lectures 6 and 7

1 Document Mining

Our next topic will take us to seemingly very different data spaces - those of textual documents. We will outline how to perform analysis in such spaces. Our key step will be representation of documents as (potentially very high-dimensional) vectors. This will be done through identifying co-ordinates with specific words or phrases e.g. "equation", "solution", "sun" etc. and specifying for each document the value ('strength') each co-ordinate takes for that document.

Let $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$ be our document set and $\mathcal{T} = \{t_1, t_2, \ldots, t_T\}$ be the set of terms. In a realistic scenario, $|\mathcal{T}| = T$ can be a large number and we will need to pick the relevant terms that best represent the variety of documents in our dataset.

We want to transform a document d_j into a vector:

$$d_i \mapsto \mathbf{d}^i = (x_{i,1}, x_{i,2}, \ldots, x_{i,T})^T \in \mathbb{R}^T$$

Each $x_{i,k} \geq 0$ is a 'weight' corresponding to the term t_k, describing how 'important' the term is in the document d_i. How should we calculate this weight? We propose a couple of ideas:

i) $x_{i,k}$ is binary - is the term in the document or not? This is a bit simple, but can be efficient to calculate.

ii) $x_{i,k}$ represents the frequency of the term in the document, denoted $N_{i,k}$. This is better, but shorter and longer documents will have values of $x_{i,k}$ that are not directly comparable.

iii) $x_{i,k} = \frac{N_{i,k}}{|d_i|}$ is the relative frequency of the term, where $|d_i|$ denotes the length of the document. The undesirable influence of the document length on $x_{i,k}$ is now diminished.

Finally we realize that, of course, if a term is frequent in a given document, than it is important for it. But what if that term is also frequent in almost all other documents in our dataset (library)? Intuitively, in such a case the term would not characterize the document very well - it will not make the document "stand out" among the other documents in the library. Hence, we would like to assign a term high weight for a given document, if it is *both* frequent in it *and* it makes the document unique - e.g. it occurs sparsely across other documents. We have the following observations:

1) The more frequent term t_k is in document d_i, the more it is representative of its content;

2) The more documents t_k occurs in, the 'less discriminating' it is.

We can quantify 1) by using the *relative frequency* (probability) of the term in the document,

$$\frac{N_{i,k}}{|d_i|} = Pr(t_k \mid d_i) = P(t_k). \tag{1}$$

For 2), we need to know the proportion of documents in the document set (library) $\mathcal{D}$ that contain t_k. This can easily be computed by

$$\frac{N_k}{N} = Pr(t_k \mid \mathcal{D}) = Q(t_k), \tag{2}$$

where N_k is the number of documents that contain t_k and $N = |\mathcal{D}|$ is the library size. Note that the probability distributions P and Q are both over the same term set $\mathcal{T}$. We say that they share the same support. Combining (1) and (2), we can find an appropriate way to compute $x_{i,k}$:

$$x_{i,k} = P(t_k) \cdot \log_2 \frac{1}{Q(t_k)} = \frac{N_{i,k}}{|d_j|} \cdot \log_2 \frac{N}{N_k}. \tag{3}$$

This method is known as *term frequency inverse document frequency*, or TFIDF for short. Indeed, for term t_k to have high importance in document d_i, it needs to be frequent in d_i (high term frequency $P(t_k)$) *and* infrequent in the library (low $Q(t_k)$, high inverse document frequency). Intuitively, the log function squashes high values of $1/Q(t_k)$, as it really does not matter, provided the term is quite unique in the library, whether $Q(t_k)$ is as small as 0.001, or 0.0001... We will now justify the TFIDF expression in the framework of Information Theory.

