2 Pairwise alignment

2.1 Introduction

The most basic sequence analysis task is to ask if two sequences are related. This is usually done by first aligning the sequences (or parts of them) and then deciding whether that alignment is more likely to have occurred because the sequences are related, or just by chance. The key issues are: (1) what sorts of alignment should be considered; (2) the scoring system used to rank alignments; (3) the algorithm used to find optimal (or good) scoring alignments; and (4) the statistical methods used to evaluate the significance of an alignment score. Figure 2.1 shows an example of three pairwise alignments, all to the same region of the human alpha globin protein sequence (SWISS-PROT database identifier HBA\_HUMAN). The central line in each alignment indicates identical positions with letters, and ‘similar’ positions with a plus sign. (‘Similar’ pairs of residues are those which have a positive score in the substitution matrix used to score the alignment; we will discuss substitution matrices shortly.)

(a) HBA\_HUMAN GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKL G+ +VK+HGKKV A+++++AH+D++ +++++LS+LH KL HBB\_HUMAN GNPKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKL

(b) HBA\_HUMAN GSAQVKGHGKKVADALTNAVAHV---D--DMPNALSALSDLHAHKL ++ ++++H+ KV + +A ++ +L+ L+++H+ K LGB2\_LUPLU NNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG

(c) HBA\_HUMAN GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSD----LHAHKL GS+ + G + +D L ++ H+ D+ A +AL D ++AH+ F11G11.2 GSGYLVGDSLTFVDLL—VAQHTADLLAANAALLDEFPQFKAHQE

Figure 2.1 Three sequence alignments to a fragment of human alpha globin. (a) Clear similarity to human beta globin. (b) A structurally plausible alignment to leghaemoglobin from yellow lupin. (c) A spurious highscoring alignment to a nematode glutathione S-transferase homologue named F11G11.2. 12

In the first alignment there are many positions at which the two corresponding residues are identical; many others are functionally conservative, such as the pair D–E towards the end, representing an alignment of an aspartic acid residue with a glutamic acid residue, both negatively charged amino acids. Figure 2.1b also shows a biologically meaningful alignment, in that we know that these two sequences are evolutionarily related, have the same three-dimensional structure, and function in oxygen binding. However, in this case there are many fewer identities, and in a couple of places gaps have been inserted into the alpha globin sequence to maintain the alignment across regions where the leghaemo globin has extra residues. Figure 2.1c shows an alignment with a similar number of identities or conservative changes. However, in this case we are looking at a spurious alignment to a protein that has a completely different structure and function. How are we to distinguish cases like Figure 2.1b from those like Figure 2.1c? This is the challenge for pairwise alignment methods. We must give careful thought to the scoring system we use to evaluate alignments. The next section introduces the issues in how to score alignments, and then there is a series of sections on methods to find the best alignments according to the scoring scheme. The chapter finishes with a discussion of the statistical significance of matches, and more detail on parameter ising the scoring scheme. Even so, it will not always be possible to distinguish true alignments from spurious alignments. For example, it is in fact extremely difficult to find significant similarity between the lupin leghaemo globin and human alpha globin in Figure 2.1b using pairwise alignment methods.

2.2 The scoring model

When we compare sequences, we are looking for evidence that they have diverged from a common ancestor by a process of mutation and selection. The basic mutational processes that are considered are substitutions, which change residues in a sequence, and insertions and deletions, which add or remove residues. Insertions and deletions are together referred to as gaps. Natural selection has an effect on this process by screening the mutations, so that some sorts of change may be seen more than others. The total score we assign to an alignment will be a sum of terms for each aligned pair of residues, plus terms for each gap. In our probabilistic interpretation, this will correspond to the logarithm of the relative likelihood that the sequences are related, compared to being unrelated. Informally, we expect identities and conservative substitutions to be more likely in alignments than we expect by chance, and so to contribute positive score terms; and non-conservative changes are expected to be observed less frequently in real alignments than we expect by chance, and so these contribute negative score terms.

Using an additive scoring scheme corresponds to an assumption that we can consider mutations at different sites in a sequence to have occurred independently (treating a gap of arbitrary length as a single mutation). All the algorithms in this chapter for finding optimal alignments depend on such a scoring scheme. The assumption of independence appears to be a reasonable approximation for DNA and protein sequences, although we know that interactions between residues play a very critical role in determining protein structure. However, it is seriously inaccurate for structural RNAs, where base pairing introduces very important longrange dependencies. It is possible to take these dependencies into account, but doing so gives rise to significant computational complexities; we will delay the subject of RNA alignment until the end of the book (Chapter 10).

Substitution matrices

We need score terms for each aligned residue pair. A biologist with a good intuition for proteins could invent a set of 210 scoring terms for all possible pairs of amino acids, but it is extremely useful to have a guiding theory for what the scores mean. We will derive substitution scores from a probabilistic model. First, let us establish some notation.We will be considering a pair of sequences, x and y, of lengths n and m, respectively. Let xi be the ith symbol in x and yj be the j th symbol of y. These symbols will come from some alphabet A; in the case of DNA this will be the four bases {A, G, C, T}, and in the case of proteins the twenty amino acids. We denote symbols from this alphabet by lower-case letters like a,b. For now we will only consider ungapped global pairwise alignments: that is, two completely aligned equal-length sequences as in Figure 2.1a. Given a pair of aligned sequences, we want to assign a score to the alignment that gives a measure of the relative likelihood that the sequences are related as opposed to being unrelated.We do this by having models that assign a probability to the alignment in each of the two cases; we then consider the ratio of the two probabilities.

The unrelated or random model R is simplest. It assumes that letter a occurs independently with some frequency qa, and hence the probability of the two sequences is just the product of the probabilities of each amino acid:

P(x, y|R) = i qxi j qyj . (2.1)

In the alternative match model M, aligned pairs of residues occur with a joint probability pab. This value pab can be thought of as the probability that the residues a and b have each independently been derived from some unknown original residue c in their common ancestor (c might be the same as a and/or b). This gives a probability for the whole alignment of P(x, y|M) =i pxi yi . The ratio of these two likelihoods is known as the odds ratio: P(x, y|M) P(x, y|R)=i pxi yi i qxi i qyi=i pxi yi qxi qyi. In order to arrive at an additive scoring system, we take the logarithm of this ratio, known as the log-odds ratio: S = is(xi , yi ), (2.2)

where s(a,b) = log pab qaqb (2.3) is the log likelihood ratio of the residue pair (a,b) occurring as an aligned pair, as opposed to an unaligned pair.

As we wanted, equation (2.2) is a sum of individual scores s(a,b) for each aligned pair of residues. The s(a,b) scores can be arranged in a matrix. For proteins, for instance, they form a 20×20 matrix, with s(ai ,aj ) in position i , j in the matrix, where ai ,aj are the ith and j th amino acids (in some numbering). This is known as a score matrix or a substitution matrix. An example of a substitution matrix derived essentially as above is the BLOSUM50 matrix, shown in Figure 2.2. We can use these values to score Figure 2.1a and get a score of 130. Another commonly used set of substitution matrices are called the PAM matrices. A detailed description of the way that the BLOSUM and PAM matrices are derived is given at the end of the chapter.

An important result is that even if an intuitive biologist were to write down an ad hoc substitution matrix, the substitution matrix implies ‘target frequencies’ pab according to the above theory [Altschul 1991]. Any substitution matrix is making a statement about the probability of observing ab pairs in real alignments.

2.3 Alignment algorithms

Given a scoring system, we need to have an algorithm for finding an optimal alignment for a pair of sequences.Where both sequences have the same length n, there is only one possible global alignment of the complete sequences, but things become more complicated once gaps are allowed (or once we start looking for local alignments between subsequences of two sequences). There are 2nn= (2n)!(n!)222n√πn (2.7) possible global alignments between two sequences of length n. It is clearly not computationally feasible to enumerate all these, even for moderate values of n. The algorithm for finding optimal alignments given an additive alignment score of the type we have described is called dynamic programming. Dynamic programming algorithms are central to computational sequence analysis. All the remaining chapters in this book except the last, which covers mathematical methods, make use of dynamic programming algorithms. The simplest dynamic programming alignment algorithms to understand are pairwise sequence alignment algorithms. The reader should be sure to understand this section, because it lays an important foundation for the book as a whole. Dynamic programming algorithms are guaranteed to find the optimal scoring alignment or set of alignments. In most cases heuristic methods have also been developed to perform the same type of search. These can be very fast, but they make additional assumptions and will miss the best match for some sequence pairs. We will briefly discuss a few approaches to heuristic searching later in the chapter. Because we introduced the scoring scheme as a log-odds ratio, better alignments will have higher scores, and so we want to maximise the score to find the optimal alignment. Sometimes scores are assigned by other means and interpreted as costs or edit distances, in which case we would seek to minimise the cost of an alignment. Both approaches have been used in the biological sequence comparison literature. Dynamic programming algorithms apply to either case; the differences are trivial exchanges of ‘min’ for ‘max’. We introduce four basic types of alignment. The type of alignment that we want to look for depends on the source of the sequences that we want to align. For each alignment type there is a slightly different dynamic programming algorithm. In this section, we will only describe pairwise alignment for linear gap scores, with cost d per gap residue. However, the algorithms we introduce here easily extend to more complex gap models, as we will see later in the chapter. We will use two short amino acid sequences to illustrate the various alignment methods, HEAGAWGHEE and PAWHEAE. To score the alignments, we use the BLOSUM50 score matrix, and a gap cost per unaligned residue of d = ?8. Figure 2.3 shows a matrix si j of the local score s(xi , yj ) of aligning each residue pair from the two example sequences. Identical or conserved residue pairs are highlighted in bold. Informally, the goal of an alignment algorithm is to incorporate as many of these positively scoring pairs as possible into the alignment, while mini-mising the cost from unconserved residue pairs, gaps, and other constraints.

