The Probability of Discrete Strings Over Sequential Data

Casey S. Schroeder

cs@csschroeder.com

East Windsor - CT - USA

May 25, 2020

Abstract

Assuming an alphabet (e.g. {u, d}) and Markov transition matrix between letters, the basic question we answer is, *In a sequence of n transitions, what is the probability of a string (e.g. udu) occurring at least once before the final transition n.* Using this basic method, are able to solve the more complex task of evaluating the probability of statements of the form $\psi_1 \wedge \neg\psi_2$ where a given ψ is a disjunction of primitive statements of the form, *string p will occur between transition 0 and n.* We then show that this solution can be applied to boolean statements of arbitrary logical complexity involving our primitive statements as primitives. Finally, we give a method of evaluating the probability of sequences of the statements $\psi_1 \wedge \neg\psi_2$.

Keywords: strings, sequences, markov, inclusive/exclusive, pattern derivatives, discrete patterns

1 Introduction

In the most basic case, consider a random variable which can take one of two values u or d. A string *uddu* is said to occur in the sequence of length n = 10 *duududdduud* at t = 8. Our analysis starts with the question, what is the probability of getting at least one occurrence of some arbitrary string p (e.g. *uddu*) within a sequence of length n produced from successive draws of a random iid variable X (e.g. $\{u,d\}$).

Besides the fundamental nature of this analysis, what makes this question interesting from the start is that even if u and d each have the same probability of occurring, there exist strings of the same length which have different probability of occurring (one or more times) in sequences of fixed length. One can see this even with strings of length m = 2, and sequences of length n = 3. The possible sequences are *uuu, uud, udu, udd, duu, dud, ddu, ddd* and *ud* occurs in 4, *uud, udu, udd, dud* while *uu* occurs in only 3, *uuu, uud, duu*. More generally for $n <= 9$ we have the following table ref. [2].

n	u	d	uu	ud	du	dd
1	1/2	1/2	0/2	0/2	0/2	0/2
2	3/4	3/4	1/4	1/4	1/4	1/4
3	7/8	7/8	3/8	4/8	4/8	3/8
4	15/16	15/16	8/16	11/16	11/16	8/16
5	31/32	31/32	19/32	26/32	26/32	19/32
6	63/64	63/64	43/64	57/64	57/64	43/64
7	127/128	127/128	94/128	120/128	120/128	94/128
8	255/256	255/256	201/256	247/256	247/256	201/256
9	511/512	511/512	423/512	502/512	502/512	423/512

This result is in some sense intuitive to a layperson, as *uu* is more 'patterned' than *ud*, and is therefore more surprising. To one with passing familiarity of iid variables the result seems somewhat counter-intuitive at first. But an understanding can already be seen in the case of m = 2 and n = 3. If one sums up all of the occurrences of *uu* in all sequences of length n, it is the same as the sum of all the occurrences of *ud* in all of the sequences of length n. In particular, for n = 3, as above, this sum is 4 (with *uu* occurring twice in *uuu*). The reconciliation of this fact with the above table is that when *uu* does occur, it tends to occur in bunches, since when it does occur, you are already half way to another occurrence! The result is that more sequences

of length n do not have *uu* than do not have *ud*, because *uu* overlaps with itself.

Another way to state the importance of the result is in terms of wait times. The corresponding result is that the expected wait time is longer for the string *uu* than the string *ud* in an infinite sequence ref. [1]), and that is also not cured by counting *uuu* as one instance instead of two. In terms of wait times, however, the paradox is a bit stronger. It seems strange that the probability of getting *uu* and *ud* at any given time t in our sequence is the same and the wait time for *uu* is longer than for *ud*, but it is!

One does not have to search far to motivate the importance to decision making of a procedure for finding such probabilities. Imagine any industry concerned with production, shipment, or capital reserve. Suppose one is concerned with levels falling under some threshold for two successive periods. The probability of levels falling under this threshold for one period is, say, .05. We assign this event *d*, while the "other" region occurs 95% of the time. Our concern is then to find the probability, r, of striking the string *dd* (at least once) over some horizon. If we have this, then we know, e.g., the probability 1-r of staying out of trouble (e.g. solvent). In what follows we give a procedure for this and other far more complex situations.

2 Basic Method

In an initial work ref. [2] the author follows a method introduced by ref. [1] for the evaluation of such probabilities with computational efficiency. Here we introduce a more general method which uses a Markov transition matrix and applies to the iid case as a special instance.

