MPEI — summary

Paulo J S G Ferreira

4th November 2014

Contents

1 Fundamentals

1.1 What definition?

The probability of an event is a number in the interval $[0, 1]$ that expresses "how likely" the event is to occur. The values 0 and 1 denote impossibility and absolute certainty, respectively.

How to best define the probability of an event? One way is to consider n identical experiments that may or may not lead to the event. Let m be the number of times that the event is observed (its *absolute frequency*). The limit $n \to \infty$ of the *relative frequency* m/n, if it exists, would be the probability of the event.

It is often better to avoid the limit and even the need to perform "n experiments". We could define probability of an event as the ratio m/n, where m is the number of "favourable cases" (the number of ways in which the event can occur) and n the "total number of cases" (the total number of possibilities). However, one needs to assume that the possibilities are equally likely.

There is a more general method. The events are regarded as subsets of a set U, that represents the universe of possibilities (also called the sample set) and the probability of each event is determined by the following set of axioms or principles.

1.2 Principles

Let U be the (non-empty) sample set. The probability $P(A)$ of any event $A \subset U$ satisfies (i) $P(A) \geq 0$ (ii) $P(U) = 1$ (iii) the probability of the union[1] of disjoint events is the sum of the probabilities:

$$\boxed{P(A + B) \;=\; P(A) + P(B)} \qquad \text{if } AB = \emptyset.$$

In general, one has the inequality $P(A + B) \leq P(A) + P(B)$.

In order to determine the probability of the intersection, union and complement of any event it is necessary that the set U be closed under those operations, finite or infinite. Given the events $A_1, A_2, \ldots$ of U, it is sufficient to assume that their union and intersection is in U.

It is a consequence of these principles that the empty set $\emptyset$ in is U and has zero probability, because if $A \in U$ then $A^c \in U$ as well and $AA^c = \emptyset$. The probability of the empty set is zero because $P(A) = P(A + \emptyset) = P(A) + P(\emptyset)$. The same conclusion would follow from $P(A) + P(A^c) = 1$ by setting $A = U$, $A^c = \emptyset$.

1.3 Independence

Two events are independent if information about the occurrence of one event does not permit any inference about the occurrence of the other. A frequency analysis suggests the definition

$$\boxed{P(AB) \;=\; P(A)P(B)} \qquad A,\ B \text{ independent.}$$

[1] I denote the union of A and B by $A+B$ or $A \cup B$ and their intersection by AB or $A \cap B$. The complement of A is represented by A^c or $\bar{A}$.

The conditional probability of A given B, denoted by $P(A|B)$, must reduce to $P(A)$ if A and B are independent. A frequency-based argument suggests the definition[2]

$$P(A|B) = \frac{P(AB)}{P(B)}.$$

If A and B are independent one has $P(AB) = P(A)P(B)$ and so $P(A|B) = P(A)$, as could be expected.

1.4 Information

Probability and information are related concepts. When the occurrence of an event is communicated, the amount of information transferred depends on the probability of the event. Thus, information should be written as a function of the probability, $I(p)$. If the event is certain no information is exchanged, hence $I(1) = 0$. The more unlikely the event is, the greater the associated information. In other words, $I(p)$ should increase as p decreases.

The information associated with two independent events should be the sum of the information separately assigned to each event. Since the probability of occurrence of two independent events, with individual probabilities p and q, is given by the product pq, the information should satisfy

$$I(pq) = I(p) + I(q).$$

The logarithm satisfies $\log pq = \log p + \log q$, which seems to suggest $I(p) = \log p$. However, this function decreases with p. To fix this, consider its negative, that is,

$$I(p) = \log \frac{1}{p}.$$

This function satisfies $I(1) = 0$ and increases as p decreases, regardless of the base of the logarithm. The information associated with independent events A e B, with joint probability pq, is

$$\begin{aligned} I(pq) &= \log \frac{1}{pq} \\ &= \log \frac{1}{p} + \log \frac{1}{q} \\ &= I(p) + I(q), \end{aligned}$$

as it should. Different logarithm bases lead to different information units. When base 2 logarithms are used, the unit of information is called *bit*.

Consider an experiment with two equally likely outcomes (therefore, each has probability $1/2$). The amount of information that is transferred to communicate the result of the experiment is, in bits,

$$I(1/2) = \log_2 \frac{1}{1/2} = \log_2 2 = 1 \text{ (bit)}.$$

Note that the storage of 1 bit of information does not always require 1 bit of memory.

[2] Since B occurs it makes sense to assume $P(B) \neq 0$, as the definition suggests.

1.5 Random variables

A random variable (or simply a "variable", if there is no ambiguity) assigns to each event a real number. This enables the replacement of non-numeric results such as heads and tails by real numbers (0 and 1, for example). Usually, random variables are represented by upper case letters, such as X.