3 Markov chains and hidden Markov models

Having introduced some methods for pairwise alignment in Chapter 2, the emphasiswill switch in this chapter to questions about a single sequence. The mainaim of the chapter is to develop the theory for a very general form of probabilisticmodel for sequences of symbols, called a hidden Markov model (abbreviatedHMM). The types of question we can use HMMs and their simpler cousins,Markov models, to consider are: ‘Does this sequence belong to a particular family?’or ‘Assuming the sequence does come from some family, what can we sayabout its internal structure?’ An example of the second type of problem would beto try to identify alpha helix or beta sheet regions in a protein sequence.As well as giving examples from the biological sequence world, we also givethe mathematics and algorithms for many of the operations on HMMs in a moregeneral form. These methods, or close analogues of them, are applied in manyother sections of the book. This chapter therefore contains a fairly large amountof mathematically technical material. We have tried to organise it so that thefirst half, approximately, leads the reader through the essential algorithms usinga single biological example. In the later sections we introduce a variety of otherexamples to illustrate more complex extensions of the basic approaches.In the next chapter, we will see how HMMs can also be applied to the typesof alignment problem discussed in Chapter 2, in Chapter 5 they are applied tosearching databases for protein families, and in Chapter 6 to alignment of severalsequences simultaneously. In fact, the search and alignment applications constituteprobably the best-known use of HMMs for biological sequence analysis.However, we present HMM theory here in a less specialised context in order toemphasise its much broader applicability, which goes far beyond that of sequencealignment.The overwhelming majority of papers on HMMs belong to the speech recognitionliterature, where they were applied first in the early 1970s. One of thebest general introductions to the subject is the review by Rabiner [1989], whichalso covers the history of the topic. Although there will be quite a bit of overlapbetween that and the present chapter, there will be important differences infocus.473 Markov chains and hidden Markov modelsBefore going on to introduce HMMs for biological sequence analysis, it isperhaps interesting to look briefly at how they are used for speech recognition[Rabiner & Juang 1993]. After recording, a speech signal is divided into pieces(called frames) of 10–20 milliseconds. After some preprocessing each frame isassigned to one out of a large number of predefined categories by a process knownas vector quantisation. Typically there are 256 such categories. The speech signalis then represented as a long sequence of category labels and from that the speechrecogniser has to find out what sequence of phonemes (or words) was spoken.The problems are that there are variations in the actual sound uttered, and thereare also variations in the time taken to say the various parts of the word.Many problems in biological sequence analysis have the same structure: basedon a sequence of symbols from some alphabet, find out what the sequencerepresents. For proteins the sequences consist of symbols from the alphabet of 20amino acids, and we typically want to know what protein family a given sequencebelongs to. Here the primary sequence of amino acids is analogous to the speechsignal and the protein family to the spoken word it represents. The time-variationof the speech signal corresponds to having insertions and deletions in the proteinsequences.Let us turn to a simpler example, which we will use to introduce first standardMarkov models, of the non-hidden variety, then a simple hidden Markov model.Example: CpG islandsIn the human genome wherever the dinucleotide CG occurs (frequently writtenCpG to distinguish it from the C-G base pair across the two strands) the C nucleotide(cytosine) is typically chemically modified by methylation. There is arelatively high chance of this methyl-C mutating into a T, with the consequencethat in general CpGdinucleotides are rarer in the genome than would be expectedfrom the independent probabilities of C and G. For biologically important reasonsthe methylation process is suppressed in short stretches of the genome, suchas around the promoters or ‘start’ regions of many genes. In these regions weseemanymore CpG dinucleotides than elsewhere, and in fact more C and G nucleotidesin general. Such regions are called CpG islands [Bird 1987]. They aretypically a few hundred to a few thousand bases long.We will consider two questions: Given a short stretch of genomic sequence,how would we decide if it comes from a CpG island or not? Second, given a longpiece of sequence, how would we find the CpG islands in it, if there are any? Letus start with the first question.3.1 Markov chainsWhat sort of probabilistic model might we use for CpG island regions? Weknow that dinucleotides are important. We therefore want a model that generates?

3.1 Markov chainssequences in which the probability of a symbol depends on the previous symbol.The simplest such model is a classical Markov chain. We like to show a Markovchain graphically as a collection of ‘states’, each of which corresponds to a particularresidue, with arrows between the states. A Markov chain for DNA can bedrawn like this:TAGCwhere we see a state for each of the four letters A, C, G, and Tin the DNA alphabet.A probability parameter is associated with each arrow in the figure, whichdetermines the probability of a certain residue following another residue, or onestate following another state. These probability parameters are called the transitionprobabilities, which we will write ast :ast =P(xi =t|xi.1 =s).(3.1)For any probabilistic model of sequences we can write the probability of thesequence asP(x)=P(xL , xL.1, ..., x1)=P(xL |xL.1, ..., x1)P(xL.1|xL.2, ..., x1) ···P(x1)by applying P(X, Y ) =P(X |Y )P(Y ) many times. The key property of a Markovchain is that the probability of each symbol xi depends only on the value of thepreceding symbol xi.1, not on the entire previous sequence, i.e. P(xi |xi.1, ..., x1)=P(xi |xi.1) =. The previous equation therefore becomesaxi.1 xiP(x) =P(xL |xL.1)P(xL.1|xL.2) ···P(x2|x1)P(x1)L.=P(x1) axi.1 xi . (3.2)i=2Although we have derived this equation in the context of CpG islands in DNAsequences, it is in fact the general equation for the probability of a specific sequencefrom any Markov chain. There is a large literature on Markov chains, seefor example Cox & Miller [1965].?

3 Markov chains and hidden Markov models

AGCBE

Figure 3.1 Begin and end states can be added to a Markov chain (greymodel) for modelling both ends of a sequence.Exercise3.1The sum of the probabilities of all possible sequences of length L can bewritten (using (3.2))L. ..P(x) =...P(x1) axi.1xi .{x}x1 x2 xL i=2Show that this is equal to 1.Modelling the beginning and end of sequencesNotice that as well as specifying the transition probabilities we must also give theprobability P(x1) of starting in a particular state. To avoid the inhomogeneity of(3.2) introduced by the starting probabilities, it is possible to add an extra beginstate to the model. At the same time we add a letter to the alphabet, which wewill call B. By defining x0 =Bthe beginning of a sequence is also included in(3.2), so for instance the probability of the first letter in the sequence isP(x1 =s) =aBs .Similarly we can add a symbol Eto the end of a sequence to ensure the end ismodelled. Then the probability of ending with residue t isP(E|xL =t) =atE.To match the new symbols, we add begin and end states to the DNA model (seeFigure 3.1). In fact, we need not explicitly add any letters to the alphabet, butinstead can treat the two new states as ‘silent’ states that just serve as start andend points.Traditionally the end of a sequence is not modelled in Markov chains; it isassumed that the sequence can end anywhere. The effect of adding an explicit?

3.1 Markov chainsend state is to model a distribution of lengths of the sequence. This way the modeldefines a probability distribution over all possible sequences (of any length). Thedistribution over lengths decays exponentially; see the exercise below.Exercises3.2Assume that the model has an end state, and that the transition from anystate to the end state has probability τ. Show that the sum of the probabilities(3.2) over all sequences of length L (and properly terminating bymaking a transition to the end state) is τ(1 .τ)L.1.3.3Show that the sum of the probability over all possible sequences of anylength is 1. This proves that the Markov chain really describes a properprobability distribution over the whole space of sequences. (Hint: Use∞i

the result that, for 0 <x <1, i=0 x=1/(1 .x).)Using Markov chains for discriminationA primary use for equation (3.2) is to calculate the values for a likelihood ratiotest. We illustrate this here using real data for the CpG island example. From a setof human DNA sequences we extracted a total of 48 putative CpG islands and derivedtwo Markov chain models, one for the regions labelled as CpG islands (the‘+’ model) and the other from the remainder of the sequence (the ‘.’ model).The transition probabilities for each model were set using the equation+

c

+sta=.,(3.3)st +t.c

st.and its analogue for ast ., where c+is the number of times letter t followed lettersts in the labelled regions. These are the maximum likelihood (ML) estimators forthe transition probabilities, as described in Chapter 1.(In this case there were almost 60 000 nucleotides, and ML estimators are adequate.If the number of counts of each type had been small, then a Bayesian estimationprocess would have been more appropriate, as discussed in Chapter 11and below for HMMs.) The resulting tables are+ ACGT .ACGTA 0.180 0.274 0.426 0.120 A 0.300 0.205 0.285 0.210C 0.171 0.368 0.274 0.188 C 0.322 0.298 0.078 0.302G 0.161 0.339 0.375 0.125 G 0.248 0.246 0.298 0.208T 0.079 0.355 0.384 0.182 T 0.177 0.239 0.292 0.292where the first row in each case contains the frequencies with which an A isfollowed by each of the four bases, and so on for the other rows, so each row?

3 Markov chains and hidden Markov modelssums to one. These numbers are not the same; for example, Gfollowing Ais muchmore common than Tfollowing A. Notice also that the tables are asymmetric. Inboth tables the probability for Gfollowing Cis lower than that for Cfollowing G,although the effect is stronger in the ‘.’ table, as expected.To use these models for discrimination, we calculate the log-odds ratioL +

P(x|model +) a

xi.1xi

S(x) =log =log.

P(x|model .) a

xi.1xi

i=1L=βxi.1 xii=1where x is the sequence and βxi.1 xi are the log likelihood ratios of correspondingtransition probabilities. A table for βis given below in bits:1βA CG TA .0.740 0.419 0.580 .0.803C .0.913 0.302 1.812 .0.685G .0.624 0.461 0.331 .0.730T .1.169 0.573 0.393 .0.679Figure 3.2 shows the distribution of scores, S(x), normalised by dividing bytheir length, i.e. as an average number of bits per molecule. If we had not normalisedby length, the distribution would have been much more spread out.We see a reasonable discrimination between regions labelled CpG island andother regions. The discrimination is not very much influenced by the length normalisation.If we wanted to pursue this further and investigate the cases of misclassification,it is worth remembering that the error could either be due to aninadequate or incorrectly parameterised model, or to mislabelling of the trainingdata.3.2 Hidden Markov modelsThere are a number of extensions to classical Markov chains, which we will comeback to later in the chapter. Here, however, we will proceed immediately to hiddenMarkov models. We will motivate this by turning to the second of the twoquestions posed initially for CpG islands: How do we find them in a long unannotatedsequence? The Markov chain models that we have just built could be usedfor this purpose, by calculating the log-odds score for a window of, say, 100 nucleotidesaround every nucleotide in the sequence and plotting it. We would then1 Base 2 logarithms were used, in which case the unit is called a bit. See Chapter 11.?