Assume first that we have a partition over the space of events, represented with the alphabet $\{a_1, a_2, a_3, ..., a_i\}$; for convenience we may use letters from the standard alphabet $\{a, b, c...\}$ instead. Initially, we will assume that a string is any string from this alphabet of length m and a sequence any string from this alphabet of length n, $m < n$. If we have a Markov transition matrix, M, for the events in this alphabet, then we can construct a matrix M_m, with little m representing the length of the strings. This matrix will have $L^m x L^m = L^{2m}$ entries. In the case of $\{u, d\}$, we will have the matrix M as

$$M = \begin{array}{c} \\ u \\ d \end{array} \begin{array}{c} \begin{array}{cc} u & d \end{array} \\ \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix} \end{array}$$

for the fact that u and d are independent and the matrix M_2 (for $m = 2$) as

$$M_2 = \begin{array}{c} \\ uu \\ ud \\ du \\ dd \end{array} \begin{array}{c} \begin{array}{cccc} uu & ud & du & dd \end{array} \\ \begin{pmatrix} 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 \end{pmatrix} \end{array}$$

Where the rows represent transitions-from and the columns transitions-to. Note that this matrix is constructed algorithmically given M and m.

We then, concerning ourselves initially with a single string of length m which we are concerned to calculate the probability of striking:

1. Set the transitions-to column for the string of our concern to have zeros for every entry.
2. Take M_m^n for n the length of the sequence.
3. Multiply by an initial 'prefix' vector, representing the state of the data prior to the beginning of our sequence of length n; for instance, if our initial state is ud we take $(0, 1, 0, 0)M_m^n$.
4. Sum the entries in the final vector to get a single probability number that represents the probability q, of *not* getting the string in question.
5. Finally take $1 - q$ to get the probability of striking the string.

Note that

i this procedure works given any Markov transition matrix, M, whether or not the entries represent independence.

ii if one wants to know the probability of not striking a string, one can simply calculate q, skipping 5.

iii if one wants to know the probability of striking string p1 or p2 or p3... One can apply 1 to each row simultaneously and continue.

iv if one wants to know the probability of not striking p1 nor p2 nor p3..., one can follow iii and skip 5.

v if one wants to know the probability of striking string p1 or p2 or p3... and *not* r1 nor r2 nor r3... one can follow iii using p1 or p2 or p3... or r1 or r2 or r3... (notice, no 'not' nor 'nor') yielding probability s, and then follow iii for r1 or r2 or r3... (again no 'not' nor 'nor') yielding probability r, and finally taking the difference s - r to get the probability sought. This is easily shown with a venn diagram.

vi if one has strings of varying lengths, follow the methods as required for the maximum length, but remove all transitions-to which contain the strings of smaller lengths inside the strings of length m.

vii we can handle data of multiple dimensions provided the event regions form a partition of the data space; in other words, provided the letters of our alphabet are assigned to elements of a set of regions of the data space which form a partition.

viii note that if one wants to specify a larger event at a given location in a string, one can specify a string as, for instance, $< a, b, \{c, d\} >$, indicating $c \vee d$ at the third position and this can be compiled into the matrix by setting the columns for both $< a, b, c >$ and $< a, b, d >$ to all zeros.

The primary drawback of this method is the size of the matrix and the required computations given large alphabets (or data of high dimensionality), large m, or large n, if one is attempting to do calculations on the order of seconds rather than minutes or hours. Note that for i through v, and vii, the run time is dominated by the construction of M_m^n, which is $O(n * (L^{2m}))$. For the case of vi we also have the overhead of finding the smaller strings in the larger strings and applying the methods of viii will also involve overhead.

3 Logically Complex Statements

In the above section we introduce complex statements of the form $\psi_1 \wedge \neg\psi_2$ where a given ψ_i is a disjunction of primitive statements of the form *string p will occur between transition* 0 *and n*. We will here show that, in fact, we can find the probabilities for arbitrary boolean logical formula involving

these primitive statements, incorporating the above method into an additional procedure. Imagine you have an arbitrary logical formula where these primitive statements are the literals. You can translate any such formula into DNF as ϕ

$$T_i = (p_1 \wedge p_2 \wedge p_3 \wedge ... \wedge p_w ... \wedge \neg q_1 \wedge \neg q_2 \wedge \neg q_3 ... \wedge \neg q_y)$$

$$\phi = T_1 \vee T_2 \vee ... \vee T_z$$

for some finite w, y, and z. With a ϕ of this form we have the following algorithm for evaluating its probability.