The replacement of events by real numbers enables the use of expressions such as $P(X \leq 1)$ or $P(a < X \leq b)$. Since X takes real values, expressions such as $X \leq 1$ or $a < X \leq b$ are meaningful and determine subsets of the reals.

A variable that takes values from a finite or countable set of values is called *discrete*; otherwise, it is called *continuous*.

1.6 Mean value

The average (or mean value, or expected value) is a number that provides information about the "magnitude" of a set of numbers. The sentences "the student A has an average score of 15" and "the student B has an average score of 18" suggest that A has "lower grades" than B — although no specific information has been provided on specific grades. The average "summarises" the data.

Before defining the average of a discrete random variable X it is helpful to consider an example. A student with scores $12, 18, 15, 15$ and 18 has the arithmetic average

$$\begin{aligned} \frac{12+18+15+15+18}{5} &= \frac{12+2\times 15+2\times 18}{5} \\ &= 12\,\frac{1}{5} + 15\,\frac{2}{5} + 18\,\frac{2}{5}. \end{aligned}$$

Note that $1/5$ and $2/5$ are nothing but the relative frequencies of the grades. In general, if the student has N_k grades k out of a total of D courses, the average μ will be

$$\mu = \frac{1}{D}\sum_{k=0}^{20} k\, N_k,$$

that is,

$$\mu = \sum_{k=0}^{20} k\, f_k,$$

in which f_k is the relative frequency of grade k, $f_k = N_k/D$.

By identifying the relative frequencies with probabilities, this suggests the following definition of the average of a discrete random variable X,

$$\mu = \sum_k k\, p(k),$$

where $p(k)$ concisely expresses the probability of X taking the value k, that is, $P(X = k)$.

The following notation is also used to denote the average:

$$\mu = E[X]$$

The symbol E suggests "expected value". This notation is convenient to express certain properties of the average (for example, if X and Y are random variables, one has $E[X+Y] = E[X] + E[Y]$, and for any real α one has $E[\alpha X] = \alpha E[X]$ — as will be seen).

1.7 Variance

The variance of a random variable measures its concentration around the mean value. The sets $\{14, 15, 16\}$ and $\{10, 15, 20\}$ have the same arithmetic average, but the first is more "concentrated" around the average than the second. The concept of variance allows this difference to be quantified.

To motivate the definition, consider a data set x_i with average μ. To measure how concentrated its values are around the average, we start by evaluating the difference between each data and the average, that is, $x_i - \mu$. If these values are "small" then we expect the variance to be small. The absolute value of the difference $|x_i - \mu|$ is a natural measure of the "size" of $x_i - \mu$, insensitive to its algebraic sign. The same happens with the square of the difference, $(x_i - \mu)^2$, which has an important advantage: the function "square" is easier to handle than the function "absolute value". These considerations suggest the following definition.

The variance of the random variable X, denoted by $\sigma^2(X)$, σ_X^2, or even σ^2, is the average value of the square of its deviation to the average, that is:

$$\sigma^2 = E[(X - \mu)^2].$$

By expanding the square we find that

$$\sigma^2 = E[X^2] - \mu^2,$$

that is, the variance is the difference between the average of the squares and the square of the average. This expression is sometimes more convenient in computations. Note that the variance of a random variable with zero average is simply $E[X^2]$.

The square root of the variance, denoted by σ, is called "standard deviation".

1.8 Sum of independent variables

Just as the square of a sum contains products of cross terms, the variance of a sum contains terms similar to $E[XY]$. It is important to note that

$$E[XY] = E[X]E[Y]$$

if X and Y are independent (this stands in contrast with the average of random variables, which is always the sum of the averages, even if the variables are not independent).

The average $E[X]$ is a sum of terms of the form

$$k\,P(X = k),$$

that is values of X and their probabilities. Similarly, $E[Y]$ is a sum of terms

$$\ell P(Y = \ell).$$

As a result, the product $E[X]E[Y]$ will have terms like

$$k\ell\,P(X = k)\,P(Y = \ell),$$

which, due to the independence of X and Y, can be rewritten as

$$k\ell\,P(X = k, Y = \ell).$$

The sum of all these term is $E[XY]$ and so $E[XY] = E[X]E[Y]$.

We can now prove that the variance of the sum of independent random variables is the sum of its variances:

$$\sigma^2(X + Y) = \sigma^2(X) + \sigma^2(Y).$$

We assume without loss of generality that $E[X] = E[Y] = 0$ (this can always be achieved by subtracting the average). Then, the variances of X, Y and $X + Y$ become simply $E[X^2]$, $E[Y^2]$ and $E[(X + Y)^2]$ and

$$\begin{aligned}
\sigma^2(X+Y) &= E[(X+Y)^2] \\
&= E[X^2 + 2XY + Y^2] && \text{(expanding)} \\
&= E[X^2] + 2E[XY] + E[Y^2] && \text{(prop. mean)} \\
&= E[X^2] + 2E[X]E[Y] + E[Y^2] && \text{(independ.)} \\
&= E[X^2] + E[Y^2] && \text{(zero average)} \\
&= \sigma^2(X) + \sigma^2(Y) && \text{(zero average)}
\end{aligned}$$

The result is of course valid for sums of more than two independent variables.