3.2 Hidden Markov models1050-0.4 -0.3 -0.2 -0.1 0 0.1 0.2 0.3 0.4BitsFigure 3.2 The histogram of the length-normalised scores for all the sequences.CpG islands are shown with dark grey and non-CpG with lightgrey.A+C+ G+T+A. C. G.T.Figure 3.3 An HMM for CpG islands. In addition to the transitions shown,there is also a complete set of transitions within each set, as in the earliersimple Markov chains.expect CpG islands to stand out with positive values. However, this is somewhatunsatisfactory if we believe that in fact CpG islands have sharp boundaries, andare of variable length. Why use a window size of 100? A more satisfactory approachis to build a single model for the entire sequence that incorporates bothMarkov chains.To simulate in one model the ‘islands’ in a ‘sea’ of non-island genomic sequence,we want to have both the Markov chains of the last section present in thesame model, with a small probability of switching from one chain to the otherat each transition point. However, this introduces the complication that we nowhave two states corresponding to each nucleotide symbol. We resolve this by relabellingthe states. We now have A+,C+,G+and T+which emit A, C, Gand Trespectively in CpG island regions, and A.,C.,G.and T.correspondingly innon-island regions; see Figure 3.3.?

3 Markov chains and hidden Markov modelsThe transition probabilities in this model are set so that within each group theyare close to the transition probabilities of the original component model, but thereis also a small but finite chance of switching into the other component. Overallthere is more chance of switching from ‘+’to‘.’ than vice versa, so if left torun free, the model will spend more of its time in the ‘.’ non-island states thanin the island states.The relabelling is the critical step. The essential difference between a Markovchain and a hidden Markov model is that for a hidden Markov model there is nota one-to-one correspondence between the states and the symbols. It is no longerpossible to tell what state the model was in when xi was generated just by lookingat xi . In our example there is no way to tell by looking at a single symbol C inisolation whether it was emitted by state C+or state C.Formal definition of an HMMLet us formalise the notation for hidden Markov models, and derive the probabilityof a particular sequence of states and symbols. We now need to distinguish thesequence of states from the sequence of symbols. Let us call the state sequencethe path, π. The path itself follows a simple Markov chain, so the probability ofa state depends only on the previous state. The ith state in the path is called πi .The chain is characterised by parametersakl =P(πi =l|πi.1 =k). (3.4)To model the beginning of the process we introduce a begin state, as was introducedearlier to model the beginning of sequences in Markov chains (Figure 3.1).The transition probability a0k from this begin state to state k can be thought of asthe probability of starting in state k. It is also possible to model ends as beforeby always ending a state sequence with a transition into an end state. For conveniencewe label both begin and end states as 0 (there is no conflict here becauseyou can only transit out of the begin state, and only into the end state, so variablesare not used more than once).Because we have decoupled the symbols b from the states k, we must introducea new set of parameters for the model, ek(b). For our CpG model each state isassociated with a single symbol, but this is not a requirement; in general a statecan produce a symbol from a distribution over all possible symbols. We thereforedefineek (b) =P(xi =b|πi =k), (3.5)the probability that symbol b is seen when in state k. These are known as theemission probabilities.For our CpG island model the emission probabilities are all 0 or 1. To illustrateemission probabilities we reintroduce here the casino example from Chapter 1.?

3.2 Hidden Markov models

Example: The occasionally dishonest casino, part 1Let us consider an example from Chapter 1. In a casino they use a fair die most ofthe time, but occasionally they switch to a loaded die. The loaded die has probability0.5 of a six and probability 0.1 for the numbers one to five. Assume thatthe casino switches from a fair to a loaded die with probability 0.05 before eachroll, and that the probability of switching back is 0.1. Then the switch betweendice is a Markov process. In each state of the Markov process the outcomes of aroll have different probabilities, and thus the whole processs is an example of ahidden Markov model. We can draw it like this:Fair1: 1/62: 1/63: 1/64: 1/65: 1/66: 1/6Loaded1: 1/102: 1/103: 1/104: 1/105: 1/106: 1/20.95 0.90.050.1where the emission probabilities e() are shown in the state boxes.What is hidden in the above model? If you can just see a sequence of rolls (thesequence of observations) you do not know which rolls used a loaded die andwhich used a fair one, because that is kept secret by the casino; that is, the statesequence is hidden. In a Markov chain you always know exactly in which state agiven observation belongs. Obviously the casino wouldn’t tell you that they useloaded dice and what the various probabilities are. Yet for this more complicatedsituation, which we will return to later, it is possible to estimate the probabilitiesin the above HMM (once you have a suspicion that they use two different dice).The reason for the name emission probabilities is that it is often convenientto think of HMMs as generative models, that generate or emit sequences. Forinstance we can generate random sequences of rolls from the model of the fair/loadeddice above by simulating the successive choices of die, then rolls of thechosen die. More generally a sequence can be generated from an HMM as follows:First a state π1 is chosen according to the probabilities a0i . In that state anobservation is emitted according to the distribution eπ1 for that state. Then a newstate π2 is chosen according to the transition probabilities aπ1i and so forth. Thisway a sequence of random, artificial observations are generated. Therefore, wewill sometimes say things like P(x) is the probability that x was generated bythe model.It is now easy to write down the joint probability of an observed sequence xand a state sequence π:LP(x, π) =a0π1 eπi (xi )aπi πi+1, (3.6)i=1?

3 Markov chains and hidden Markov models

where we require πL+1 =0. For example, the probability of sequence CGCGbeingemitted by the state sequence (C+,G.,C.,G+) in our model isa0,C+×1 ×aC+,G.×1 ×aG.,C.×1 ×aC.,G+×1 ×aG+,0.Equation (3.6) is the HMM analogue of equation (3.2). However, it is not souseful in practice because in general we do not know the path. In the followingsections we describe how to estimate the path, either by finding the most likelyone, or alternatively by using an a posteriori distribution over states. Then we goon to show how to estimate the parameters for an HMM.Most probable state path: the Viterbi algorithmAlthough it is no longer possible to tell what state the system is in by looking atthe corresponding symbol, it is often the sequence of underlying states that we areinterested in. To find out what the observation sequence ‘means’ by consideringthe underlying states is called decoding in the jargon of speech recognition. Thereare several approaches to decoding. Here we will describe the most commonone, called the Viterbi algorithm. It is a dynamic programming algorithm closelyrelated to the ones covered in Chapter 2.In general there may now be many state sequences that could give rise toany particular sequence of symbols. For example, in our CpG model the statesequences (C+,G+,C+,G+), (C.,G.,C.,G.) and (C+,G.,C+,G.) would allgenerate the symbol sequence CGCG. However, they do so with very differentprobabilities. The third is the product of multiple small probabilities of switchingback and forth between the components, and hence is much smaller than the firsttwo. The second is itself significantly smaller than the first because it containstwo Cto Gtransitions which are significantly less probable in the ‘.’ componentthan in the ‘+’ component. Of these three choices, therefore, it is most likely thatthe sequence CGCGcame from a set of ‘+’ states.A predicted path through the HMM will tell us which part of the sequenceis predicted as a CpG island, because we assumed above that each state wasassigned to model either CpG islands or other regions. If we are to choose justone path for our prediction, perhaps the one with the highest probability shouldbe chosen,.π=argmax P(x, π). (3.7)πThe most probable path π.can be found recursively. Suppose the probabilityvk (i) of the most probable path ending in state k with observation i is known forall the states k. Then these probabilities can be calculated for observation xi+1 asvl (i +1) =el (xi+1)max(vk (i)akl ). (3.8)

k?

3.2 Hidden Markov model

svCGC GB1 0 0 0 0A+0 0 0 0 0C+0 0.13 0 0.012 0G+0 0 0.034 0 0.0032T+0 0 0 0 0A.0 0 0 0 0C.0 0.13 0 0.0026 0G.0 0 0.010 0 0.00021T.0 0 0 0 0Figure 3.4 For the model of CpG islands shown in Figure 3.3 and the se

quence CGCG, this is the resulting table of v. The most probable path isshown with bold face.All sequences have to start in state 0 (the begin state), so the initial condition isthat v0(0) =1. By keeping pointers backwards, the actual state sequence can befound by backtracking. The full algorithm is:Algorithm: ViterbiInitialisation (i =0):v0(0) =1, vk (0) =0for k >0.Recursion (i =1 ...L): vl (i) =el (xi )maxk (vk (i .1)akl );ptri (l) =argmaxk (vk (i .1)akl ).Termination:P(x, π.) =maxk(vk (L)ak0);π.=argmaxk (vk (L)ak0).

LTraceback (i =L ...1): π.=ptri (πi .).i.1Note that an end state is assumed, which is the reason for ak0 in the terminationstep. If ends are not modelled, this a will disappear.There are some implementational issues both for the Viterbi algorithm and thealgorithms described later. The most severe practical problem is that multiplyingmany probabilities always yields very small numbers that will give underflowerrors on any computer. For this reason the Viterbi algorithm should always bedone in log space, i.e. calculating log(vl (i)), which will make the products becomesums and the numbers stay reasonable. This is discussed in Section 3.6.Figure 3.4 shows the full table of values of vfor the sequence CGCG and theCpG island model. When we apply the same algorithm to a longer sequence thederived optimal path π.will switch between the ‘+’ and the ‘.’ components ofthe model, and thereby give the precise boundaries of the predicted CpG islandregions.Example: The occasionally dishonest casino, part 2For a sequence of dice rolls we can now find the most probable path throughthe model shown on p. 55. A total of 300 random rolls were generated from the?

