-

Inductive Inference over Macro-Molecules.

L. Allison, C. S. Wallace and C. N. Yee.

Department of Computer Science,
Monash University,
Australia 3168.

uucp: xxx@bruce.cs.monash.oz.au xxx=[lloyd, csw, cyee]
Supported by Australian Research Council grant A48830856

1. Introduction.

Minimum message length (MML) encoding techniques are applied to inductive inference problems over strings derived from sequencing biological macro-molecules[1,2]. The paper is framed in terms of DNA strings but the techniques apply to proteins and other strings.

It is assumed that any one string is random in the sense of Kolmogorov complexity. This is very nearly the case for biological macro-molecules and it is certainly very difficult to encode a typical DNA string in much less than two bits per character. The assumption is not essential to the MML method but makes its application easier. Even if each individual string is random, it may be clear that two or more strings are related in some way. This raises a number of questions: Given some strings, with what probability are they related? If they are related, in what way are they related? What parameters of the particular model of relation best apply to the strings? What can be inferred about the model of relation in general? We show how MML encoding can answer these questions.

To illustrate the MML method in string comparison, consider a *transmitter* who must send two strings to a *receiver*. They may previously have agreed on any algorithms and conventions. The *null-theory* or hypothesis is that the strings are unrelated. If this is the case, the transmitter can do no better than transmit one string and then the other which requires approximately two bits per character. Conversely, if the strings are related and the transmitter can find a "good" theory or hypothesis relating the characters of the two strings, then it may be possible to encode the two strings more succinctly. The transmitter then sends a message which (a) states the inferred theory and (b) encodes the strings using a code based on the theory. The MML method selects the theory giving the shortest message. A very detailed theory may allow a very short encoding of the strings proper but the details may add a great deal to the message length. MML encoding defines an optimal degree of detail in this trade-off.

If accurate prior probabilities for strings and relations are known, an optimal code can be designed for them. The minimum message length is minus $\log_2$ of the probability of the event described. Often such probabilities are not known but a model having one or more parameters may be agreed on. A message must include the parameter values and the best values give the shortest message. Two hypotheses may be compared on the basis of their message lengths. The more plausible hypothesis leads to the shorter message and the difference in message lengths is minus $\log_2$ of their posterior odds-ratio. Because a natural *null-theory* can be identified, this leads to hypothesis testing: no hypothesis having a message longer than the null-theory is acceptable.

There are practical and theoretical advantages to MML encoding as a method of inference. It keeps us *honest* in that anything not agreed to *apriori* must be included in the message. When real-valued parameters are involved, it implies an optimal degree of precision for their estimates. It focuses attention on the language being encoded, on what it can express and on the probabilities of messages. When accurate prior probabilities are not known, all the techniques and heuristics of data-compression and of coding theory can be used to get a good code (theory). MML encoding is *safe* in that the use of suboptimal encoding for a proposed theory cannot make the theory appear more acceptable than it should be.

The definition of relatedness is central to the string problems. There can only be one good universal definition and it must be related to the Kolmogorov complexity of a program to produce the set of strings. This is not computable in general. Fortunately simpler and yet plausible models exist. These are for the most part based on finite-state machines or other simple

machines. In the next section we illustrate the technique on the simplest one-state models for two strings.

MML encoding has had many applications, for example, Wallace and Boulton[9] produced a practical classification algorithm using the technique. Wallace and Freeman[10] described the statistical foundations of MML encoding. In string comparison, Reichert et al[6] considered the brief encoding of an alignment of a pair of strings for a single model of relation. Felsenstein[5] gave a maximum-likelihood method to infer an evolutionary tree for a family of strings in the absence of insertions and deletions. Bishop et al[4] gave a maximum-likelihood method to infer the time since two related strings diverged which is very similar to our MML method for a one-state model. Bishop and Friday[3] noted that "there is as yet no formally developed method for assessing the significance of the difference in the likelihoods (of two different [phylogenetic] trees)." The MML method does not suffer from this problem and has a formal interpretation of differences in message length and indeed has a rigorous test of statistical significance.

