Probability

From HPCM Wiki

Contents

What Are Probabilities

Probabilities are ratios of unbounded counts.

Suppose we have an experiment that we can run over and over again, and which has N possible distinct outcomes, o1, o2, ..., oN. Think of an N-sided die that can be thrown over and over again.

Suppose that after the first M repetitions of the experiment (e.g., die throws) the number of times each outcome has been observed are c1, c2, ..., cN, respectively. Then we look at the ratios

```
     c1/M, c2/M, ..., cN/M
```

and see how these ratios change as we keep repeating the experiment and increasing M. If these ratios tend to converge to particular numbers that we call `limits', we say that these limits are the `probabilities' of the outcomes, and write p1, p2, ..., pN to denote these limits, i.e., these probabilities:

```
 c1/M → p1, c2/M → p2, ..., cN/M → pn
```

So for example, if the experiment consists of throwing a normal 6-sided dice, we might expect all the outcome probabilities to converge to 1/6, so we get

```
     p1 = p2 = p3 = p4 = p5 = p6 = 1/6
```

But things might not be thus, so let us look at what else might happen.

One possibility is that the ratios converge but not to equal values, so maybe

```
  p1 = 5/60, p2 = 15/60, p3 = p4 = p5 = p6 = 10/60 = 1/6
```

in which case throwing a 1 is less likely than one would expect and throwing a 2 more likely. In this case the probabilities are still well defined, but they differ from what we expected. For dice we say that a die is `unbiased' if all the probabilities are equal, and `biased' if they are not. Gamblers may try to cheat by using biased dice which they falsely claim are unbiased, for example.

Another possibility is that the ratios do **not** converge to particular numbers, but tend to wander about over a range of values as M, the number of experiments (e.g., die throws), increases. Then the probabilities are **undefined**. For example, you are gambling with a cheater who has magnetized the die and put an electromagnet under the table which he controls with a foot switch. You can say if you like that there is an external factor, the switch position, that affects the probabilities, but if you don't know what the external factors are, and they keep changing while you repeat the experiment, then all you see is ratios that refuse to converge.

Another case where the probabilities are undefined would be if the die were to erode as you repeatedly threw it, and become unbalanced, so some sides became more probable and others less probable. Then the numbers tend to drift slowly, rather than wander back and forth. Still, if the numbers do not converge, the probabilities are undefined.

Yet another case would be where the experiment has an external factor that varies periodically with M. Suppose the experiment has the day of the week as an external factor, so if we run the experiment only on Mondays we would get different probabilities than if we ran the experiment only on Tuesdays. Suppose instead we do 100 repetitions of the experiment each weekday, and average them all together. Then we get a well defined probability that is the average of the probabilities of the different weekdays, but we do not realize that the day of the week is an external factor. So external factors can hide.

All this brings out the important point that each repetition of the experiment is **supposed to be independent** of - i.e., unaffected by - both external factors and affects from previous repetitions of the experiment.

The other important point is that probabilities are just ratios of counts, so if you can compute counts you can compute probabilities.

Activities and Events

We are going to make an important change in terminology.

First, instead of an `experiment' we are going to talk about an `activity'. Second, instead of an `outcome' we are going to talk about an `event' that happens during the activity.

For example, suppose the activity consists of throwing a 6-sided die **twice**. We shall give events in this activity names, such as 3* for the event of throwing a 3 on the first throw, *5 for throwing a 5 on the second throw, 35 for throwing a 3 on the first throw **and** a 5 on the second throw, and == for throwing the **same** number on both throws.