3 Markov chains and hidden Markov modelsRolls 315116246446644245311321631164152133625144543631656626566666Die FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLViterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLRolls 651166453132651245636664631636663162326455236266666625151631Die LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLFFFLLLLLLLLLLLLLLFFFFFFFFFViterbi LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFFRolls 222555441666566563564324364131513465146353411126414626253356Die FFFFFFFFLLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLViterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLRolls 366163666466232534413661661163252562462255265252266435353336Die LLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFViterbi LLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFRolls 233121625364414432335163243633665562466662632666612355245242Die FFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFFViterbi FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLFFFFFFFFFFFFigure 3.5 The numbers show 300 rolls of a die as described in the exam

ple. Below is shown which die was actually used for that roll (F for fair andL for loaded). Under that the prediction by the Viterbi algorithm is shown.model as described earlier. Each roll was generated either with the fair die (F)or the loaded one (L), as shown below the outcome of the roll in Figure 3.5.The Viterbi algorithm was used to predict the state sequence, i.e. which die wasused for each of the rolls. Generally, as you can see, the Viterbi algorithm hasrecovered the state sequence fairly well.Exercise.

3.4 Show that π=argmax P(π|x) is equivalent to (3.7).πThe forward algorithmFor Markov chains we calculated the probability of a sequence, P(x), with equation(3.2). The resulting values were used to distinguish between CpG islandsand other DNA for instance. We want to be able to calculate this probability foran HMM as well. Because many different state paths can give rise to the samesequence x, we must add the probabilities for all possible paths to obtain the fullprobability of x,P(x) =P(x, π). (3.9)πThe number of possible paths πincreases exponentially with the length of thesequence, so brute force evaluation of (3.9) by enumerating all paths is not practical.One approach is to use equation (3.6) evaluated at the most probable pathπ.obtained in the last section as an approximation to P(x). This implicitly assumesthat the only path with significant probability is π., a somewhat startling?

3.2 Hidden Markov modelsassumption which however in many cases is surprisingly good. In fact the approximationis unnecessary, because the full probability can itself be calculatedby a similar dynamic programming procedure to the Viterbi algorithm, replacingthe maximisation steps with sums. This is called the forward algorithm.The quantity corresponding to the Viterbi variable vk (i) in the forward algorithmisfk (i) =P(x1 ...xi , πi =k), (3.10)which is the probability of the observed sequence up to and including xi , requiringthat πi =k. The recursion equation isfl (i +1) =el (xi+1) fk (i)akl . (3.11)kThe full algorithm is:Algorithm: Forward algorithmInitialisation (i =0): f0(0) =1, fk (0) =0for k >0.Recursion (i =1 ...L): fl (i) =el (xi ) fk (i .1)akl .kTermination: P(x) =fk (L)ak0. .kLike the Viterbi algorithm, the forward algorithm (and the backward algorithmin the next section) can give underflow errors when implemented on a computer.Again this can be solved by working in log space, although not as elegantly asfor Viterbi. Alternatively a scaling method can be used. Both approaches are describedin Section 3.6.As well as their use in the forward algorithm, the quantities fk(i) have a numberof other uses, including those described in the next two sections.The backward algorithm and posterior state probabilitiesThe Viterbi algorithm finds the most probable path through the model, but aswe remarked at the time, this may not always be the most appropriate basis forfurther inference about the sequence. We might for instance want to know whatthe most probable state is for an observation xi . More generally, we may want theprobability that observation xi came from state k given the observed sequence,i.e. P(πi =k|x). This is the posterior probability of state k at time i when theemitted sequence is known.Our approach to the posterior probability is a little indirect. We first calculatethe probability of producing the entire observed sequence with the ith symbol?

3 Markov chains and hidden Markov modelsbeing produced by state k:P(x, πi =k) =P(x1 ...xi , πi =k)P(xi+1 ...xL |x1 ...xi , πi =k)=P(x1 ...xi , πi =k)P(xi+1 ...xL |πi =k), (3.12)the second row following because everything after k only depends on the state atk. The first term in this is recognised as fk (i) from (3.10) that was calculated bythe forward algorithm of the previous section. The second term is called bk (i),bk (i) =P(xi+1 ...xL |πi =k). (3.13)It is analogous to the forward variable, but instead obtained by a backward recursionstarting at the end of the sequence:Algorithm: Backward algorithmInitialisation (i =L): bk (L) =ak0 for all k.Recursion (i =L .1, ...,1): bk (i) =aklel (xi+1)bl (i +1).lTermination: P(x) =a0lel (x1)bl (1). .lThe termination step is rarely needed, because P(x) is usually found by theforward algorithm, and it is just shown for completeness.Equation (3.12) can now be written as P(x, πi =k) =fk(i)bk (i), and from itwe obtain the required posterior probabilities by straightforward conditioning,fk (i)bk (i)

P(πi =k|x) =, (3.14)P(x)where P(x) is the result of the forward (or backward) calculation.Example: The occasionally dishonest casino, part 3In Figure 3.6 the posterior probability for the die being fair is shown for thesequence of rolls shown in Figure 3.5. Notice that the posterior probability doesnot reflect which die was actually used in some places. This is to be expected,simply because a misleading sequence of rolls can occur at random.Posterior decodingA major use of the P(πi =k|x) is for two alternative forms of decoding in additionto the Viterbi decoding we introduced in the previous section. These areparticularly useful when many different paths have almost the same probabilityas the most probable one, because then it is not well justified to consider only themost probable path.?

3.2 Hidden Markov modelsP(fair)0 50 100 150 200 250300Figure 3.6 The posterior probability of being in the state corresponding tothe fair die in the casino example. The x axis shows the number of the roll.The shaded areas show when the roll was generated by the loaded die.The first approach is to define a state sequence .πi that can be used in place ofπ.i ,π.i =argmax P(πi =k|x). (3.15)kAs suggested by its definition, this state sequence may be more appropriate whenwe are interested in the state assignment at a particular point i, rather than thecomplete path. In fact, the state sequence defined by .πi may not be particularlylikely as a path through the entire model; it may even not be a legitimate path atall if some transitions are not permitted, which is normally the case.The second, and perhaps more important, new decoding approach arises whenit is not the state sequence itself which is of interest, but some other propertyderived from it. Assume we have a function g(k) defined on the states. The naturalvalue to look at then isG(i|x) =P(πi =k|x)g(k). (3.16)kAn important special case of this is where g(k) takes the value 1 for a subsetof the states and 0 for the rest. In this case, G(i|x) is the posterior probabilityof the symbol i coming from a state in the specified set. For example, with ourCpG island model, what really concerns us is whether a base is part of an islandor not. For this purpose we want to define g(k) =1for k ∈{A+,C+,G+,T+}

and g(k) =0for k ∈{A.,C.,G.,T.}. Then G(i|x) is precisely the posteriorprobability according to the model that base i is in a CpG island.In the case where we have a labelling of the states defining a partition of them(as we in fact have with the CpG island model, labelling them as ‘+’or‘.’)it is possible to use (3.16) to find the most probable label at each position of thesequence. This is not quite the most probable global labelling of a given sequence.That, however, is not entirely straightforward. See Schwartz & Chow [1990] andKrogh [1997b] for further discussion of this.Example: Prediction of CpG islandsNow CpG islands can be predicted from our model. By the Viterbi algorithm wecan find the most probable path through the model. When this path goes through?

3 Markov chains and hidden Markov modelsP(fair)0 100 200 300 400 500 600 700 800 900 1000Figure 3.7 The posterior probability of the die being fair, but using proba

bility 0.01 for switching to the loaded die (cf. Figure 3.6).the +states, a CpG island is predicted. For the set of 41 sequences, each witha putative CpG island, all the islands are found except for two (false negatives),and 121 new ones are predicted (false positives). The real CpG islands are quitelong (of the order of 1000 bases), whereas the predicted ones are short, and aCpG island is usually predicted as several short ones. By applying the two simplepost-processing steps (1) concatenate predictions less than 500 bases apart (2)discard predictions shorter than 500, the number of false positives are reduced to67.Using posterior decoding, the same two CpG islands are missed and 236 falsepositives are predicted. Using the same post-processing as above this number isreduced to 83. For this problem, there is not a big difference between the twomethods, except that the posterior decoding predicts even more very short islands.It is possible that some of the false positives are real CpG islands. The twofalse negatives are perhaps wrongly labelled, but it is also possible that a moresophisticated model is needed for capturing all the features of these signals.Example: The occasionally dishonest casino, part 4The model for the casino is changed, so there is only a probability of 0.01 forswitching from fair to loaded. Obviously the probability of staying with the fairdie must then be 0.99, but all other probabilities are unchanged. From this model1000 random rolls are generated. From these rolls the most probable path foundby the Viterbi algorithm never visits the loaded die state. In Figure 3.7 the posteriorprobability for the dice being fair is shown for these rolls. Although not perfect,posterior decoding would predict something reasonably close to the truth.3.3 Parameter estimation for HMMsProbably the most difficult problem faced when using HMMs is that of specifyingthe model in the first place. There are two parts to this: the design of thestructure, i.e. what states there are and how they are connected, and the assignmentof parameter values, the transition and emission probabilities akl and ek (b).In this section we will discuss the parameter estimation problem, for which there?