2 Counting

Motivation: avoid counters with many bits when the data are huge. A counter with n bits can count at most 2^n events. This is the limit to beat.

2.1 First solution

To double the number of events that one can count with a given counter, one may increment the counter with probability $1/2$, each time an event occurs. The idea is to *increment the counter one half the number of times*.

Imagine that there is a function `rand()` that returns a random real X, belonging to the interval $[0, 1]$. The function has "no preference" concerning the range of its values, that is, it follows a uniform distribution. This implies that the probability of $X > 0.5$ and $X < 0.5$ are equal. Since these probabilities add up to 1, each must be equal to $1/2$.

One may take random decisions with probability $1/2$ using the function `rand()`. The function to increment the counter with probability $1/2$ is now easy to write:

```
if (rand() < 0.5) then
    increment_counter
endif
```

Using `octave`, the result after (say) 100 events is easily simulated:

```
# generate 100 independent random vars in [0,1]
x = rand(1, 100);
# find how many are < 0.5
n = sum(x < 0.5);
```

After executing this code, n will contain the value of the counter after 100 events.

The value of the counter is in fact a random variable itself, determined by a number of random experiments (one for which event). So it is meaningful to ask: what is the mean value of the counter after k events?

To find out, I will associate a random variable with each event. Let X_i be the variable that represents increment i, with value 1 if the counter was incremented, and value zero otherwise. Since $P[X_i=0]$ and $P[X_i=1]$ are equal to $1/2$, one has

$$E[X_i] \;=\; 0\times P[X_i=0]+1\times P[X_i=1] = P[X_i=1] = \frac{1}{2}.$$

The value of the counter after k events is the sum of the k increments, that is

$$S = X_1 + X_2 + \cdots + X_k.$$

Its mean value is

$$\begin{aligned} E[S] &= E[X_1 + X_2 + \cdots + X_k] \\ &= E[X_1] + E[X_2] + \cdots + E[X_k] \\ &= \underbrace{\frac{1}{2} + \cdots + \frac{1}{2}}_{k \text{ terms}} = \frac{k}{2}. \end{aligned}$$

Since the mean value of the counter after k events is $k/2$, the number of events can be estimated using *twice the number stored in the counter*.

The variance of any X_i is

$$\sigma^2(X_i) = E[X_i^2] - (E[X_i])^2 = E[X_i^2] - \frac{1}{4}.$$

Since

$$E[X_i^2] \;=\; 0^2 \times P[X_i=0] + 1^2 \times P[X_i=1] = \frac{1}{2},$$

we have

$$\sigma^2(X_i) = \frac{1}{2} - \frac{1}{4} = \frac{1}{4}.$$

Since the variables X_i are independent, the variance of the sum S is

$$\begin{aligned} \sigma^2(S) &= \sigma^2(X_1 + X_2 + \cdots + X_k) \\ &= \sigma^2(X_1) + \sigma^2(X_2) + \cdots + \sigma^2(X_k) \\ &= \underbrace{\frac{1}{4} + \cdots + \frac{1}{4}}_{k \text{ terms}} = \frac{k}{4}, \end{aligned}$$

which means that $\sigma = \sqrt{k}/2$. For $k = 10\,000$ events, the average is 5 000 and the standard deviation 50.

2.2 Probability distribution

After the occurrence of k events, what is the probability of the counter having the value n? Consider for example $k = 4$, and let X_1, X_2, X_3 and X_4 be the binary variables that describe if the counter is incremented or not after each of the events. There are 16 possibilities:

X_1	X_2	X_3	X_4	value of the counter
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	2
0	1	0	0	1
0	1	0	1	2
0	1	1	0	2
0	1	1	1	3
1	0	0	0	1
1	0	0	1	2
1	0	1	0	2
1	0	1	1	3
1	1	0	0	2
1	1	0	1	3
1	1	1	0	3
1	1	1	1	4

The probabilities can now be found by counting:

$$p(0)=\frac{1}{16},\quad p(1)=\frac{4}{16},\quad p(2)=\frac{6}{16},\quad p(3)=\frac{4}{16},\quad p(4)=\frac{1}{16}.$$

When the probability of incrementing the counter is p and the probability of not incrementing it is $1-p$, the probability of observing a sum equal to n after k events is the following:

$$p(n) = \binom{k}{n} p^n (1-p)^{k-n}.$$

2.3 To probe further (in the practical lectures)

2.3.1. Write a program to simulate the counter. Perform several simulations and find the average of the obtained counts. Compare the average with $E[S]$.