Then our event names are:

```
  i*   Throwing an i with the first throw, for some 1 <= i <= 6
  *j   Throwing a j with the second throw, for some 1 <= j <= 6
  ij   Throwing an i with the first throw and a j with
       the second throw, for some 1 <= i,j <= 6.
  ==   Throwing the same number with both throws.
```

Note that

```
 i* is the union of i1, i2, i3, i4, i5, and i6.
 *j is the union of 1j, 2j, 3j, 4j, 5j, and 6j.
 == is the union of 11, 22, 33, 44, 55, and 66.
```

Now to define probabilities one must repeat the activity a very large number of times, and the repetitions must be independent of external factors and the affects of previous repetitions. We generally do **not** talk about this, and merely imply it. We assume the activity is repeatable in principal, though usually it is not repeated in practice.

Furthermore, we assume that if we did repeat the activity, then the count ratios will converge to limits so probabilities are well defined. Thus if cE is the number of times event E happens when repeating the activity M times, we assume that cE/M converges to a limit, cE/M → pE, and we call this limit pE the probability of event E in the activity.

Again the main point is that if you can compute the counts of activity events you can compute the probabilities, if they exist. However, there is some danger of forgetting that for many activities and events the probabilities may not be well defined. Of course, in the rest of this discussion we will assume that the probabilities are always defined.

So to consider some events in our example. Clearly the first throw must produce one of the 6 possible results, so

```
   c1* + c2* + c3* + c4* + c5* + c6* = M
```

and therefore

```
   p1* + p2* + p3* + p4* + p5* + p6* = 1
```

Also, assuming that the die is unbiased is the same as assuming that we will get the same probabilities if we switch numbers on the sides of the die, so assuming the die is unbiased we get

```
p1* = p2* = p3* = p4* = p5* = p6*
```

and given that these sum to 1, they all must = 1/6. Similarly p*j = 1/6.

What about p11, p12, etc.? We will make the assumption that the two die throws in our activity are `independent' of each other. What this means is that the probability of throwing an i on the first throw does **not depend** upon the result of the second throw, and the probability of throwing a j on the second throw does **not depend** upon the result of the first throw. Or

```
pi1 = pi2 = pi3 = pi4 = pi5 = pi6  for 1 <= i <= 6
p1j = p2j = p3j = p4j = p5j = p6j  for 1 <= j <= 6
```

and since we already know that pi* = p*j = 1/6, we deduce that pij = 1/36 for 1 <= i,j <= 6.

Independent and Dependent Events

In the above example activity we assumed that the second die throw was independent of the first, or equivalently, that

```
pij = pkj for all 1 <= i,j,k <= 6
```

Now consider another example activity, in which **first** a **6-sided** unbiased die is thrown to get the value i, and then **second** an unbiased **i-sided** die is thrown to get the value j. So for example, if the first throw is a 2, then a 2-sided die (i.e., a coin), is thrown second, and if the first throw is a 5, then a 5-sided die is thrown second. The same kind of reasoning as above gives

```
p1* + p2* + p3* + p4* + p5* + p6* = 1
p1* = p2* = p3* = p4* = p5* = p6* = 1/6
pi1 + ... + pii = 1/6  for 1 <= i <= 6
pi1 = ... = pii = i/6i  for 1 <= i <= 6, so
pij = 1/6i  for 1 <= i <= 6, 1 <= j <= i
```

Therefore the probabilities for the outcomes of the second throw are `dependent' upon the outcome of the first throw.

Now let us talk a bit more abstractly. Suppose we have an activity with two events, E and F. Let pE be the probability that E happens, pF the probability that F happens, and pEF be the probability that **both E and F** happen. Then we introduce the purely mathematical definition:

> E and F are `independent' if and only if pEF = pE * pF.
> E and F are `dependent' otherwise.

Why do we define things thus? Well, if E and F are independent in the intuitive sense, we expect cEF/cE and cF/M to approach the same limits, or in words, the proportion of F's that happen when E's happen is the same as the proportion of F's that happen regardless of whether E happens. But cEF/cE = (cEF/M)(cE/M), so passing from counts to limits, we expect (cEF/M)/(cE/M) and cF/M to approach the same limits, so pEF/pE = pF provided pE ≠ 0. But this implies pEF = pE*pF, which is our mathematical definition of independence. Additionally note that if pE = 0 then necessarily pEF = 0 and E and F are independent by our mathematical definition.

