P_{Y|X=}$ over labels $\mathcal{Y}$. *NLP task:* map any piece of text x to a single label, typically the one predicted to have highest probability under $P_{Y|X=x}$.

To address both the statistical and the NLP task, we are going to make a convenient separation. We will rely on statistics to approximate the mapping from x to a probability distribution $P_{Y|X=x}$, and we will rely on some other strategy to map from $P_{Y|X=x}$ to a single class in $\mathcal{Y}$. This strategy is what we call a **decision rule**, and there are a few available. By and large, the most common decision rule in text classification is the *most probable class* rule, which predicts the class assigned highest probability under the model (the so-called *mode* of the distribution):

$$y^\star = \arg\max_{k \in \mathcal{Y}} P_{Y|X}(k|x) \,. \tag{3.1}$$

The operator $\arg\max_{e \in \mathcal{E}} f(e)$ returns the element in the set $\mathcal{E}$ for which the function $f(e)$ attains highest value. In general, this value is not unique, but we will mostly act as if it were. Example: if $f(x) = 0.7^x \times 0.3^{1-x}$, then $\arg\max_{x \in \{0,1\}} f(x) = 1$ because $f(1) > f(0)$.

3.2 Tabular CPDs

Statistically, we are interested in realising the mapping $x \mapsto P_{Y|X=x}$. If $\mathcal{X}$ were finite, we could hope to associate a Categorical distribution with each and every one of the possible assignments of X, and independently specify each and every such distribution.

The notation $C \sim \text{Cat}(\theta_1, \ldots, \theta_K)$ is pronounced 'C is a random variable with a Categorical distribution'. A Categorical distribution over K categories is prescribed by K probability values $\theta_1, \ldots, \theta_K$, one for each class. To be valid, they must be positive (*i.e.*, $\theta_k \geq 0$) and add to 1 (*i.e.*, $\sum_{k=1}^{K} \theta_k = 1$). The Categorical probability mass function (pmf) assigns probability $\text{Cat}(k|\theta_1, \ldots, \theta_K) = \theta_k$ to $C = k$.

That is, we could model $Y|X = x \sim \text{Cat}(\boldsymbol{\theta}^{(x)})$ using a Categorical distribution with a parameter $\boldsymbol{\theta}^{(x)} \in \Delta_{K-1}$ that is specific to the text x. We could then use *maximum likelihood estimation* (MLE) to determine $\theta_o^{(c)}$ for every conditioning context $c \in \mathcal{X}$ and every outcome $o \in \mathcal{Y}$ using a dataset of N observations $\mathcal{D} = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$. The MLE solution is:

$$\theta_o^{(c)} = \frac{\sum_{n=1}^{N} [x^{(n)} = c] \times [y^{(n)} = o]}{\sum_{k \in \mathcal{Y}} \sum_{n=1}^{N} [x^{(n)} = c] \times [y^{(n)} = k]}\,, \tag{3.2}$$

We use $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_K)^\top$ to denote a fixed-dimensional (column) vector. When we need to 'name' this vector or associate it with a specific context, we use a superscript: *e.g.*, $\boldsymbol{\theta}^{(\text{pretty good})}$) is the vector associated with the document 'pretty good'.

The *probability simplex*, denoted Δ_{K-1}, is a subset of $\mathbb{R}^K_{\geq 0}$. If $\boldsymbol{\theta} \in \Delta_{K-1}$, then $0 \leq \theta_k \leq 1$ for any $k \in \{1, \ldots, K\}$, and $\sum_{k=1}^{K} \theta_k = 1$. Think of it as the space of K-dimensional 'probability vectors'.

which only requires counting the number of times $(X = c, Y = o)$ co-occur in the data and divide that by the marginal counts for $X = c$ (*i.e.*, the number of times $X = c$ occurred, or, equivalently, the number of times it co-occurred with anyone of the possible outcomes of Y). This way to represent a conditional probability distribution (cpd) is called a *tabular representation* of the cpd. It is called that way because we can store all probability values $P_{Y|X}(y|x) = \theta_y^{(x)}$ in a table, where each row is the probability vector $\boldsymbol{\theta}^{(x)}$ that prescribes the cpd $P_{Y|X=x}$. See Table 3.3. The difficulty in using such an approach is that in text classification x is a whole sentence, paragraph, or document, drawn from an unbounded (or, more formally, countably infinite) space of outcomes. To be able to map any x to a distribution $P_{Y|X=x}$, we would have to store and estimate as many parameters as there are pairs (x, y) in the space $\mathcal{X} \times \mathcal{Y}$. First,

The Iverson bracket $[\alpha]$ converts a Boolean expression to a real number: it is 1 when the predicate α is true, and 0 otherwise. For example, if $x = \langle \text{it, is, pretty, good} \rangle$, $[x_3 = \text{pretty}] = 1$ and $[x_2 = \text{are}] = 0$.