2.3.2. Show that the average of the sum of two random variables is the sum of the average of each of the random variables, $E[X+Y] = E[X] + E[Y]$.

2.3.3. Discuss the expression *increment the counter one half the number of times*.

2.3.4. What is the probability distribution of S? That is, what is the probability of S being zero, one, etc., which can be denoted by $p(k)$?

2.3.5. What is the variance of S?

2.3.6. Find the variance of the estimates obtained by the program and compare it with the value obtained in 2.3.5.

2.3.7. Does anything change if the counter *overflows*?

2.4 A simple change

Is it possible to enlarge the range of the counter even more? Suppose that the range is to be multiplied by 64 instead of 2. The natural solution is to increment

the counter with probability 1/64 instead of 1/2. The variables X_i have to be defined as follows:

$$X_i = \begin{cases} 1, & \text{with probability } 1/64, \\ 0, & \text{with probability } 63/64. \end{cases}$$

The average value of each increment X_i is now

$$\begin{aligned} E[X_i] &= 0 \times P[X_i = 0] + 1 \times P[X_i = 1] \\ &= 1 \times P[X_i = 1] = \frac{1}{64}. \end{aligned}$$

The value of the counter after k events is still given by the sum of the k increments,

$$S = X_1 + X_2 + \cdots + X_k,$$

and its average value is

$$E[S] = E[X_1] + \cdots + E[X_k] = \frac{1}{64} + \cdots + \frac{1}{64} = \frac{k}{64}.$$

In this case, the number of events can be estimated using $64n$, where n is the value stored in the counter.

2.5 To probe further (in the practical lectures)

2.5.1. What are the advantages and disadvantages of using 64 instead of 2?

2.5.2. What is the variance of S?

2.5.3. Write a program to simulate the counting process. Average the counting results over several simulations. Compare that average with $E[S]$.

2.5.4. Find the variance of the counts obtained by simulation and compare with the result obtained in 2.5.2.

2.5.5. What is the probability distribution of S?

2.6 Second solution

In this case, the probability of incrementing the counter decreases as its value increases. When the value of the counter is n, the probability of incrementing it is 2^{-n} (see Fig. 1). Note that the *average increment* is now 2^{-n}. Thus, when the counter contains n, its average will increase one unit after 2^n events, as in the following sequence:

Events	*Value of the counter*	*Number of events*
x	1	1
x		
x	2	3
x		
x		
x		
x	3	7
x		
x		
x		
x		
x		
x		
x		
x	4	15

Note that $2^n - 1$ seems to be the correct estimate for the number of events. In fact, as shown next, the average value of 2^n, where n is the value stored in the counter after k events, is $k + 1$, which confirms that.

There are only two situations that may lead to a count of n after k events:

- The count was $n - 1$ after $k - 1$ events and the counter is not incremented.
- The count was n after $k - 1$ events and the counter is incremented.

These two situations lead to the following equation, in which P_k^n denotes the probability of the counter having value n after k events:

$$\boxed{P_k^n = P_{k-1}^{n-1}\, 2^{-(n-1)} + P_{k-1}^n\, (1 - 2^{-n}).}$$

The equation mentions the preceding event $k - 1$, and so it makes sense to assume $k \geq 1$. The counter will contain 1 after the first event, that is, $P_1^1 = 1$ and $P_1^0 = 0$ (in fact, $P_k^0 = 0$, $k \geq 1$). Also,

$$\sum_{n=1}^{k} P_k^n = 1$$

for any $k \geq 1$, because after k events the value of the counter has to be one of the integers $0, 1, \ldots, k$.

Since the value n of the counter after k events is a random variable, 2^n is another random variable. Its average can be found as follows:

$$\begin{aligned} \sum_{n=1}^{k} 2^n P_k^n &= \sum_{n=1}^{k} 2^n\, P_{k-1}^{n-1}\, 2^{-(n-1)} + \sum_{n=1}^{k} 2^n\, P_{k-1}^n\, (1 - 2^{-n}) \\ &= 2 \sum_{n=1}^{k} P_{k-1}^{n-1} + \sum_{n=1}^{k} P_{k-1}^n\, (2^n - 1) \\ &= 2 + \left(\sum_{n=1}^{k-1} 2^n\, P_{k-1}^n \right) - 1 \\ &= 1 + \sum_{n=1}^{k-1} 2^n\, P_{k-1}^n. \end{aligned}$$

This shows that the average value of 2^n after k events is equal to its average after $k - 1$ events, plus one unit; or equal to its average after $k - 2$ events, plus two units; and so forth until reaching the average after one event, plus $k - 1$ units. The value of the counter after the first event is $n = 1$, and so the average of 2^n after the first event is 2. Thus,

$$\sum_{n=1}^{k} 2^n P_k^n = k + 1.$$

After k events, the average of 2^n, in which n is the value in the counter, is $k + 1$. This confirms the previous suggestion: $2^n - 1$ is an estimate for k, the number of events.