2. One-State Models.

We can consider a generative or a mutation model of string relation. In the *generative model*, a simple machine reads an input string of *generation instructions* in a special alphabet and outputs the two given strings in parallel (figure 1). In the *mutation model*, a different simple machine reads two inputs in parallel. One input is one of the given strings. The other input is a string of *mutation instructions* in a special alphabet. The machine outputs the second given string. For the restricted relation models we consider, it can be shown that to every generative model there corresponds a mutation model and vice-versa. In what follows we usually adopt the generative form, but our use of the words ‘insert’, ‘delete’ and ‘change’ for certain generative instructions is common practice inherited from the mutation form. An *indel* is either an insert or a delete. A sequence of generative instructions represents a specific relation or *alignment* between two strings. Note that when considering evolution from a common ancestor, an alignment corresponds to a subset of the possible common ancestors of the strings and can be thought of as a string of fuzzy characters.

generative instructions	string A	string B
match(A)	A	A
insert(C)	–	C
match(T)	T	T
change(A,C)	A	C
delete(A)	A	–
match(G)	G	G

Figure 1. Generative Example.

It is common practice to assign weights or *costs* to each instruction. The cost of an instruction sequence is the sum of the costs of its instructions. Given strings A and B, an instruction sequence that generates A and B and achieves the minimum possible cost gives an *optimal alignment*. The minimum cost defines a function D(A,B) on strings. Writing D[i,j] for D(A[1..i],B[1..j]), D can be calculated by the well known dynamic programming algorithm (DPA, figure 2). For simplicity we assume that all matches have an equal cost, all changes have an equal cost and all indels have an equal cost. The choice of minimum made in the general step of the DPA defines a path from D[0,0] to D[|A|,|B|] which gives an optimal alignment. Here, we relate costs to probabilities of generative instructions. If something is known of these probabilities, this gives an objective way to choose costs. If probabilities are not known, it allows them to be inferred from given strings.

To encode an alignment of A and B, we transmit a string of match, change, insert and delete instructions. Note that the number of instructions must be included in the message. The probabilities of the instructions are required to design an optimal code. In this section we assume a simple one-state machine. The length of (a code word for) an instruction includes a contribution for the characters of the strings and we use these instruction lengths (figure 3) as costs in the DPA.

Given fixed probabilities and two strings, two

```
Boundary Conditions:
 D[0, 0] = 0
 D[i, 0] = indel × i,    i=1..|A|
 D[0, j] = indel × j,    j=1..|B|

General Step, i=1..|A| and j=1..|B|:
 D[i,j]=min(D[i-1,j] + indel,
            D[i,j-1] + indel,
            D[i-1,j-1] +
            if A[i]=B[j] then match
                         else change )
```

Figure 2. Dynamic Programming Alg’.

instruction	code length
match	$-\log_2(p(m))+2$
change	$-\log_2(p(c))+\log_2(12)$
insert	$-\log_2(p(i))+2$
delete	$-\log_2(p(d))+2$

$p(m)+p(c)+p(i)+p(d)=1$
symmetry: $p(i)=p(d)$

Figure 3. Costs of 1-State Machine.

alignments can be compared on the basis of their message lengths. The more plausible alignment has the shorter message length. The difference in message lengths gives $\log_2$ of the posterior odds-ratio of the alignments.

When probabilities of instructions are not known in advance, they must be included in the alignment message. One way is to add a header which states the probabilities used to an appropriate accuracy[10]. The optimality of an alignment depends on the probabilities used. When these are not known in advance, a search is necessary to find good values. Reasonable initial values are assumed and an alignment, optimal for these probabilities, is obtained. Probabilities for the next step are obtained from instruction frequencies in the alignment. The process converges in a few steps.

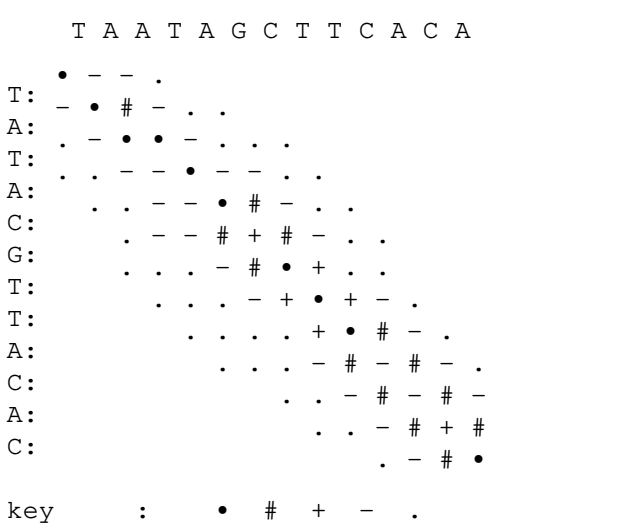
Note that an arbitrary set of instruction costs, such as those used in the edit-distance[7] or longest common subsequence problems, implies certain probabilities. These can be discovered by normalisation[1].