$x \in \mathcal{X}$	$\theta_1^{(x)}$	$\theta_2^{(x)}$	$\theta_3^{(x)}$	$\theta_4^{(x)}$	$\theta_5^{(x)}$
amazing characters, surprising ending	0.	0.	0.1	0.2	0.7
no depth, kinda obvious	0.5	0.3	0.2	0.	0.

Table 3.3: Two examples of tabular cpds for sentiment classification. Conceptually, to complete this table, we need infinitely many rows, one for each possible document $x \in \mathcal{X}$.

we cannot store infinitely many parameters. Second, we cannot really estimate infinitely many parameters from data, as in practice we will only observe a finite number of labelled documents.

In a tabular representation of a collections of cpds $P_{Y|X}$, the probability value $P_{Y|X}(y|x)$ of known pairs (x, y) are stored in a table. Pairs that were never seen before are considered outside the support of the distribution, and thus get 0 probability.

Our first attempt at text classification will be constrained to such tabular representations of cpds, but, for viable solutions, we will have to make a more efficient use of this technique.

3.3 Naive Bayes classifier

Instead of storing the K probability values that prescribe the cpd $P_{Y|X=x}$ in a table, the *naive Bayes classifier* (NBC) *infers* them for any given $x \in \mathcal{X}$ using a compact joint distribution over X and Y and Bayes rule. To see how this happens, let's use some probability calculus to rewrite the probability $P_{Y|X}(y|x)$ in terms of other (possibly interesting) quantities.

We begin with the definition of conditional probability

$$P_{Y|X}(y|x) = \frac{P_{YX}(y,x)}{P_X(x)} \,, \tag{3.3a}$$

and apply chain rule to factorise the joint probability in the numerator

$$= \frac{P_Y(y)P_{X|Y}(x|y)}{P_X(x)} \,. \tag{3.3b}$$

When doing so, we conveniently start with the class probability $P_Y(y)$, which allows us to introduce the factor $P_{X|Y}(x|y)$ in the expression. This factor is different from $P_{Y|X}(y|x)$ in that we are *generating* text, rather than conditioning on it. While there are infinitely many cpds of the kind $P_{Y|X=x}$, one for each possible document of interest, there are only K cpds of the kind $P_{X|Y=y}$, one for each label.

For any one label y, the cpd $P_{X|Y=y}$ is not trivial either, since there is no plausible way to bound the sample space of X. When generating a high-dimensional data structure (*e.g.*, a document), we can choose to make some simplifying assumptions, even *naive* ones, as far as we are willing to live with the consequences (*i.e.*, they limit the analysis in some way). Let's assume that words in x are independent of one another given the class y. Under such *conditional independence assumption* it holds that:

This conditional independence assumption is not realistic. Put yourself on the shoes of a reviewer, you decide you will write a negative review, then you write it, the words you produce are not independent of one another, they likely form a coherent argument. But this assumption is *needed* at this point, we motivate it from no more than convenience.

$$P_{X|Y}(x|y) \overset{\text{ind.}}{=} \prod_{i=1}^{l} P_{W|Y}(w_i|y) \,, \tag{3.4a}$$

We sometimes write something on top of the 'equals to' operator (*e.g.*, $\overset{\text{ind.}}{=}$). This is meant to indicate that the equality holds because of a certain assumption (as opposed to it holding in general). For example, the statement $P_{AB}(a,b) = P_A(a)P_{B|A}(b|a) = P_B(b)P_{A|B}(a|b)$ holds in general (it is the chain rule of probabilities), whereas $P_{AB}(a,b) \overset{\text{ind.}}{=} P_A(a)P_B(b)$ only holds when the two random variables are independent of one another.

which we can use to rewrite the conditional further:

$$P_{Y|X}(y|x) \overset{\text{ind.}}{=} \frac{P_Y(y)\prod_{i=1}^{l} P_{W|Y}(w_i|y)}{P_X(x)} \,. \tag{3.4b}$$