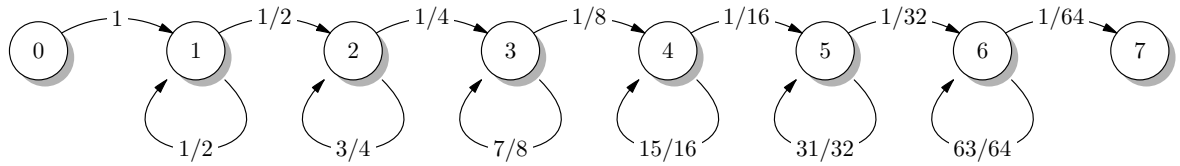


Figure 1: The probability of incrementing the counter decreases as its value increases. In this case, when the value of the counter is n, the probability of incrementing it is 2^{-n}.

The average value of the square of 2^n after k events is

$$\begin{aligned}\sum_{n=1}^{k} 2^{2n} P_k^n &= \sum_{n=1}^{k} 2^{n+1} P_{k-1}^{n-1} + \sum_{n=1}^{k} P_{k-1}^n (2^{2n} - 2^n) \\ &= \sum_{n=1}^{k} 2^{n+1} P_{k-1}^{n-1} + \sum_{n=1}^{k} 2^{2n} P_{k-1}^n \\ &\quad - \sum_{n=1}^{k} 2^n P_{k-1}^n. \qquad (1)\end{aligned}$$

In the first term, perform the substitution $u = n - 1$:

$$\begin{aligned}\sum_{n=1}^{k} 2^{n+1} P_{k-1}^{n-1} &= \sum_{u=0}^{k-1} 2^{u+2} P_{k-1}^u \\ &= 4 \sum_{u=1}^{k-1} 2^u P_{k-1}^u = 4k,\end{aligned}$$

using the result for the average value. The third term can be simplified in a similar manner:

$$\sum_{n=1}^{k} 2^n P_{k-1}^n = \sum_{n=1}^{k-1} 2^n P_{k-1}^n = k,$$

again using the result for the average value. Returning to (1),

$$\sum_{n=1}^{k} 2^{2n} P_k^n = 3k + \sum_{n=1}^{k-1} 2^{2n} P_{k-1}^n.$$

This means that the average of the square of 2^n after k events is equal to the average of the same square after $k-1$ events, plus $3k$ units. Note the structure of the relation:

$$\begin{aligned}F(k) &= 3k + F(k-1) \\ F(k-1) &= 3(k-1) + F(k-2) \\ F(k-2) &= 3(k-2) + F(k-3) \\ &\vdots \quad \vdots \\ F(2) &= 3 \cdot 2 + F(1).\end{aligned}$$

This leads to the following conclusion:

$$\sum_{n=1}^{k} 2^{2n} P_k^n = \frac{3k(k+1)}{2} + 1.$$

To find the variance we only need to subtract the square of the average:

$$\sigma^2 = \frac{3k(k+1)}{2} + 1 - (k+1)^2 = \frac{k(k-1)}{2}.$$

2.7 To probe further (in the practical lectures)

2.7.1. Write a program to simulate the counter. Perform several simulations and find the average of $2^n - 1$, where n is the counter value. Compare with the theoretical value.

2.7.2. What are the limits for counters with 4 bits? And 8 bits?

3 Set membership and cardinality

Consider the following question: given a bit string s (of arbitrary size) and a set S, does s belong to S? The answer is easy when the set S is "small". For the huge sets that occur in *big data* problems, the answer is not as easy.

3.1 Hashing

It is possible to regard a bit string s as an integer. In turn, the integer can be viewed as an index into a vector v. This leads to a simple search method. First, set all the elements of v to zero. Then, for each s belonging to S, set $v[s] = 1$. To check if the bit string t is in S, look at $v[t]$. The bit string t belongs to the set S if and only if $v[t] = 1$.

There are no difficulties when the elements of S can be represented with few bits (for example, when they have 20 bits it is enough to use a vector about about a million bits).

However, when they have arbitrary sizes, it is convenient to consider an intermediate step: a function, called *hashing function*, that assigns an integer of fixed size to any input bit string. As an example[3] (DJB31MA):

```
uint hash(const uchar* s, int len, uint seed)
{
    uint h = seed;
    for (int i=0; i < len; ++i)
        h = 31 * h + s[i];
    return h;
}
```

Each element of S can be hashed, and therefore associated with an integer. The integer can then be mapped to a vector of appropriate size, n. The method is flexible, but it introduces a new difficulty.