3.3 Parameter estimation for HMMsis a well-developed theory. In the next section we will consider model structuredesign, which is more of an art.The framework in which we will be working is to assume that we have a set ofexample sequences of the type that we want the model to fit well, known as trainingsequences. Let these be x1, ..., xn. We assume that they are independent, andthus that the joint probability of all the sequences given a particular assignmentof parameters is the product of the probabilities of the individual sequences. Infact, we work in log space, and so with the log probability of the sequences,n11

l(x, ..., xn|θ) =log P(x, ..., xn|θ) =log P(xj |θ), (3.17)j=1where θrepresents the entire current set of values of the parameters in the model(all the as and es). This is equal to the log likelihood of the model; see Chapter 11.Estimation when the state sequence is knownJust as it was easier to write down the probability of a sequence when the pathwas known, so it is easier to estimate the probability parameters when the pathsare known for all the examples. Frequently this is the case. An example wouldbe if we were given a set of genomic sequences in which the CpG islands werealready labelled, based on experimental data. Other examples would be for anHMM that predicted secondary structure, with training sequences obtained fromthe set of proteins with known structures, or for an HMM predicting genes fromgenomic sequences, where the transcript structure has been determined by cDNAsequencing.When all the paths are known, we can count the number of times each particulartransition or emission is used in the set of training sequences. Let these be Akland Ek(b). Then, as shown in Chapter 11, the maximum likelihood estimators forakl and ek (b)are givenbyakl =Akl.l.Akl.and ek (b) =Ek (b).b.Ek (b). (3.18)The estimation equation for akl is exactly the same as for a simple Markov chain.As always, maximum likelihood estimators are vulnerable to overfitting if thereare insufficient data. Indeed if there is a state k that is never used in the set ofexample sequences, then the estimation equations are undefined for that state,because both the numerator and denominator will have value zero. To avoid suchproblems it is preferable to add predetermined pseudocounts to the Akl and Ek (b)before using (3.18).Akl =number of transitions k to l in training data +rkl ,Ek (b) =number of emissions of b from k in training data +rk (b).?

3 Markov chains and hidden Markov modelsThe pseudocounts rkl and rk(b) should reflect our prior biases about the probabilityvalues. In fact they have a natural probabilistic interpretation as the parametersof Bayesian Dirichlet prior distributions on the probabilities for each state(see Chapter 11). They must be positive, but do not need to be integers. Smalltotal values l.rkl.or b.rk (b) indicate weak prior knowledge, whereas largertotal values indicate more definite prior knowledge, which requires more data tomodify it.Estimation when paths are unknown: Baum–Welch and ViterbitrainingWhen the paths are unknown for the training sequences, there is no longer adirect closed-form equation for the estimated parameter values, and some form ofiterative procedure must be used. All the standard algorithms for optimisation ofcontinuous functions can be used; see for example Press et al. [1992]. However,there is a particular iteration method that is standardly used, known as the Baum–Welch algorithm [Baum 1972]. This has a natural probabilistic interpretation.Informally, it first estimates the Akl and Ek (b) by considering probable paths forthe training sequences using the current values of akl and ek (b). Then (3.18) isused to derive new values of the as and es. This process is iterated until somestopping criterion is reached.It is possible to show that the overall log likelihood of the model is increasedby the iteration, and hence that the process will converge to a local maximum.Unfortunately, there are usually many local maxima, and which one you end upwith depends strongly on the starting values of the parameters. The problem oflocal maxima is particularly severe when estimating large HMMs, and later wewill discuss various ways to help deal with it.More formally, the Baum–Welch algorithm calculates Akl and Ek (b)asthe expectednumber of times each transition or emission is used, given the trainingsequences. To do this it uses the same forward and backward values as the posteriorprobability decoding method. The probability that akl is used at position i insequence x is (see Exercise 3.5)fk (i)aklel (xi+1)bl (i +1)

P(πi =k, πi+1 =l|x, θ) =. (3.19)P(x)From this we can derive the expected number of times that akl is used by summingover all positions and over all training sequences,fj j

Akl =1k (i)aklel (xi+1)blj (i +1), (3.20)P(xj )

jiwhere fkj (i) is the forward variable fk (i) defined in (3.10) calculated for sequence?

3.3 Parameter estimation for HMMsj, and blj (i) is the corresponding backward variable. Similarly, we can find theexpected number of times that letter b appears in state k,Ek (b) =1fkj (i)bkj (i), (3.21)

P(xj )

j {i|xj =b}

iwhere the inner sum is only over those positions i for which the symbol emittedis b.Having calculated these expectations, the new model parameters are calculatedjust as before using (3.18). We can iterate using the new values of the parametersto obtain new values of the As and Es as before, but in this case we are convergingin a continuous-valued space, and so will never in fact reach the maximum. It istherefore necessary to set a convergence criterion, typically stopping when thechange in total log likelihood is sufficiently small. Other stop criteria than the loglikelihood change can be used for the iteration. For instance the log likelihoodcan be normalised by the number of sequences n and maybe also by the sequencelengths, so that you consider the change in the average log likelihood per residue.We can summarise the Baum–Welch algorithm like this:Algorithm: Baum–WelchInitialisation: Pick arbitrary model parameters.Recurrence:Set all the A and E variables to their pseudocount values r (or to zero).For each sequence j =1 ...n:Calculate fk (i) for sequence j using the forward algorithm (p. 59).Calculate bk (i) for sequence j using the backward algorithm (p. 60).Add the contribution of sequence j to A (3.20) and E (3.21).Calculate the new model parameters using (3.18).Calculate the new log likelihood of the model.Termination:Stop if the change in log likelihood is less than some predefined thresholdor the maximum number of iterations is exceeded. .As indicated here, it is normal to add pseudocounts to the A and E valuesjust as in the case where the state paths are known. This works well, but thenormal Bayesian interpretation in terms of Dirichlet priors does not carry throughrigorously in this case; see Chapter 11.The Baum–Welch algorithm is a special case of a very powerful general approachto probabilistic parameter estimation called the EM algorithm. This algorithmand the derivation of Baum–Welch is given in Section 11.6 ofChapter 11.?

3 Markov chains and hidden Markov modelsAn alternative to the Baum–Welch algorithm is frequently used, which we willcall Viterbi training. In this approach, the most probable paths for the training sequencesare derived using the Viterbi algorithm given above, and these are usedin the re-estimation process given in the previous section. Again, the process isiterated when the new parameter values are obtained. In this case the algorithmconverges precisely, because the assignment of paths is a discrete process, andwe can continue until none of the paths change. At this point the parameter estimateswill not change either, because they are determined completely by thepaths. Unlike Baum–Welch, this procedure does not maximise the true likeli

1

hood, i.e. P(x, ..., xn|θ) regarded as a function of of the model parameters θ.Instead, it finds the value of θthat maximises the contribution to the likelihood1

P(x, ..., xn , π.(x1), ..., π.(xn)|θ) from the most probable paths for all the sequences.Probably for this reason, Viterbi training performs less well in generalthan Baum–Welch. However, it is widely used, and it can be argued that when theprimary use of the HMM is to produce decodings via Viterbi alignments, then itis good to train using them.Example: The occasionally dishonest casino, part 5We are suspicious that a casino is operated as described in the example on p. 55,but we do not know for certain. Night after night we collect data by simply observingrolls. When we have enough, we want to estimate a model. Assume thedata we collected were the 300 rolls shown in Figure 3.5. From this sequence ofobservations a model was estimated by the Baum–Welch algorithm. Initially allthe probabilities were set to random numbers. Here are diagrams of the modelthat generated the data (identical to the one in the example on p. 55) and the estimatedmodel.1: 1/62: 1/63: 1/64: 1/65: 1/61: 1/102: 1/103: 1/104: 1/105: 1/100.95 0.90.050.11: 0.192: 0.193: 0.234: 0.085: 0.231: 0.072: 0.103: 0.104: 0.175: 0.050.73 0.710.270.296: 1/6 6: 1/2 6: 0.08 6: 0.52FairLoadedFairLoadedYou can see they are fairly similar, although the estimated transition probabilitiesare quite different from the real ones. This is partly a problem of local minima,and by trying more times it is actually possible to obtain a model closer to the correctone. However, from a limited amount of data it is never possible to estimatethe parameters exactly.?

3.3 Parameter estimation for HMMsTo illustrate the last point, 30 000 random rolls were generated (data are notshown!), and a model was estimated. This came very close to the correct one:0.930.881: 0.17 1: 0.102: 0.17 0.07 2: 0.113: 0.173: 0.104: 0.174: 0.115: 0.17 0.12 5: 0.106: 0.15 6: 0.48FairLoadedTo see how good these models are compared to just assuming a fair die all thetime, the log-odds per roll was calculated using the 300 observations for the threemodels:The correct model0.101 bitsModel estimated from 300 rolls 0.097 bitsModel estimated from 30 000 rolls 0.100 bitsThe worst model estimated from 300 rolls has almost the same log-odds as thetwo other models. That is because it is being tested on the same data as it wasestimated from. Testing it on an independent set of rolls yields significantly lowerlog-odds than the other two models.Exercises3.5Derive the result (3.19). Use the fact that1P(πi =k, πi+1 =l|x, θ) =P(x, πi =k, πi+1 =l|θ),

P(x|θ)and that this again can be written in terms of P(x1, ..., xi , πi =k|θ) andP(xi+1, ..., xL , πi+1 =l|x1, ..., xi , θ, πi =k)=P(xi+1, ..., xL , πi+1 =l|θ, πi =k).3.6 Derive (3.21).Modelling of labelled sequencesIn the above example with CpG islands we have seen how HMMs can be used topredict the labelling of unannotated sequences. In these examples we had to trainthe models of CpG islands separately from the model of non-CpG islands andthen combine them into a larger HMM afterwards. This separate estimation canbe quite tedious, especially if there are more than two different classes involved.Also, if the transitions between the submodels are ambiguous, so for instance agiven sequence can use more than one transition from the CpG submodel to theother submodel, then the estimation of the transitions is not a simple counting?