This can be further rewritten to express the denominator using the same types of factors used in the numerator:

$$= \frac{P_Y(y)\prod_{i=1}^{l} P_{W|Y}(w_i|y)}{\sum_{k\in\mathcal{Y}} P_{YX}(k,x)} \tag{3.4c}$$

$$= \frac{P_Y(y)\prod_{i=1}^{l} P_{W|Y}(w_i|y)}{\sum_{k\in\mathcal{Y}} P_Y(k)P_{X|Y}(x|k)} \tag{3.4d}$$

$$= \frac{P_Y(y)\prod_{i=1}^{l} P_{W|Y}(w_i|y)}{\sum_{k\in\mathcal{Y}} P_Y(k)\prod_{i=1}^{l} P_{W|Y}(w_i|k)} \,. \tag{3.4e}$$

What we just did was to infer $P_{Y|X}(y|x)$, which conditions on a high-dimensional outcome x and thus cannot be represented efficiently using tabular cpds, from a joint distribution P_{YX} that factorises in terms of simple cpds. In fact, we only have two types of cpds in the final expression: i) one unconditional distribution over classes P_Y; and ii) a set of class-conditioned distributions over words, each of the kind $P_{W|Y=y}$. Both $\mathcal{Y}$ and $\mathcal{W}$ are finite and small enough that we could represent the collection of all cpds using tables. A tabular Categorical distribution $Y \sim \text{Cat}(\boldsymbol{\phi})$ with $\boldsymbol{\phi} \in \Delta_{K-1}$, and K tabular Categorical distributions

$W|Y = y \sim \text{Cat}(\boldsymbol{\pi}^{(y)})$ with $\boldsymbol{\pi}^{(y)} \in \Delta_{V-1}$, one for each $y \in \mathcal{Y}$. Table 3.4 and Table 3.5 illustrate what we achieved. Compare them to Table 3.3.

$Y=1$	$Y=2$	$Y=3$	$Y=4$	$Y=5$
ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5

Table 3.4: Prior over 5 classes.

$Y=y$	$W=1$	$W=2$	...	$W=V$
1	$\pi_1^{(1)}$	$\pi_2^{(1)}$	...	$\pi_V^{(1)}$
2	$\pi_1^{(2)}$	$\pi_2^{(2)}$	...	$\pi_V^{(2)}$
3	$\pi_1^{(3)}$	$\pi_2^{(3)}$	...	$\pi_V^{(3)}$
4	$\pi_1^{(4)}$	$\pi_2^{(4)}$	...	$\pi_V^{(4)}$
5	$\pi_1^{(5)}$	$\pi_2^{(5)}$	...	$\pi_V^{(5)}$

Table 3.5: The parameters of the class-conditioned cpds. There are 5 such cpds, one per value of the sentiment label. Each cpd takes V parameters, one for each of the V known words in the language (assume words were mapped to integers 1 to V).

Parameter estimation

The parameters of the various cpds in this model—shown in Tables 3.4 and 3.5—can be estimated via MLE:

$$\phi_k = \frac{\sum_{n=1}^{N}[y^{(n)} = k]}{N}\,, \tag{3.5a}$$

$$\pi_v^{(k)} = \frac{\sum_{n=1}^{N}\sum_{i=1}^{l_n}[y^{(n)} = k] \times [w_i^{(n)} = v]}{\sum_{o \in \mathcal{X}}\sum_{n=1}^{N}\sum_{i=1}^{l_n}[y^{(n)} = k] \times [w_i^{(n)} = o]}\,, \tag{3.5b}$$

where we use l_n as the length of the nth document (*i.e.*, $l_n = |x^{(n)}|$).

Take a moment to understand these expressions. For the prior parameter ϕ_k, which prescribes the probability $P_Y(k)$ of a class assignment $Y = k$, we need the sample frequency of $Y = k$ in the dataset: we iterate over each of the N data points, count how many times $y^{(n)}$ is exactly k, and divide by the total number of data points N. We need to do that for every possible value of k. For the class-conditioned parameter $\pi_v^{(k)}$, which prescribes the probability $P_{W|Y}(v|k)$ of a word assignment $W = v$ given a class assignment $Y = k$, we need the sample frequency of the pair $(Y = k, W = v)$ in the subset of observed documents that contains all instances labelled with class k: we iterate over each of the N data points (*i.e.*, $n = 1, \ldots, N$) and over the positions of each text (*i.e.*, $i = 1, \ldots, l_n$) counting how many times $y^{(n)}$ is exactly k at the same time that $w_i^{(n)}$ is exactly v, divide by the number of times $y^{(n)}$ is exactly k regardless of the value of $w_i^{(n)}$. The denominator in that case is in fact equal to $\sum_{n=1}^{N} l_n \times [y^{(n)} = k]$: the total number of words in the subset of documents labelled as instances of class k.