1. Apply the Inclusive/Exclusive principle to change $P(\phi)$ into
$$P(\phi) = P(T_1)+P(T_2)+...-P(T_1 \wedge T_2)-P(T_1 \wedge T_3)-...+/-P(T_1 \wedge T_2 \wedge T_3 ... \wedge T_z) \quad (1)$$

2. For a given conjunctive term, collect all of the negated terms q_i into a conjunction Q and write
$$T_1 \wedge T_2 \wedge ... = (p_1^1 \wedge Q) \wedge (p_1^2 \wedge Q) \wedge ... (p_2^1 \wedge Q) \wedge (p_2^2 \wedge Q) \wedge ... \quad (2)$$

3. Applying the Inclusive/Exclusive principle, again, to the right hand side in 2, you get
$$\begin{aligned} P((p_1^1 \wedge Q) \wedge (p_1^2 \wedge Q) \wedge ... (p_2^1 \wedge Q) \wedge (p_2^2 \wedge Q) \wedge ...) = \\ P(p_1^1 \wedge Q) + P(p_1^2 \wedge Q) + ... + P(p_2^1 \wedge Q) + P(p_2^2 \wedge Q) + ... \\ -P((p_1^1 \wedge Q) \vee (p_2^1 \wedge Q)) + ... \end{aligned} \quad (3)$$

4. Note that every term will be a combination which takes the form (e.g. the last line) $P((p_i^j \wedge Q) \vee (p_k^l \wedge Q) \vee ...)$ and this will reduce to a statement of the form $P(p_i^j \vee p_k^l \vee ... \wedge Q)$, which is the form which we already know how to evaluate from the previous section.

Converting an arbitrary formula to DNF is not efficient and neither is applying I/E once, much less repeatedly. We are dealing with a method of high complexity. The reason to point out this method is not its practicality. There are two good reasons to point it out. The first is that it gives one a target to shoot for with approximate methods. The second is that it shows that one can use a collection of statements with the form $p_i^j \vee p_k^l \vee ... \wedge Q$ as probabilistic proxies for arbitrary logical formula involving these primitive statements.

4 Statements Complex in Time

We present here a method for handling sequences of complex *statements*. We allow statements of the form

$$\phi = \psi_1 \wedge \neg\psi_2$$

where a given

$$\psi_i = p_1 \vee p_2 \vee p_3 ... \vee p_r \tag{4}$$

for finite r. This is the most general form of statements found in section 2 and the form to which all others can be reduced for the sake of evaluation. We can form sequences of such statements

$$S =< \phi_1, \phi_2, ..., \phi_k > \tag{5}$$

Here S is understood as covering a data series length n, but the ϕ_i are understood to strike at some time $t < n$ for the first time, provided the probability of ϕ for a sequence of length t, minus the probability of ϕ for a sequence of length t-1, is non-zero. The equation becomes:

$$P(S, n) = \sum_{j=0}^{n} pr(\phi_1, j) P(S - \phi_1, n - j) \tag{6}$$

Where $pr(\phi, j)$ is the probability of ϕ occurring for the first time at j just mentioned.

We have until now, furthermore, assumed that the window for which strings are to strike - or not to strike - is of length n, the same for all strings. This assumption is not necessary. We can assume instead that strings have start and end times which vary across strings within a given ϕ. With this stipulation removed, instead of removing all strings in the disjunction from our matrix at the very beginning, we remove them, instead according to the timing of their windows. This will change the matrix dynamically with time, instead of fixing it at the beginning of our multiplication, but nothing else changes. This can be used simultaneously with the equation for $P(S, n)$ above.

5 A Note on The Market For Pattern Derivatives

The statements we have described make for a natural class of financial derivatives. It is further noted here that the full market for such derivatives can be supported as long as arbitrary disjunctions of strings can be priced; which they can if one has a transition matrix over the elementary states, as shown in our initial analysis.

This is because the position of holding a 'pattern' (as the author has called them in previous publications) of the form $\phi = \psi_1 \wedge \neg\psi_2$, is simulated with holding one of the form $\psi_{1,2}$, understood as containing all the strings in ψ_1 and ψ_2, and selling ψ_2. Furthermore, any arbitrarily complex logical formula can be priced based on the market price of certain contracts of the form ϕ, as described by our algorithm. The price equilibrium of every logical 'pattern derivative' will therefore be set by the prices of pattern derivatives of the form ψ.

Note that this reduction applies also to sequences of patterns, but that sequences, as we have described them, do not allow for arbitrary logical forms of patterns in the sequence. They are restricted to $\phi = \psi_1 \wedge \neg\psi_2$. It is an open question whether this restriction can be lifted, or made more general. The answer appears related to the issue of incorporating derivatives timed to payout when the pattern is struck (American variety), rather than the (European) variety supported here, which pays out at the end of the term (n).

The interested reader can find more discusson on the markets for pattern derivatives in the work ref. [2] and ref. [3]. These give more tractable methods for the form $\phi = \psi_1 \wedge \neg\psi_2$, in the case of independence over time (i.e. not the general Markov case).

This work was originally written March, 2019.

References

[1] Blom, G. and Thorburn D. (1982). How many random digits are required until given sequences are obtained? J. Applied Prob. 19, 518-531.

[2] Schroeder, C. S. and DiPierro M. (2012) Pattern Derivatives. Intl J. of Financial Markets and Derivatives.

[3] Schroeder, C. S. and DiPierro M. (2014) Pattern Derivatives Extended. Unpublished, available at: www.csschroeder.com/papers