We therefore introduce the following new notation. p(F|E) is the probability that F happens given that E has happened. Because we expect this to be the limit of cEF/cE = (cEF/M)/(cE/M) → pEF/pE, even if E and F are dependent, we have p(F|E) = pEF/pE, even if E and F are dependent, **provided** pE ≠ 0. If pE = 0 then p(F|E) is undefined, but if pEF and pE are defined and pE ≠ 0, then p(F|E) is defined.

p(F|E) is called a `conditional probability'; it is the probability of F given condition E. So in our second example activity above, where if the first die throw was i so the second die thrown was i-sided,

```
 p(*j|i*) = pij/pi* = (1/6i)/(1/6) = 1/i  for 1 <= i <= 6, 1 <= j <= i
```

Its interesting to work out some of the more complex probabilities in the second example activity where

```
pi* = 1/6
pij = 1/6i
p(*j|i*) = pij/pi* = 1/i
p(i*|*j) = pij/p*j
```

But what is p*j? Well,

```
p*1 = p11 + p21 + p31 + p41 + p51 + p61
    = (1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6) / 6
p*2 =       p22 + p32 + p42 + p52 + p62
    = (    1/2 + 1/3 + 1/4 + 1/5 + 1/6) / 6
p*3 =             p33 + p43 + p53 + p63
    = (          1/3 + 1/4 + 1/5 + 1/6) / 6
p*4 =                   p44 + p54 + p64
    = (                1/4 + 1/5 + 1/6) / 6
p*5 =                         p55 + p65
    = (                      1/5 + 1/6) / 6
p*6 =                               p66
    = (                            1/6) / 6
```

and as p(i*|*j) = pij/p*j:

```
p(i*|*1) = (1/6i)/p*1 = 1 / (1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6)i
p(i*|*2) = (1/6i)/p*2 = 1 / (    1/2 + 1/3 + 1/4 + 1/5 + 1/6)i
p(i*|*3) = (1/6i)/p*3 = 1 / (          1/3 + 1/4 + 1/5 + 1/6)i
p(i*|*4) = (1/6i)/p*4 = 1 / (                1/4 + 1/5 + 1/6)i
p(i*|*5) = (1/6i)/p*5 = 1 / (                      1/5 + 1/6)i
p(i*|*6) = (1/6i)/p*6 = 1 / (                            1/6)i
```

The message here is first, that dependent probabilities may not be simple, and second, that dependent probabilities are not that hard to compute if you have a computer.

Now that we understand what independent means mathematically, let us look back at our original situation where we had an experiment with outcomes o1, o2, o3, ..., oN and we needed to run M **independent** repetitions of the experiment. Independence means specifically that if we group the M independent runs into M/2 pairs and look at the pairs of outcomes oij we expect that pij = pi * pj. And if we group into M/3 triples we expect pijk = pi*pj*pk. And so forth. This gives us another subtle way that experiment repetitions may not be independent.

So, for example, if we hire a lazy fellow to run 10,000 repetitions of the experiment we might find that the results look OK until we group them into pairs and discover that if the first experiment of a pair had outcome oi then the second always had outcome oi, so the lazy fellow was only running 5,000 repetitions and recording each outcome twice! So his 10,000 outcomes are **not** the result of independent repetitions of the experiment.