3 Markov chains and hidden Markov modelsSequence x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 ...Labels . . . + + + + . . . ...State12345678....++++f = 0fcalculatedas usualf = 0fcalculatedas usualf = 0fcalculatedas usualFigure 3.8 The forward table for a model with four states labelled +andfour labelled .. Each column corresponds to an observation and each rowto a state of the model. The first ten residues shown, x1,..., x10, are assumedto be labelled ...++++....problem. There is, however, a more straightforward method to estimate everythingat once, which we will describe now.The starting point is the combined model of all the classes, where we haveassigned a class label to each state. To model CpG islands the natural labels are‘+’ for the island states and ‘.’ for the non-island states. We also have labels onthe observations x =x1, ..., xL , which we we call y =y1, ..., yL .The yi is ‘+’ifxi is part of a CpG island and ‘.’ otherwise. In the Baum–Welch algorithm (orthe Viterbi alternative) we now only allow valid paths through the model whencalculating the f s and bs. A valid path is one where the state labels and sequencelabels are the same, i.e., πi has label yi . During the forward and backward algorithmsthis corresponds to setting fl (i) =0 and bl (i) =0 for all the states l witha label different from yi (see Figure 3.8).Discriminative estimationUnless there are ambiguous transitions between submodels, the above estimationprocedure gives the same result as if the submodels were estimated separatelyby the Baum–Welch algorithm and then combined with appropriate transitionsafterwards. This actually corresponds to maximising the likelihoodML

θ=argmax P(x, y|θ).θUsually our primary interest is in obtaining good predictions of y, so it is preferableto maximise P(y|x, θ) instead. This is called conditional maximum likelihood(CML),CMLθ=argmax P(y|x, θ); (3.22)θsee for example Juang & Rabiner [1991] and Krogh [1994]. A related criterion iscalled maximum mutual information or MMI [Bahl et al. 1986].?

3.4 HMM model structureThe likelihood P(y|x, θ) can be rewritten asP(x, y|θ)

P(y|x, θ) =,

P(x|θ)where P(x, y|θ) is the probability calculated by the forward algorithm for la-

belled sequences described above, and P(x|θ) is the probability calculated by thestandard forward algorithm disregarding all the labels. There is no EM algorithmfor optimising this likelihood, and the estimation becomes more complex; see forexample Normandin & Morgera [1991] and the references above.3.4 HMM model structureChoice of model topologySo far we have assumed that transitions are possible from any state to any otherstate. Although it is tempting to start with a fully connected model, i.e. one inwhich all transitions are allowed, and ‘let the model find out for itself’ whichtransitions to use, it almost never works in practice. For problems of any realisticsize it will usually lead to very bad models, even with plenty of trainingdata. Here the problem is not over fitting, but local maxima. The less constrainedthe model is, the more severe the local maximum problem becomes.There are methods that attempt to adapt the model topology based on the databy adding and removing transitions and states [Stolcke & Omohundro 1993;Fujiwara, Asogawa & Konagaya 1994]. However, in practice successful HMMsare constructed by carefully deciding which transitions are to be allowed in themodel, based on knowledge about the problem under investigation.To disable the transition from state k to state l corresponds to setting akl =0.If we use Baum–Welch estimation (or the Viterbi approximation) then akl willstill be zero after the re-estimation process, because when the probability is zerothe expected number of transitions from k to l will also be zero. Therefore all themathematics is unchanged even if not all transitions are possible.We should choose a model which has an interpretation in terms of our knowledgeof the problem. For instance, to model CpG islands it was important that themodel was capable of giving a different probability to a CG dinucleotide in theisland states from in the non-island states, because that was expected to be themain determinator for CpG islands.Duration modellingWhen modelling a phenomenon where for instance the nucleotide distributiondoes not change for a certain length of DNA, the simplest model design is tomake a state with a transition to itself with probability p. We did this with bothour CpG island and our dishonest casino example. After entering the state there?

3 Markov chains and hidden Markov modelsis a probability 1 .p of leaving it, so the probability of staying in the state for lresidues isP(l residues) =(1 .p) pl.1. (3.23)(The emission probabilities are disregarded.) This exponentially decaying distributionon lengths (called a geometric distribution) can be inappropriate in someapplications, where the distribution of lengths is important and significantly differentfrom exponential. More complex length distributions can be modelled byintroducing several states with the same distribution over residues and transitionsbetween each other. For instance a (sub-) model like this:will give sequences of a minimum length of 5 residues and an exponentially decayingdistribution over longer sequences. Similarly, a model like this:can model any distribution of lengths between 2 and 10.A more subtle way of obtaining a non-geometric length distribution is to use anarray of n states, each with a transition to itself of probability p and a transitionto the next of probability 1 .p:pppp1.p1.p1.p1.pObviously the smallest sequence length such a model can capture is n. For anygiven path of length l through the model, the probability of all its transitionsis pl.n(1 .p)n (we are disregarding emission probabilities for now, as above).l.1

The number of possible paths through the states is n.1 , so the total probabilitysummed over all possible paths isl .1P(l) =pl.n(1 .p)n . (3.24)n .1This distribution is called a negative binomial and it is shown in Figure 3.9 forp =0.99 and n ≤5. For small lengths the number of paths through the modelgrows faster than the geometrical distribution decays, and therefore the distributionbecomes bell-shaped. The number of paths depends on the model topology,and it is possible to make more general models where the number of paths has adifferent dependence on n and l. For continuous Markov processes the types of?

3.4 HMM model structure00.0010.0020.0030.004P(l)n=1n=2n=3n=4n=50 200 400 600 800 1000lFigure 3.9 The probability distribution over lengths for models with p =0.99 and n identical states, with n ranging from 1 to 5.distributions that can be obtained are called Erlang distributions or more generallyphase-type distributions, see for example Asmussen [1987].Alternatively, it is possible to model the length distribution explicitly. As lengthis equivalent to time in many signal processing applications, this is called durationmodelling. The price one has to pay is that algorithms are much slower. SeeRabiner [1989] for more details.Silent statesWe have already seen examples of states that do not emit symbols in an HMM,the begin and end states. Such states are called silent states or null states, andthey can also be useful in other places in an HMM. In Chapter 5 we will see anexample where all states in a chain of states need to be connected to all states laterin the chain. The length of such a chain is often 200 states or more, and connectingthem appropriately with transitions would require roughly 20 000 transitionprobabilities (assuming 200 states). This number is too large to be reliably estimatedfrom realistic datasets. Instead, by using silent states, we can get awaywith around 800 transitions.The situation is as follows: to allow for arbitrary deletions a chain of statesneeds to be completely ‘forward connected’.Instead we can connect all the states to a parallel chain of silent states, representedhere by circles.?

3 Markov chains and hidden Markov modelsBecause the silent states do not emit any letters, it is possible to get from any‘real’ state to any later ‘real’ state without emitting any letters.A price is paid for the reduction in the number of parameters. The fully connectedmodel can have for instance high probability transitions from state 1 tostate 5 and from state 2 to state 4, but low probability ones for transitions 1 to 4and 2 to 5. This would not be possible with the model using silent states.So long as there are no loops consisting entirely of silent states, it is easy to extendall the HMM algorithms to incorporate them. The condition that there are noloops mean that the states can be numbered so that any transition between silentstates goes from a lower to a higher numbered state. For the forward algorithm,the change is as follows:(i) For all ‘real’ states l, calculate fl (i +1) as before from fk (i) for states k.(ii) For any silent state l,set fl (i +1) to k fk (i +1)akl for ‘real’ states k.(iii) Starting from the lowest numbered silent state l add k fk (i +1)akl tofl (i +1) for all silent states k <l.The change to the Viterbi algorithm is exactly the same (sums replaced by maximisationof course), and for the backward algorithm the change is essentially thesame except in the third step the silent states are updated in reverse order.If there are loops consisting entirely of silent states, the situation gets a littlemore complicated. It is possible to eliminate the silent states from the calculationby calculating (exactly) the effective transition probabilities between real statesin the model, which involves inverting the transition matrix for the Markov modelof silent states [Cox & Miller 1965]. Often, however, these effective transitionscorrespond to a fully connected model, and this leads to a substantial increase inthe complexity of the model. Usually it is best to simply make sure such loops donot exist.Exercises3.7Calculate the total number of transitions needed in a forward connectedmodel as the one shown above with a length of L. Calculate the samenumber for a model with silent states (as above).3.8Show that the number of paths through an array of n states is indeedl.1n.1 for length l as in (3.24).3.9Consider the model with n states with self-loops giving rise to equation(3.24). What is the probability for the most likely path through the model?

3.5 More complex Markov chainsfor a sequence of length l (when ignoring emission probabilities)? Is thistype of length modelling useful with the Viterbi algorithm?3.5 More complex Markov chainsHigh order Markov chainsAn nth order Markov process is a stochastic process where each event dependson the previous n events, soP(xi |xi.1, xi.2, ..., x1) =P(xi |xi.1, ..., xi.n). (3.25)The Markov chains we have discussed so far are of order 1.An nth order Markov chain over some alphabet Ais equivalent to a first orderMarkov chain over the alphabet An of n-tuples. This follows from the simple factthat P(xk |xk.1 ...xk.n) =P(xk , xk.1 ...xk.n+1|xk.1 ...xk.n) (the probability ofA and B given B is the probability of A given B). That is, the probability of xkgiven the n-tuple ending in xk.1 is equal to the probability of the n-tuple endingin xk given the n-tuple ending in xk.1.Consider the simple example of a second order Markov chain for sequences ofonly two different characters A and B. A sequence is translated to a sequence ofpairs, so for instance the sequence ABBAB becomes AB-BB-BA-AB. The equivalentfour-state first order Markov chain will look like this:AA ABBA BBIn this equivalent model not all transitions are allowed (or alternatively, some ofthe transition probabilities are zero). This is because only two different pairs canfollow a given letter; the state AB for instance can only be followed by the statesBA and BB. No sequence exists that can go from state AB to state AA. Similarly,a second order model for DNA is equivalent to a first order model over an alphabetof the 16 dinucleotides. A sequence of five bases, CGTCA, corresponds to achain of four states, CG-GT-TC-CA, in a dinucleotide model.Despite the theoretical equivalence between an nth order model and a first ordermodel, the framework of high order models (meaning models of order greaterthan 1) is sometimes more convenient. Theoretically the high order models aretreated in a way completely equivalent to first order models.?