The notation above can be daunting to read. So, for clarity, we will define a notation shortcut. We define some auxiliary functions to count joint occurrences of outcomes in the data. For example, with

$$\text{count}_Y(k) = \sum_{n=1}^{N}[y^{(n)} = k] \tag{3.6a}$$

$$\text{count}_{YW}(k, v) = \sum_{n=1}^{N}\sum_{i=1}^{l_n}[y^{(n)} = k] \times [w_i^{(n)} = v] \tag{3.6b}$$

we can re-write Equations 3.5a and 3.5b as:

$$\phi_k = \frac{\text{count}_Y(k)}{\sum_{y \in \mathcal{Y}} \text{count}_Y(y)} \tag{3.7a}$$

$$\pi_v^{(k)} = \frac{\text{count}_{YW}(k, v)}{\sum_{w \in \mathcal{W}} \text{count}_{YW}(k, w)}\,. \tag{3.7b}$$

For clarity, we subscript these functions with the random variables whose outcomes we are counting.

Time and space. If you look carefully, the essential operation needed for the MLE solution is to iterate over all labelled documents counting something in memory (*i.e.*, occurrences of a given class for the prior, co-occurrences of classes and words for the class-conditioned cpds). If L is the length of the longest document in the collection, the worst case time complexity of the MLE procedure is $\mathcal{O}(N \times L)$, for gathering the co-occurrence counts takes iterating over all positions ($i = 1, \ldots, L$) of all documents in the dataset ($n = 1, \ldots, N$). Without any optimisation for sparse tables, and assuming that each parameter takes one unit of space, the entire model occupies $\mathcal{O}(K + K \times V) = \mathcal{O}(K \times V)$ units of space—one per parameter of the prior and one per parameter of each class-conditioned cpd.

Terminology

For a given pair (x, y), the quantity $P_Y(y)$ is known as the *prior probability* of y (called that way because it is the probability a class receives before we know the document); the quantity $P_{X|Y}(x|y)$ is known as the *likelihood of y given x*;[3] which makes the quantity $P_{Y|X}(y|x)$ the *posterior probability* of y given x. The posterior probability expresses our revised belief about y upon observing x.

3: You read it correctly. $P_{X|Y}(x|y)$ is the probability of a choice of x given some fixed y, but, when we talk about that same quantity as a function of some choice of y for a fixed x, we call it the likelihood of y given x. In machine learning and NLP literature, the *likelihood of y given x* is often shortened to the *likelihood of y* or even just *likelihood*, since x is fixed. In recent papers, you will encounter the expression 'the likelihood of the document', which is a rather informal (if not incorrect) way of referring to the likelihood of the class given the document.

The independence assumption we make, namely, that words in x are conditionally independent of one another given the class y is known as the 'bag of words' assumption, or 'bag of words' representation of the document. This term *bag of words* comes from thinking of a document in the following way: transcribe x to a piece of paper, cut this piece of paper into different parts separating the words in it, then put all these little pieces of paper in a bag and shuffle it. The view of the document you get retains no information about the order in which words occurred, all information it retains is which words occurred in x and how many times they occurred. When we think of a document as a bag of words, rather than a sequence of tokens with arbitrary length, we can use a V-dimensional vector of counts to represent the document, which, depending on how long we expect our documents to be, can be convenient for implementation using a programming language.

The count vector $\mathbf{c} \in \mathbb{N}_0^V$ defined such that $c_v = \sum_{i=1}^{l} [w_i = v]$ is the number of times word $v \in \mathcal{W}$ occurs in document $x = \langle w_1, \ldots, w_l \rangle$ is known as the *bag of words encoding* of the document. Note that by the definition of $\mathbf{c}$, $\sum_{v=1}^{V} c_v = l$.

Text analysis

When we analyse a novel document $x_* = \langle w_1, \ldots, w_l \rangle$, we may be interested in the determining the cpd $P_{Y|X=x_*}$ and/or one of its properties.