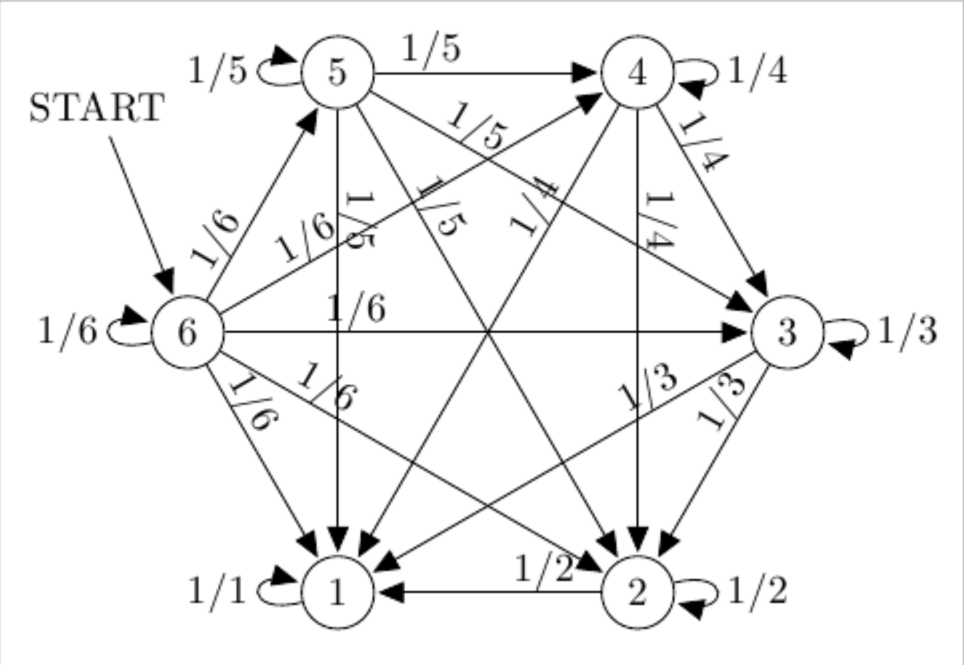
Markov Processes

Suppose we have an activity in which actions occur one after the other. At any point there are several possible `next' action outcomes, o1, o2, ..., oN, and for each a probability, p1, p2, ..., pN, such that

```
p1 + p2 + ... + pN = 1
```

If in addition these probabilities only depend upon the outcome of the last action, then the activity is called a `Markov Process'.

We can represent a Markov Process by a directed graph whose nodes are action outcomes and whose arrows represent transitions between action outcomes induced by actions. Each arrow can be labeled with its probability. The activity we described above in which a 6-sided die was thrown to get a value i and then an i-sided die was thrown is a Markov Process in which throws of n-sided dice are the actions and the result of throws are the action outcomes.



Dice Markov Process

However, it is convenient to generalize the activity to one where there are many die throws, and each throw uses an i-sided die, where i is the result of the previous throw, or is 6 for the first throw. This activity is a Markov Process with the graph given by the Dice Markov Process figure.

In this graph each node represents the result of the last die thrown, which is also the number of sides of the next die to be thrown. The 6 node is a bit of a special case as it is also the START node, and at start there is no last die throw, but we start by throwing a 6-sided die.

A node labeled n is the source of n arrows each labeled with 1/n, which is the probability that a throw of an n-sided unbiased die will have a result equal to the label of the destination node of the arrow. Thus from node 2 there are 2 arrows, each labeled 1/2, one with destination node 1, and one with destination node 2 (the little looping arrow). And in general the arrows sourced at node n have labels 1/n and destinations labeled 1, 2, ..., n.

What you can compute from such a Markov Process is the probability of being at a given node after K throws. Let pn[K] be this probability for the node labeled n. Then

```
p6[0] = 1, p5[0] = p4[0] = p3[0] = p2[0] = p1[0] = 0
```

are the probabilities of being at nodes after K = 0 throws, that is, at the START.

We need an equation which will allow us to compute pn[K+1] from the pm[K] probabilities for all nodes m. To do this we need to carefully understand what the labels on the arrows mean.