3 Markov chains and hidden Markov modelsGTCAGATGAGCAAAGTCAGACTCGCAATTATGAACGTATCCCAGTAACGCCstartcodonstopcodonDNA sequencegenescodonscodonscodonsAGCGCAFigure 3.10 The organisation of genes in prokaryotes.Finding prokaryotic genesAn example is given by a model for identifying prokaryotic genes. Genes ofprokaryotes (bacteria) have a very simple one-dimensional structure. A gene codingfor a protein starts with a start codon, then has a number of codons codingfor amino acids, and ends with a stop codon; see Figure 3.10. Codons are DNAnucleotide triplets of which 61 code for amino acids and three are stop codons.In order to focus on the modelling, many complications such as frame shifts andnon-protein genes are ignored here.It is very easy to find good gene candidates by simply looking for stretches ofDNA with the correct structure, i.e. starting with one of the three possible startcodons, continuing with a number of non-stop codons and ending with one ofthe three stop codons. Such a gene candidate is called an open reading frame orjust an ORF. Usually there are many overlapping ORFs that have the same stopcodon, but different start codons. (The term ORF is often used for the maximalopen reading frame between two stop codons, but we shall use it for all possiblegene candidates.) There are many more ORFs than real genes, and here we willsketch possible ways of distinguishing between a non-coding ORF and a realgene.In this example DNA from the bacterium E. coli is used (the dataset is describedin detail in Krogh, Mian & Haussler [1994]). We consider only genesmore than 100 nucleotides long. In the dataset there are 1100 such genes. Thisset is arbitrarily divided into a training set of 900 for training our models, and atest set containing the remaining 200 genes.?

3.5 More complex Markov chains80400-0.05 0.00 0.05 0.10Bits per nucleotideFigure 3.11 Histograms of the log-odds per nucleotide for all NORFs(grey) and genes (black line) according to a first order Markov chain. Becauseof the large number of NORFs, the histogram bin size is five timessmaller for the NORFs.We estimate a first order model just as we did for the CpG islands early inthis chapter and test how well it discriminates genes from other ORFs. In the testset we found roughly 6500 ORFs with a length of more than 100 bases. ORFsthat share the stop codon with a known real gene were not included, because theywould generally score very well and make our subsequent analysis more difficult.The remaining ORFs that are not labelled as coding will be called NORFs (fornon-coding ORFs).In Figure 3.11 a histogram is shown of the log-odds per nucleotide. As the nullmodel for calculating log-odds we used the simplest possible, with the probabilityfor each nucleotide equal to the frequency by which it occurs in all the data.The average log-odds per nucleotide for all the genes is 0.018, whereas it is halfas much (0.009) for the NORFs, but the variance makes it almost useless fordiscrimination. You could fool yourself into thinking that the model had a decentdiscriminative power if you plotted the histogram of log-odds without dividing bythe sequence length, because the genes are longer on average than NORFS, andtherefore also the total log-odds is larger for the NORFs. Almost all the apparentinformation about genes would come from the length distribution and not fromthe model.It is worth noticing that the average of the histogram is not at 0 bits, and thatthe averages of the two distributions (genes and NORFs) are quite close. Thisindicates that the Markov chain has indeed found a non-random correlation betweennucleotide pairs, but it is essentially the same in coding and non-codingregions. In a second order chain, the probability of a nucleotide depends on thetwo previous ones, so it spans the length of a codon. Therefore we also tried asecond order model, but the result is almost identical to the one for the first ordermodel, so we do not show the histogram. It would probably not help much?

3 Markov chains and hidden Markov modelsto switch to a Markov chain of even higher order, because these models do notseparate the three reading frames, i.e. the three different nucleotide positions inthe codon.It is possible to make a high order inhomogeneous Markov chain (discussedin the next section) for modelling the bases in three different reading frames,but since our goal is to score ORFs, we will do it differently. The sequences aretransformed to sequences of codons. An arbitrary symbol is assigned to each ofthe 64 codons, and all genes and NORFs are translated to this alphabet (yieldingsequences of one-third the length of the nucleotide sequences). Notice that thistransformation is slightly different from the one above for transforming an nthorder model into a first order one, because the triplets are non-overlapping.A 64-state first order Markov chain was estimated from the translated sequencesand tested on the genes in the test set and the NORFs in exactly thesame way as the models above. The result is shown in Figure 3.12. Although theseparation is not perfect, we see that it is much better than for the other model.Notice that the distribution we compare to in the log-odds score now is a uniformdistribution over codons. The grey peak is centred around 0, indicating that theMarkov chain has found a signal that is special to coding regions, and that codonusage is essentially random in the average NORF, and that a significant fractionof the NORFs scoring highly represent real genes that are not labelled as such inour data. It is likely that most of the ORFs scoring above 0.3–0.35 bits in this plotare overlapping with real genes. The NORF histogram uses a smaller bin size (asin Figure 3.11), and if the same bin size was used, the NORF histogram wouldbe about five times higher.If the log-odds is not normalised by sequence length the discrimination improvessignificantly, because real genes tend to be longer than NORFs, seeFigure 3.12.Exercises3.10 Calculate the number of parameters in the above codon model. The datasetcontains on the order of 300 000 codons. Would it be feasible to estimatea second order Markov chain from this dataset?3.11 How can the above gene model be improved?Inhomogeneous Markov chainsAs we saw above, a successful Markov model of genes needs to model the codonstatistics. This can also be done without translating to another alphabet. It is wellknown that in genes the three codon positions have quite different statistics, andtherefore it is natural to use three different Markov chains to model coding regions.The three models are numbered 1 to 3 according to the position in the?

3.5 More complex Markov chains80400-1.0 -0.5 0.0 0.5 1.0Bits per nucleotideBits5004003002001000-100100 200 300 400 500 600 700 800Sequence lengthFigure 3.12 The top plot shows the histograms of NORFs and genes for theMarkov chain of codons (cf. Figure 3.11). Below, the log-odds is shown asa function of length for genes (+) and NORFs (·).codon. Assuming that x1 is in codon position 3, the probability of x2, x3, ...wouldthen be12312

aaaaa···x1 x2 x2x3 x3 x4 x4 x5 x5 x6where the parameters for model k are called ak . This is called an inhomogeneousMarkov chain. Here we assumed the chain was first order, but it is of coursepossible to extend it to order n. The estimation of the parameters is a straightforwardextension of the estimation of the homogeneous models described inSection 3.1: for a second order inhomogeneous Markov chain as above the parametersof model 1 are estimated by counting the triplets with the last base incodon position 1, and similarly for model 2 and 3.Inhomogeneous Markov chains are used extensively in the GENEMARK gene-

finding program [Borodovsky & McIninch 1993], which is currently the mostwidely used method for prokaryotic genefinding. Inhomogeneous models of orderup to five of coding regions have been combined with homogeneous modelsof the non-coding regions to localise genes in a number of different bacterialgenomes.?

3 Markov chains and hidden Markov modelsThe first order model described above can also be constructed as an HMM,with the number of states equal to three times the length of the alphabet (a totalof 12 for DNA). Higher order models can be made by adding many additionalstates to the HMM. However, it is also possible to have nth order Markov emissionprobabilities in the states of an HMM, in which the emission probabilitiesare conditioned on the n previous characters, so the emission probabilities (3.5)becomeek (b|b1, ..., bn) =P(xi |πi =k, xi.1 =b1, ..., xi.n =bn).All the algorithms derived for standard HMMs can be used with only obviousalterations for models with these emissions. Such models are also being used forgenefinding [Krogh 1998].Exercise3.12Draw the HMM that corresponds to the first order inhomogeneous Markovchain given above.3.6 Numerical stability of HMM algorithmsEven on modern floating point processors we will run into numerical problemswhen multiplying many probabilities in the Viterbi, forward, or backward algorithms.For DNA for instance, we might want to model genomic sequences of100 000 bases or more. Assuming that the product of one emission and one transitionprobability is typically 0.1, the probability of the Viterbi path would then beof the order of 10.100000. Most computers would behave badly with such numbers:either an underflow error would occur and the program would crash; or,worse, the program would keep running and produce arbitrary wrong numbers.There are two different ways of dealing with this problem.The log transformationFor the Viterbi algorithm we should always use the logarithm of all probabilities.Since the log of a product is the sum of the logs, all the products are turnedinto sums. Assuming the logarithm base 10, the log of the above probability of10.100000 is just .100000. Thus, the underflow problem is essentially solved.Additionally, the sum operation is faster on some computers than the product, soon these computers the algorithm will also run faster.We will put a tilde on all the model parameters after taking the log, so forexample a.kl =log akl . Then the recursion relation for the Viterbi algorithm (3.8)becomesVl (i +1) =e.l (xi+1) +max(Vk (i) +a.kl ),

k?

3.6 Numerical stability of HMM algorithmswhere we use V for the logarithm of v. The base of the logarithm is not importantas long as it is larger than 1 (such as 2, e, and 10).It is more efficient to take the log of all the model parameters before runningthe Viterbi algorithm, to avoid calling the logarithm function repeatedly duringthe dynamic programming iteration.For the forward and backward algorithms there is a problem with the log transformation:the logarithm of a sum of probabilities cannot be calculated from thelogs of the probabilities without using exponentiation and log functions, whichare computationally expensive. However, the situation is not in practice so bad.Assume you want to calculate .r =log( p +q) from the log of the probabilities,p.=log p and .q =logq. The direct way is to do .r =log(exp( .p) +exp( .q)). Bypulling out .p, one can write this asr.=p.+log(1 +exp( .q .p.)).It is possible to approximate the function log(1 +exp(x)) by interpolation froma table. For a reasonable level of accuracy, the table can actually be quite small,assuming we always pull out the largest of .p and .q, because exp( .q .p.) rapidlyapproaches zero for large ( .p .q.).Scaling of probabilitiesAn alternative to using the log transformation is to rescale the f and b variables,so they stay within a manageable numerical interval [Rabiner 1989]. For each idefine a scaling variable si , and define new f variablesfl (i)

f.l (i) =. (3.26)ij=1 sjFrom this it is easy to see thatf.l (i +1) =1el (xi+1) f.k (i)akl ,si+1kso the forward recursion (3.11) is only changed slightly. This will work howeverwe define si , but a convenient choice is one that makes f.l (i) =1, which meanslthatsi+1 =el (xi+1) f.k(i)akl .lkThe b variables have to be scaled with the same numbers, so the recursion stepin (3.3) becomesb.

k (i) =1aklb.

l (i +1)el (xi+1)silThis scaling method normally works well, but in models with many silent?