Posterior distribution. Let's start with inferring the entire posterior cpd, that is, inferring $P_{Y|X}(k|x_*)$ for every value $k \in \mathcal{Y}$. Recall that by Bayes rule we know that:

$$P_{Y|X}(k|x_*) = \frac{P_Y(k) \prod_{i=1}^{l} P_{W|Y}(w_i|k)}{\sum_{y \in \mathcal{Y}} P_Y(y) \prod_{i=1}^{l} P_{W|Y}(w_i|y)} \tag{3.8}$$

$$= \frac{\phi_k \times \prod_{i=1}^{l} \pi_{w_i}^{(k)}}{\sum_{y \in \mathcal{Y}} \phi_y \times \prod_{i=1}^{l} \pi_{w_i}^{(y)}} \,. \tag{3.9}$$

The elementary probability factors in this computation are parameters stored in our tabular cpds. Note that the term in the numerator, as well as each of the terms inside the sum in the denominator, requires multiplying $|x_*| + 1$ probability values together. If assessing each elementary probability value takes one unit of time, denoted $\mathcal{O}(1)$, then each of those terms takes $|x_*| + 1$ units of time, denoted $\mathcal{O}(|x_*|)$. We have to compute only one such term in the numerator, but we need to compute K such terms in the denominator (because we need to marginalise the class assignment out), wich leads to the overall procedure costing time proportional to $\mathcal{O}(K \times |x_*|)$.

The naive Bayes model uses joint distributions, conditional independence, and Bayes rule to essentially trade storage for computation (*i.e.*, we cannot store and estimate infinitely many parameters, but we can store and estimate $K + K \times V$ parameters). Rather than storing the distribution $P_{Y|X=x}$ for every x, we compute it (infer it) from the elementary ones only when it is needed.

Posterior mode. Sometimes all we care about is the most probable class under the model. In those cases, we do not need to normalise the distribution. That's true because

$$y^\star = \arg\max_{k \in \mathcal{Y}} \; P_{Y|X}(k|x_*) \tag{3.10a}$$

the conditional probability can be rewritten

$$= \arg\max_{k \in \mathcal{Y}} \; \frac{P_{YX}(k, x_*)}{P_X(x_*)} \tag{3.10b}$$

and its denominator is not a function of our choice of class k

$$= \arg\max_{k \in \mathcal{Y}} \; P_{YX}(k, x_*) \tag{3.10c}$$

and thus can be dropped, as it does not affect the argmax:

$$= \arg\max_{k \in \mathcal{Y}} \; P_Y(k) \prod_{i=1}^{l} P_{W|Y}(w_i|k) \tag{3.10d}$$

$$= \arg\max_{k \in \mathcal{Y}} \; \phi_k \prod_{i=1}^{l} \pi_{w_i}^{(k)} \,. \tag{3.10e}$$

Determining which of the available classes is the mode of the distribution still requires evaluating all K joint probabilities, and thus it still takes time $\mathcal{O}(K \times |x_*|)$.

Marginal probability. When we expressed the posterior cpd, we also expressed an interesting quantity, but perhaps we did not give it much attention. The *marginal probability* of the document x_* is the quantity that

normalises the joint distribution into a posterior cpd:

$$P_X(x_*) = \sum_{k \in \mathcal{Y}} P_{YX}(k, x_*) \tag{3.11a}$$

$$= \sum_{k \in \mathcal{Y}} P_Y(k) \prod_{i=1}^{|x_*|} P_{W|Y}(w_i|k) \tag{3.11b}$$

$$= \sum_{k \in \mathcal{Y}} \phi_k \times \prod_{i=1}^{|x_*|} \pi_{w_i}^{(k)} \,. \tag{3.11c}$$

The marginal probability plays an important role when learning from incomplete datasets (*e.g.*, datasets that miss annotation for some of the documents), a problem commonly referred to as *semi-supervised learning*. For now, we will not study semi-supervised learning, but soon we will encounter other applications of marginal probabilities.

Smoothing tabular cpds

Oftentimes, when analysing novel documents, we will encounter words that we have never seen before. Tabular cpds cannot accommodate outcomes that were never seen before, they are effectively outside the support of the model and receive 0 probability.

In an attempt to overcome this data sparsity problem, it is common to *smooth* the maximum likelihood estimates of the class-conditioned distributions $P_{W|Y}$ stealing some probability mass from outcomes that were observed and reserving it to future outcomes.

The simplest such technique is called *Laplace smoothing* or 'add α smoothing'. The idea is to change MLE as follows:

$$\pi_w^{(k)} = \frac{\alpha + \text{count}_{YW}(k, w)}{V\alpha + \sum_{o \in \mathcal{W}} \text{count}_{YW}(k, o)} \,. \tag{3.12}$$

The value of α cannot be estimated from the training data alone, and, instead, needs to be fixed manually using an assessment of performance on heldout data (see sections 4.7 and 4.8 of [1] for evaluation and cross validation).

[1]: Jurafsky et al. (2023), *Speech and Language Processing*

When, we are analysing novel documents, the probability of some unknown word $W = u$ given $Y = k$ is therefore set to $\frac{\alpha}{V\alpha + \text{count}_Y(k)}$.