Consider an arrow Z with arrow label pZ, source node labeled m, and destination node labeled n. What is pZ? Consider the events:

```
m[K]        after K throws we are at node m
m[K]n[K+1]  after K throws we are at node m
            and after K+1 throws we are at node n
```

Then pZ = p(n[K+1] | m[K]) = pm[K]n[K+1] / pm[K] is the conditional probability of arriving at node n after K+1 throws given that we have arrived at node m after after K throws. That is, if we repeat M times the activity of throwing the die K+1 times from start and count the events,

```
cm[K]n[K+1] / cm[K] → p(n[K+1] | m[K]) = pZ
```

Now as

```
cn[K+1] = sum over all nodes m of cm[K]n[K+1]
```

we get

```
cn[K+1] / M = sum over all nodes m of
                 ( cm[K]n[K+1] / cm[K] ) * (cm[K] / M)
```

and on passing to the limit

```
pn[K+1] = sum over all nodes m of
                 p(n[K+1] | m[K]) * pm[K]
```

or in different words,

```
pn[K+1] = sum over all arrows Z with destination node n of:
              pZ * pm[K]
              where the source of the arrow Z is labeled m
                    and the arrow Z is labeled pZ
```

In our specific dice throwing case this becomes

```
p1[M+1] = p1[M] + p2[M]/2 + p3[M]/3 + p4[M]/4 + p5[M]/5 + p6[M]/6
p2[M+1] =         p2[M]/2 + p3[M]/3 + p4[M]/4 + p5[M]/5 + p6[M]/6
p3[M+1] =                   p3[M]/3 + p4[M]/4 + p5[M]/5 + p6[M]/6
p4[M+1] =                             p4[M]/4 + p5[M]/5 + p6[M]/6
p5[M+1] =                                       p5[M]/5 + p6[M]/6
p6[M+1] =                                                 p6[M]/6
```

If we compare this to the activity we investigated in Dependent and Independent Events we find that pi* = pi[1] = 1/6 and p*j = pj[2].

The general Markov Process works the same way as our dice throwing process, except for terminology. Action outcomes are usually called `**states'** and actions are usually called `**steps'**. As computer algorithms step through states, one might expect that Markov Processes would be useful for analyzing some computer algorithms.

Different Markov Processes have very different node labels. But it is always true of the arrow labels that

```
each arrow label pZ is in the range 0 ≤ pZ ≤ 1
for each state (node) m:
    1 = sum of pZ over all arrows Z whose source is node m
            where pZ is the label of the arrow Z
```