[3] In fact, the example leads a family of examples, since the argument `seed` is allowed to vary.

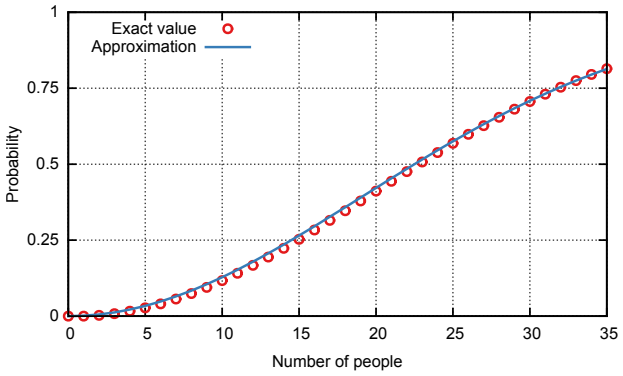


Figure 2: Probability of finding in a group of m people at least two with the same birthday, for several values of m. The probability exceeds $1/2$ when $m \geq 23$.

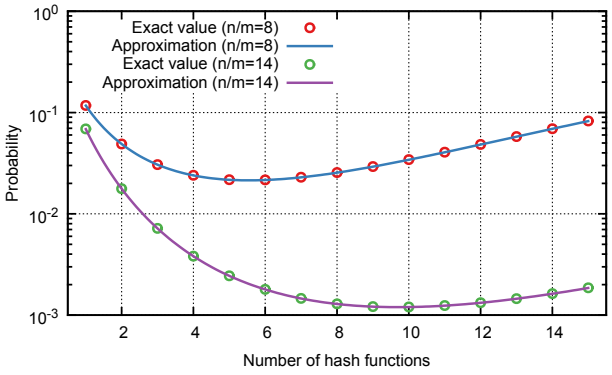


Figure 3: Error probability (false positive) as a function of the number of hash functions.

When two distinct elements are mapped to the same position in the vector, there is a *collision*. The analysis of collisions is simplified by assuming that the mapping is randomly performed and uniformly distributed (that is, each of the n results is equally likely).

The probability of having zero collisions in a vector of size n after inserting m elements is

$$\begin{aligned} p &= \frac{n-1}{n} \times \frac{n-2}{n} \times \cdots \times \frac{n-(m-1)}{n} \\ &= \left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)\cdots\left(1-\frac{m-1}{n}\right) \\ &\approx e^{-1/n} e^{-2/n} \cdots e^{-(m-1)/n} \\ &= e^{-m(m-1)/2n} \approx e^{-m^2/2n}. \end{aligned}$$

The probability of having at least one collision is

$$q \approx 1 - e^{-m^2/2n}.$$

We have $q = 1/2$ when

$$\frac{m^2}{2n} = \log 2 \approx 0.693,$$

that is,

$$m \approx \sqrt{1.386\, n} \approx 1.177\sqrt{n}.$$

An example (Fig. 2): for $n = 365$, the probability of a collision $\geq 1/2$ for $m > 22.49$ (this is the birthday paradox).

3.2 To probe further (optional work)

3.2.1. Design and write a program to forge digital signatures based on the given hashing function, for 32 bits.

3.3 Bloom filter

A Bloom filter uses k hash functions over the same bit vector. I will assume that each hash function produces n uniformly distributed results, and that they are independent.

Let b_i be the value of bit i. Initially, all bits are set to zero. To insert the first element in the filter, we use the k hash functions, which determine which bits need to be set to 1. The probability of $b_i = 1$ after using the first hash function is $1/n$, and so the probability of $b_i = 0$, under the same conditions, is $1 - 1/n$. To insert the element in the filter this operation has to be repeated k times, since there are k hash functions. The entire process is then repeated to insert the remaining elements of S.

Since there are k hash functions and m elements to insert in the filter, the probability of having $b_i = 0$ after handling all m elements is (recall that we are assuming independence)

$$\left(1-\frac{1}{n}\right)^{km} = \Bigg[\underbrace{\left(1-\frac{1}{n}\right)^{m}}_{a}\Bigg]^{k} = a^k.$$

The probability of $b_i = 1$, under the same conditions, is

$$1 - \left(1-\frac{1}{n}\right)^{km} = 1 - a^k.$$

There is an error (a false positive) when we look up a bit string that is not in S, but find the corresponding k bits all set to 1. The probability of this event, after inserting m elements, is therefore

$$p = \left[1 - \left(1-\frac{1}{n}\right)^{km}\right]^k = \left(1-a^k\right)^k. \qquad (2)$$