For given costs, optimal alignments are not unique in general. All optimal alignments and indeed all non-optimal alignments contribute to the probability of A and B being related in some way. This has been recognised by Bishop et al[4]. Even for two identical strings, eg. ACGT:ACGT, there is a small contribution from sub-optimal alignments. In computing an MML alignment, D[i,j] is minus the $\log_2$ of the probability that A[1..i] and B[1..j] arose through being related in *one* most probable way; the 'min' in the general step of the DPA corresponds to choosing the alternative having the largest probability. If probabilities are instead *added*, for they correspond to exclusive events, then D[i,j] is the $(-\log_2)$ probability that A[1..i] and B[1..j], arose through being generatively related in *some* unspecified way. We call this the *r-theory*. The addition of probabilities relies on a probabilistic argument but a corresponding code can be devised[8] and reduces the message length of the r-theory below that of an optimal alignment. As before, it is necessary to iterate to discover the probabilities to use in encoding the theory. To do this, frequencies of instructions are computed in D in a weighted fashion according to probabilities of transitions.

The *null-theory* is that strings A and B arose through being unrelated and this is the complement of the r-theory. The null-theory is encoded by encoding string A and then encoding string B. Note, the lengths of A and B must be included - see later. The odds-ratio of the strings' being related against their not being related is equal to the binary exponential of the null-theory length minus the r-theory length. It is then a simple matter to calculate the posterior probability of the strings being related.

Running the r-theory algorithm in reverse, D'[i+1,j+1] is the $(-\log_2)$ probability that A[i+1..|A|] and B[j+1..|B|] arose through being related. Thus, D[i,j]+D'[i+1,j+1]−D[|A|,|B|] is the $(-\log_2)$ probability that the "true" alignment related A[1..i] to B[1..j] and A[i+1..|A|] to B[j+1..|B|]. A density plot of this function (figure 4) is a good way of visualising possible alignments. For identical sections, there is near but not absolute certainty of the obvious alignment. Note the areas marked with '#' where there is uncertainty (GC:CG) or a branch (ACAC:CACA).

The r-theory message includes the (weighted) number of instructions. In the absence of more information, we encode this number according to Rissanen's $\log_2^*$ universal prior. The null-theory message must include the lengths of string A and of string B. If these two lengths



null-theory= 62.4 r-theory= 58.5
probability related=0.94

Figure 4. Alignment Densities.

are encoded independently then it gives a penalty to the null-theory and favours the r-theory for strings of similar length. In some circumstances this may be appropriate but we do not generally consider strings similar only on grounds of length but require similarity of structure also. The generative model with p(i)=p(d) has a built in expectation that the difference in the lengths of A and B is $O(\sqrt{(|A|+|B|)})$. We build a similar expectation into the null-theory and transmit |A|+|B| using the $\log_2^*$ prior and |A|−|B| using a binomial prior. If p(i)=p(d)=1/2, the r-theory and the null-theory have the same prior over the difference in lengths. When p(i)=p(d)<1/2 the r-theory has a tighter distribution on the difference.

The *homology* of two strings is the fraction of characters matching in an optimal alignment, say a Sellers alignment. When trying to judge the significance of homology it is common practice to scramble the letters in the two strings and recalculate the homology for a base-line figure. This procedure is not necessary with the MML method which has a built in and rigorous test of significance. Simulation experiments on string pairs having controlled homology show that MML favours the r-theory over the null theory whenever the homology exceeds that found between unrelated random strings.

Note that when considering evolution from a common ancestor, the r-theory embodies the set of all possible common ancestors, each appropriately weighted. The density plot can be thought of as a fuzzy alignment.

3. Other Models.

We note that string alignment algorithms employing linear costs, a+b×L, for indels of length L, are in common use. The constant 'a' is a start-up cost to discourage the inference of many short block-indels. We can model this with a three-state machine. There is a small probability of entering an insert (or delete) state but a higher probability of continuing in it once there. The DPA has been modified to maintain three values in each cell of the matrix D. One value corresponds to each state. As before probabilities are *added* to calculate an r-theory. The algorithm is run forward and backwards and results combined to get a density plot of alignments.

More elaborate finite-state models are possible. For example, a five-state model can account for both short and long inserts and deletions. This may be appropriate to deal with combinations of introns and exons.

There are a great many models of relation between strings. An important class of model is based on simple finite-state machines. The inference algorithms are variations on the dynamic programming algorithm. Each model is defined by probabilities of machine instructions and by probability distributions on the run-lengths of blocks of matches, changes and indels. The linear-cost algorithms discussed above imply geometric distributions for run-lengths. A linear cost amounts to using a unary code which is optimal for a geometric distribution. An arbitrary cost function w(L) implies a prior probability distribution $2^{-w(L)}$ on run-lengths, after normalisation and modulo the characters. Any problem-specific knowledge of actual probabilities can be built into a cost function. Conversely, if a particular cost function performs well it gives information about mechanisms applying to strings.