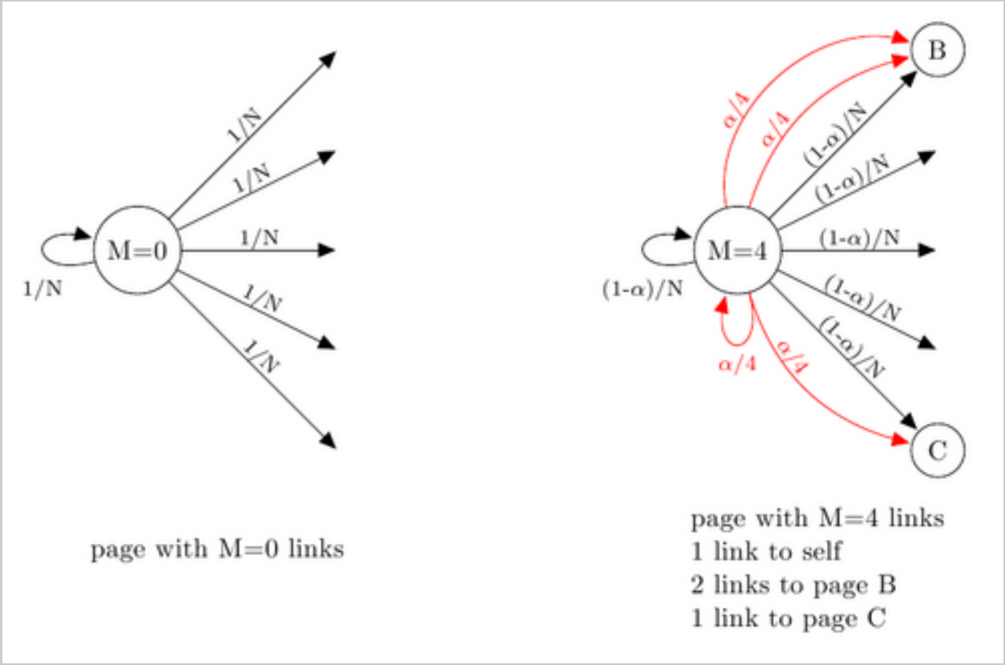
An Application: The PageRank Algorithm

An application of Markov Processes is the PageRank (http://en.wikipedia.org/wiki/PageRank) algorithm used by Google to determine which web pages will be most useful to a user of a search engine.

Consider a set of web pages which contain links to other web pages. Consider a user randomly hopping between these web pages mostly using links. We compute the probability that the user will be at a given page after K hops, and discover that as K increases these probabilities converge to limits, so we can compute the probability of the user being at a particular page in the long run (after many hops). Pages with higher probabilities are then more important, and will be listed first in Google searches.

The algorithm uses a parameter α which is the probability that a user at a page with links will not follow any of the links but will instead hop in an unbiased manner to an arbitrary page.

The specific process is as follows. The random user first chooses a page at random in an unbiased manner. If there are N pages, then at the beginning the probability that the user is at any page is 1/N.



PageRank Markov Process

Then the user moves as follows. If the user is at page p, and the page has links, then with probability α the user makes an unbiased choice of one of the links and goes to its destination. This means that if there are M links the user has probability α/M of going to the destination of each link (if there are several links with the same destination, the total probability for the destination is the sum of the link labels). Even if there are links from the current page, the user with probability 1-α chooses a completely arbitrary page from among the N total pages at random, meaning that each page gets a probability of (1-α)/N. Lastly, if the current page has zero links, the user simply goes to a random page, with the probability 1/N for each possible page.

This is a Markov Process with pages as states (nodes) and arrows for possible hops between pages. At the start each page has probability 1/N. For a page with M>0 links, there is an arrow with probability α/M for each link destination (several links may have the same destination and a link may link the page with itself). In this case there is also an arrow from the page to every page with probability (1-α)/N. But for a page with M=0 links, there is just one arrow from the page to every page with probability 1/N.

Pseudo code for computing the probabilities is:

```
// pn[K] is the probability of being at page n after K hops
//
for all pages n, pn[0] = 1/N
for K = 1, 2, 3, ...:
    for all pages n, pn[K] = 0
    for all pages m:
        let M = number of links in page m
        if M = 0:
             for all pages n:
                 pn[K] += (1/N) * pm[K-1]
        else if M > 0:
             for all pages n:
                 pn[K] += ((1-α)/N) * pm[K-1]
             for all links L in page m:
                 let page n be the destination of L
                 pn[K] += (α/M) * pm[K-1]
```

Probabilities in Computer Science

Above we emphasized various ways in which real-world probabilities could fail to exist. When is this relevant to computer science?

In the PageRank Algorithm, no claim is made that the probabilities computed correspond to any real user or group of users. The only claim is that when these probabilities are used to rank search results, presenting results with higher computed probabilities first, search engine users are happy. Therefore the probabilities are completely virtual, and not real-world.

There are two general classes of probabilities in computer algorithms. In the first class the probabilities are those of particular input data or combinations of data. In this class one should worry about how well defined the probabilities are. In the second class the probabilities all come from throwing a 2^m sided unbiased die inside the computer. If we assume the die is what the theory says it should be, then the probabilities definitely exist and can usually be computed. The only question is: can a computer algorithm be constructed which is the equivalent to throwing and almost perfect 2^m sided die. And the answer to this is yes.

See the Randomized Algorithms page for applications of probabilities in Computer Science.

Retrieved from "http://problems1.seas.harvard.edu/wiki-hpcm/index.php?title=Probability&oldid=283"

- This page was last modified on 1 February 2013, at 10:39.
- This page has been accessed 119 times.