To determine the value of k that minimises the probability of error p (see Fig. 3) we minimise $\log p$, which has a more tractable structure:

$$\log p = k \log(1 - a^k).$$

Finding the derivative and setting it to zero leads to

$$(1-a^k)\log(1-a^k) - a^k \log a^k = 0,$$

the solution of which is $a^k = 1/2$. The best value of k would therefore be

$$k_{\text{opt}} = \frac{\log 1/2}{\log a} = \frac{\log 1/2}{m \log(1-1/n)} \approx \frac{n \log 2}{m} \approx \frac{0.693\, n}{m}.$$

However, this value is not an integer, and so in practice we need to use the closest integer. If we could have $a^k = 1/2$ for an integer k, the equation (2) would lead to

$$p_{\text{opt}} = 2^{-k_{\text{opt}}},$$

which can therefore be taken as a lower bound to the error probability.

3.4 Histogram or frequency estimation

The replacement of the binary flags in a Bloom filter with counters enables the estimation of the number of times that any given set element has been seen. The algorithm is an obvious extension of the basic Bloom filter algorithm: the k hash functions are used to find k counters, which are then incremented by one unit. To estimate the frequency of a given symbol, hash the symbol using the same k hash functions and find the minimum value over all k counter values. Given the possibility of collisions, and hence of counting errors, the least count is the one less affected by errors.

3.5 To probe further (in the practical lectures)

3.5.1. Implement a Bloom filter or a counting Bloom filter (the choice is yours).

3.5.2. There is a more precise description of the work and a Java example in e-learning.ua.pt. Use a programming language of your choice.

4 Minhash and applications

4.1 Finding duplicates

A common task is to identify duplicate or very similar elements in a set of objects (such as books, web pages or digital pictures). The obvious solution is to compare all pairs of objects:

```
for (i=0; i < n; ++i)
  for (j=0; j < i; ++j)
     if (obj(i) == obj(j))
       Print obj(i) and obj(j) are duplicates
```

Assume that there are one million objects to compare that is, $n = 10^6$. It is easy to see that we would have to perform on the order of 10^{12} comparisons (as many as the number of distinct pairs of objects). Assume further that we are able to compare 1 million objects per second. (keep in mind that the objects can be books or images). At that rate, the time required to complete the task would be close to 10^6 seconds, that is, more than 10 days. However, if the number of objects doubles, the time necessary increases by a factor of four. This solution is not feasible and a different approach must be found.

4.2 Minhash

To find a better approach, it is convenient to characterise the content of the object (book, web page, digital picture, digital music, etc.) in a simple way.

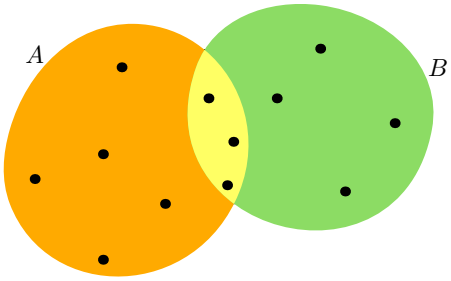


Figure 4: $|A \cap B| = 3$, $|A \cup B| = 12$, therefore the Jaccard index of A and B is $J(A, B) = 3/12$.

Hashing provides a good alternative: after identifying the key elements $s_1, s_2, \ldots, s_n$ of the object A, replace them by $h(s_1), h(s_2), \ldots, h(s_n)$, where h is a hashing function. I will use the abbreviation $h(A) := \{h(s_1), h(s_2), \ldots, h(s_n)\}$.

The next step is to find a way of measuring the similarity of any two given objects. I will use the *Jaccard index*:

$$J(A, B) := \frac{|A \cap B|}{|A \cup B|}.$$

Here, $|S|$ denotes the number of elements of the set S. The Jaccard index is a number in the range $[0, 1]$. When the two sets are disjoint, $J(A, B) = 0$ and when the sets have exactly the same elements $J(A, B) = 1$ (Fig. 4).

Last, but not the least, it is not necessary to compare all the numbers $h(A)$ and $h(B)$ to estimate the similarity of A and B. Is is enough to compare the two smallest elements of each set, which are called "minhash". For example, the minhash of A is

$$\min h(A) = \min\{h(s_1), h(s_2), \ldots, h(s_n)\}.$$

The key fact is the following: the probability of the minhash of A be equal to the minhash of B is the Jaccard index of A and B:

$$P[\min\, h(A) = \min\, h(B)\,] \quad = \quad J(A, B).$$

The proof can be based on the following argument: any element of the union of the two sets A and B has the same probability of having the least hashed value, and the required probability increases in direct proportion to the size of $A \cap B$.