2 Information theoretic point of view

Suppose Alice has a 4-sided *fair* dice and wants to relay what value was landed on the dice to Bob. She cannot do this directly as Bob is overseas, but she can speak to him through some communication channel that accepts binary messages. We can use the following coding scheme to represent each singular event when we transmit it through the channel:

$$\begin{aligned} \text{Landing a one} &- 00 \\ \text{Landing a Two} &- 01 \\ \text{Landing a Three} &- 10 \\ \text{Landing a Four} &- 11 \end{aligned}$$

Note that in this case there seems to be a relationship between the number of bits needed and the probability of the possible events that can occur (i.e. $\frac{1}{4}$). In particular,

$$\log_2 \frac{1}{\frac{1}{4}} = \log_2 4 = 2 \text{ bits}.$$

Using similar reasoning, if Alice used an 8-sided fair dice, then she would need 3 bits of information to describe all eight possible events. Again,

$$\log_2 \frac{1}{\frac{1}{8}} = \log_2 8 = 3 \text{ bits.}$$

In the general case (not uniform distribution), consider a random variable X and the associated probability $P(x)$ of an event x. If $P(x)$ is high and x actually happens, we are not surprised to see x happening and hence do not learn much from realizing that x was observed. On the other hand, if $P(x)$ is small and x happens, we are surprised and consider it a useful bit of information. Hence, the information (or the amount of statistical surprise) in the event x is inversely proportional to its probability of occurrence, $\frac{1}{P(x)}$. It can be shown that, like in the examples above, that the optimal message length in bits to use in order to convey the information that x happened is

$$\log_2 \frac{1}{P(x)} \text{ bits.} \tag{4}$$

We now draw many realizations from the random variable X and each time pass the information about the observed event through a communication channel using $\log_2 \frac{1}{P(x)}$ bits. What is the average number of bits per realization when coding all of the events? This is just the expected value of (4):

Definition 2.1. For a random variable X distributed with $P(x)$, let A be the event set. The *entropy* of X is defined as

$$H(X) = \mathbb{E}\left[\log_2 \frac{1}{P(x)}\right] = \sum_{x \in A} P(x) \cdot \log_2 \frac{1}{P(x)}.$$

Now consider the following scenario: Suppose that for a random variable X, distributed with $P(x)$, when designing a code for the events $x \in A$, we wrongly assume X is distributed according to a probability distribution $Q(x)$, $Q \neq P$. Hence each event $x \in A$ will be coded with a message of length

$$\log_2 \frac{1}{Q(x)} \text{ bits.} \tag{5}$$

But the events are actually generated with probability P, meaning that our average code length is now the expected value of $\log_2 \frac{1}{Q(x)}$ with respect to the distribution P:

Definition 2.2. [1] Let P and Q be probability distributions over the event set A. The *cross-entropy* from P to Q is defined as

$$H^c(P, Q) = \sum_{x \in A} P(x) \cdot \log_2 \frac{1}{Q(x)}. \tag{6}$$

[1] see `https://en.wikipedia.org/wiki/Cross_entropy` for details

Of course, since we built the code using a wrong distribution Q (instead of P), the average code length will be higher:

$$H^c(P,Q) \geq H(P),$$

with equality, if (and only if) $P = Q$. The value of $H^c(P,Q)$ thus reflects (quantifies) how different are the two distributions P and Q. This difference is sometimes expressed as a normalized quantity, i.e. the number of *additional* bits we need to use as a consequence of mis-specifying Q for P: $H^c(P,Q) - H(P)$. This quantity is known as K-L divergence from P to Q:

$$\begin{aligned} D_{KL}[P||Q] &= \mathbb{E}_P\left[\log_2 \frac{1}{Q(x)}\right] - \mathbb{E}_P\left[\log_2 \frac{1}{P(x)}\right] \\ &= \sum_{x\in A} P(x)\cdot \log_2\left(\frac{1}{Q(x)}\right) - \sum_{x\in A} P(x)\cdot \log_2\left(\frac{1}{P(x)}\right) \\ &= \sum_{x\in A} P(x)\cdot \left[\log_2\left(\frac{1}{Q(x)}\right) + \log_2 P(x)\right] \\ &= \sum_{x\in A} P(x)\cdot \log_2\left(\frac{P(x)}{Q(x)}\right). \end{aligned}$$

Using (1), (2) and comparing the TFIDF expression (3) with (6), we see that TFIDF weights can simply be viewed as contributions to the cross-entropy from the term distribution in the particular document to the term distribution across the whole library.