Other types of relation model are possible. Block mutations such as duplicates, reversals and complements are known to occur in macro-molecules. Models allowing such possibilities can easily be devised. The MML method allows such rival models to be compared on real strings. Given enough data it allows the "true" model to be inferred.

4. Three or More Strings.

An evolutionary tree, or more properly an *evolutionary forest*, can relate three or more strings. A set of strings can be encoded according to such a forest. That forest having the shortest message length is the most plausible and differences in message lengths give odds ratios. Each arc of a tree corresponds to a mutation machine. Note that the models of mutation that we consider are reversible and there is no way to infer a root for a tree unless a constant rate of evolution is assumed or other evidence is found. Note also that for two strings there are just two possible forests; the null-theory implies two singleton trees and the r-theory implies a single tree.

For example, given three strings and the unique unrooted tree over them, there are fourteen observable kinds of triple in a three-way alignment: XXX, XXY, XYX, YXX, XYZ, XX−, X−X, −XX, XY−, X−Y, −XY, X− −, −X−, − −X where X, Y and Z are differing characters and '−' is a null. Using a one-state model, each of the three arcs (or machines) in the tree has three parameters or two degrees of freedom. This gives nine parameters and six degrees of freedom in total; respectively six and four if a constant rate of mutation is assumed. If the parameters are known, they imply probabilities for each kind of triple and allow the strings to be coded according to this tree using a natural extension of the methods described previously. If the parameters are unknown, their values must be inferred from the strings and included in the message.

Given k strings of length n, a naive k-way comparison takes $O(n^k)$ time which is impractical for k>3 or even for k=3 and large n. In practice the probability density for two similar strings (figure 4) is concentrated in a narrow band. All probabilities above some small threshold value

lie in an area much less than n^2. This holds the promise that an approximate algorithm can estimate the message length for a tree in time close to $O(kn^2)$. An approximate r-theory algorithm that is faster than $O(n^2)$ for two similar strings is also likely. These avenues are being investigated.

As before, no forest having a message longer than the null-theory is acceptable. When no forest is given, the search for a good one will have to be guided by heuristics of which several are well known.

5. Summary.

We used minimum message length encoding to infer and compare theories or hypotheses over macro-molecules and other strings. MML encoding explicitly shows the connection between a model of relation on strings and a string comparison algorithm. The parameters of a given model can be inferred from data. The odds-ratio of two theories over a set of strings can be computed. The existence of a good null-theory leads to hypothesis testing. The posterior probability of two strings being related can be calculated.

6. References.

[1] L. Allison, C. S. Wallace & C. N. Yee. When is a string like a string? International Symposium on Artificial Intelligence and Mathematics, Ft.Lauderdale, Jan 3–5 1990.

[2] L. Allison & C. N. Yee. Minimum message length encoding and the comparison of macro-molecules. Bulletin of Mathematical Biology 52(3), 431-453, 1990.

[3] M. J. Bishop & A. E. Friday. Molecular sequences and hominid phylogeny.
Major Topics in Primate and Human Evolution. B. Wood, L. Martin & P. Andrews (eds). CUP, 150-156, 1986.

[4] M. J. Bishop, A. E. Friday & E. A. Thompson. Inference of evolutionary relationships.
in *Nucleic Acid and Protein Sequence Analysis, a Practical Approach.* M. J. Bishop & C. J. Rawlings (eds). IRL Press, 359-385, 1987.

[5] J. Felsenstein. Evolutionary trees from DNA sequences: a maximum likelihood approach. J. Molecular Evolution 17, 368-376, 1981.

[6] T. A. Reichert, D. N. Cohen & K. C. Wong. An application of information theory to genetic mutations and the matching of polypeptide sequences. Journal of Theoretical Biology 42, 245-261, 1973.

[7] P. H. Sellers. The theory and computation of evolutionary distances: pattern recognition. Journal of Algorithms 1, 359-373, 1980.

[8] C. S. Wallace. Classification by minimum message length inference. Symposium on the Theory and Application of Minimal Length Encoding. Stanford, March 27-29 1990.

[9] C. S. Wallace & D. M. Boulton. An information measure for classification. Computer Journal 11(2), 185-194, 1968.

[10] C. S. Wallace & P. R. Freeman. Estimation and inference by compact coding. Journal of the Royal Statistical Society series B 49(3), 240-265, 1987.