4.3 Signature matrix

Consider now the *signature matrix* of the objects A_i, for the k hash functions $h_1, h_2, \ldots, h_k$:

	A_1	A_2	$\cdots$	A_n
h_1	$\min h_1(A_1)$	$\min h_1(A_2)$	$\cdots$	$\min h_1(A_n)$
h_2	$\min h_2(A_1)$	$\min h_2(A_2)$	$\cdots$	$\min h_2(A_n)$
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
h_k	$\min h_k(A_1)$	$\min h_k(A_2)$	$\cdots$	$\min h_k(A_n)$

Suppose that A_1 and A_2 are very similar, with Jaccard index $J(A_1, A_2) = 0.9$. What is the probability of every

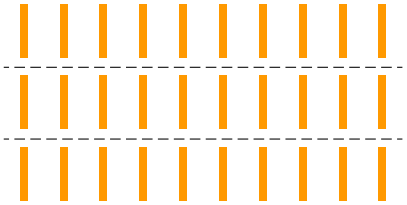
minhash of A_1 being equal to the corresponding minhash of A_2? It is 0.9^k, which tends rapidly to zero as k increases. The comparison of entire columns of the signature matrix does not lead to interesting results.

4.4 LSH

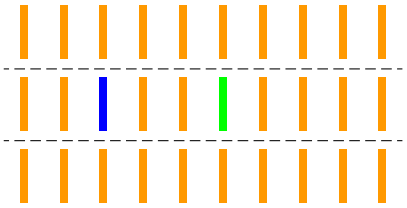
There are advantages in dividing the signature matrix in bands with a certain number of rows. In the next example there are three objects, A_1, A_2 and A_3, and three bands, each having two rows:

	A_1	A_2	A_3
h_1	$\min h_1(A_1)$	$\min h_1(A_2)$	$\min h_1(A_3)$
h_2	$\min h_2(A_1)$	$\min h_2(A_2)$	$\min h_2(A_3)$
h_3	$\min h_3(A_1)$	$\min h_3(A_2)$	$\min h_3(A_3)$
h_4	$\min h_4(A_1)$	$\min h_4(A_2)$	$\min h_4(A_3)$
h_5	$\min h_5(A_1)$	$\min h_5(A_2)$	$\min h_5(A_3)$
h_6	$\min h_6(A_1)$	$\min h_6(A_2)$	$\min h_6(A_3)$

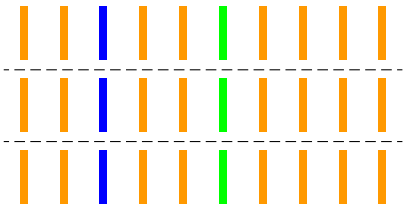
It is convenient to represent the bands in an abbreviated way, which nevertheless shows the relevant blocks of the signature matrix. The next example refers to 10 objects and 3 bands:



Consider the two blocks marked in the following figure:

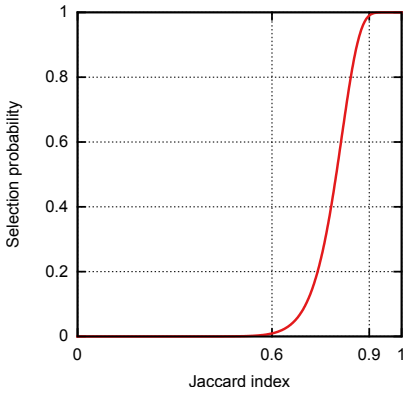


The probability of all elements of the blue block being equal to the corresponding elements of the green block is J^r, where J is the Jaccard index of the two objects and r is the number of rows in each band. The probability that this does not happen is of course $1 - J^r$. The probability that it does not happen in any band, that is, in the blue and green blocks shown in the next figure,



is $(1 - J^r)^b$, where b is the number of bands. Thus, the probability of finding at least one band in which every minhash of the blue and green blocks match is

$$P = 1 - (1 - J^r)^b.$$



This equation is the basis of an algorithm for detecting similar objects. The algorithm is known as LSH, or *locally sensitive hashing*. The idea is to consider as "similar" all objects for which there is a minhash match over at least one band. There will be false positives and false negatives, but their probability can be controlled by proper choice of the parameters b and r.

Imagine that we want to select with probability < 0.01 all objects with Jaccard index 60% or less, but that we want to select with probability > 0.99 all objects with Jaccard index at least 90%. It is possible to find r and b so that these two conditions are verified. The solution is $b \approx 20$ and $r \approx 15$. The curve $P(J) = 1 - (1 - J^r)^b$ for these values of b and r is shown in Fig. 4.4.

4.5 To probe further (optional work)

4.5.1. Download the file `ratings.dat` from the "MovieLens 10M dataset" (google: movielens dataset). The file has 10 000 054 lines. Each line has a number that identifies one user (there are 69 878 users) and another number that identifies a movie rated by that user (there are 10 677 movies).

4.5.2. There are users that have rated exactly the same set of movies. Find possible candidates in less than a second. Check if they are indeed duplicates.