3 Markov chains and hidden Markov modelsstates, such as the one we describe in Chapter 5, underflow errors can stilloccur.ExercisesL

3.13Use (3.26) to prove that P(x) ==1 sj with the above choice of si .Itisj

of course wiser to calculate log P(x) =j log sj .3.14Use the result of the previous exercise to show that the equation (3.20)actually simplifies when using the scaled f and b variables. Also, derivethe result (3.21) for the scaled variables.3.7 Further readingMore basic introductions to HMMs include Rabiner & Juang [1986] and Krogh[1998].Some early applications of HMM-like models to sequence analysis was doneby Borodovsky et al. [1986a; 1986b; 1986c] who used inhomogeneous Markovchains as described on p. 76. This later led to the GENEMARK genefinder program[Borodovsky & McIninch 1993]. Cardon & Stormo [1992] introduced an expectationmaximisation (EM) method, which has many similarities with an HMM,for modelling protein binding motifs. Later applications of HMMs to genefindinginclude Krogh, Mian & Haussler [1994], Henderson, Salzberg & Fasman[1997], and Krogh [1997a,1997b,1998] as well as systems combining neural networksand HMMs [Stormo & Haussler 1994; Kulp et al. 1996; Reese et al. 1997;Burge & Karlin 1997]. Such hybrid systems are also becoming quite popularfor other applications; see for instance Bengio et al. [1992], Frasconi & Bengio[1994], Renals et al. [1994], Baldi & Chauvin [1995], and Riis & Krogh [1997].Churchill [1989] used HMMs for modelling compositional differences betweenDNA from mitochondria and from the human X chromosome and bacteriophagelambda, and later for studying the compositional structure of genomes[Churchill 1992]. Other applications include a three-state HMM for predictionof protein secondary structure [Asai, Hayamizu & Handa 1993], a HMM withten states in a ring for modelling an oscillatory pattern in nucleosomes [Baldi etal. 1996], detection of short protein coding regions and analysis of translationinitiation sites in cyanobacteria [Yada & Hirosawa 1996; Yada, Sazuka & Hirosawa1997], characterization of prokaryotic and eukaryotic promoters [Pedersenet al. 1996], and recognition of branch points [Tolstrup, Rouzé & Brunak 1997].Several other applications of HMMs will be discussed in the context of profileHMMs in Chapters 5 and 6.

4 Pairwise alignment using HMMs

Now that we have acquired new technical machinery from hidden Markov modeltheory, we return for a brief chapter to pairwise sequence alignment. In Chapter 2we introduced finite state automata with multiple states as a convenient descriptionof more complex dynamic programming algorithms for pairwise alignment.It is also possible to consider them as a basis for a probabilistic interpretationof the gapped alignment process, by converting them into HMMs. One advantageof this approach is that we will be able to use the resulting probabilisticmodel to explore questions about the reliability of the alignment obtained by dynamicprogramming, and to explore alternative (suboptimal) alignments. Indeed,by weighting all alternatives probabilistically, we will be able to score the similarityof two sequences independent of any specific alignment. We can also buildmore specialised probabilistic models out of simple pieces, to model more complexversions of sequence alignment, as discussed previously for FSAs.Let us first review briefly the finite state automaton that we introduced forpairwise alignment with affine gap penalties. We required three states, M correspondingto a match, and two states corresponding to inserts, which we namehere X and Y as shown in Figure 4.1. The recurrence relations for updating theM(+1,+1)Y(+0,+1)X(+1,+0) -e-e-d-ds(xi,yj)s(xi,yj)s(xi,yj)MYX εεδδqxiqyj1.ε1.ε1.2δ pxiyjFigure 4.1 A finite state machine diagram for affine gap alignment on theleft, and the corresponding probabilistic model on the right.81

4 Pairwise alignment using HMMsvalues of these states in the dynamic programming matrix are..V M(i .1, j .1),V M(i, j) =s(xi , yj ) +max V X(i .1, j .1), (4.1).V Y(i .1, j .1);V M(i .1, j) .d,

V X(i, j) =maxV X(i .1, j) .e;V M(i, j .1) .d,

V Y(i, j) =maxV Y(i, j .1) .e.These equations are appropriate for global alignment. As previously, we will generallygive detailed equations for global alignment, while indicating what changesneed to be made for local alignment.4.1 Pair HMMsWe need to make two sets of changes to an FSA as shown on the left side of Figure4.1 to turn it into an HMM. First, as shown on the right of Figure 4.1, we mustgive probabilities both for emissions of symbols from the states, and for transitionsbetween states. For example, state M has emission probability distributionpab for emitting an aligned pair a:b, and states X and Y will have distributionsqa for emitting symbol a against a gap. Because state X emits symbols xi fromsequence x, we write qxi inside the circle representing state X. We also specifytransition probabilities between the states, which must satisfy the requirementthat the probabilities for all the transitions leaving each state sum to one. Allowingfor symmetry, there are two free parameters for the transition probabilitiesbetween the three main states. We denote the transition from M to an insert state(X or Y) by δ, and the probability of staying in an insert state by ε.However, the resulting model shown on the right side of Figure 4.1 does notgenerate a full model that will provide a probability distribution over all possiblesequences. To do that, we need to define a Begin and an End state, as shownin Figure 4.2. In effect these formalise the initialisation and termination conditionsthat we needed for the dynamic programming algorithms in Chapter 2. Wewill see below that more complex arrangements of Begin and End states cancorrespond to local and other types of alignments. Adding an explicit End stateintroduces the need for another parameter, the probability of a transition into theEnd state, which we assume for now to be the same from each of M, X and Y; wecall it τ. This will in effect determine the average length of an alignment from themodel. For now, we will set the transitions from the Begin state to be the same asfrom the M state (we could have just said that we will start in M, but we wantedto make clear that initialisation can be given independent consideration as wellas termination).

4.1 Pair HMMsMYXqxi1.ε.τδεεδ1.2δ.τqyjpxiyjτ1.ε.τ τττ1.2δ.τδδBegin EndFigure 4.2 The full probabilistic version of Figure 4.1.This gives us a probabilistic model that is very similar to a hidden Markovmodel as we defined it in Chapter 3. The difference is that instead of emittinga single sequence it emits a pairwise alignment. We will call this type of modela pair HMM to distinguish it from the more standard types of HMMs that emitsingle sequences. All the algorithms from Chapter 3 carry across to pair HMMs,although they need an extra dimension of search space because of the extra emittedsequence. For example, instead of writing vk (i) for the Viterbi probabilities,we write vk (i, j). We will give below the explicit sets of equations for the keyalgorithms, applied to the basic pair HMM shown in Figure 4.2.Just as a standard HMM can generate a sequence, our pair HMM can generatean aligned pair of sequences. This is done by starting in the Begin state, andcycling over the following two steps: (1) pick the next state according to thedistribution of transition probabilities leaving the current state; (2) pick a symbolpair to be added to the alignment according to the emission distribution in the newstate. The process stops when a transition is made into the End state. Because wehave probabilities for each step, we can also keep track of the total probability ofgenerating a particular alignment that we have made. This is just the product ofthe probabilities of each individual step.The most probable path is the optimal FSA alignmentThe Viterbi algorithm from Chapter 3 will allow us to find the most probable paththrough a pair HMM given sequences x and y. The correct form for the globalpair HMM of Figure 4.2 is as follows. To make the equations simpler, we definethe Begin state to be M. As in the previous chapter, we use lower-case symbolsv.(i, j) for probability values, and upper-case V .(i, j) for log-odds scores. Wegive the Viterbi algorithm first in terms of probabilities:

4 Pairwise alignment using HMMsAlgorithm: Viterbi algorithm for pair HMMsInitialisation:vM(0,0) =1. vX(0,0) =vY(0,0) =0.All v.(i, .1), v.(.1, j) are set to 0.vM(0,0) =1. All other v.(i,0), v.(0, j) are set to 0.Recurrence: i =0, ..., n, j =0, ..., m except (0,0);..

(1 .2δ.τ)vM(i .1, j .1),vM(i, j) =pxi yj max (1 .ε.τ)vX(i .1, j .1),.

(1 .ε.τ)vY(i .1, j .1);δvM(i .1, j),vX(i, j) =qxi maxεvX(i .1, j);δvM(i, j .1)vY(i, j) =qyj ,maxεvY(i, j .1).Termination:Ev=τmax(vM(n, m), vX(n, m), vY(n, m)). .To find the best alignment, we keep pointers and trace back as usual. Of course,to get the alignment itself we keep track of which residues are emitted at each stepin the path during the traceback, as in Chapter 2, as well as (or even in place of)the sequence of states as for the type of HMM described in Chapter 3.Although it is clear that the recurrence equations of the pair HMM Viterbialgorithm have the same sort of form as those for the state machine version ofpairwise alignment (4.1), it is instructive to see the exact form of the correspondence.First, we have to transform into log-odds ratios with respect to the randommodel. In fact, now we have a full probabilistic model for our alignment, weshould also have one for our random model, with a proper termination condition.Previously we have ignored the fact that our random model could not produce sequencesof varying length in a proper probabilistic fashion. Here is a new randommodel, which is also a pair HMM.YXqxi1.ηqyjBegin End1.η1.ηηηηη1.ηThe main states are X and Y, which emit the two sequences in turn, independentlyof each other. Each has a loop back onto itself with probability (1 .η). As well asBegin and End states, there is also a silent state in between X and Y, indicated bya smaller circle. This does not emit any symbols, but is used to gather inputs fromboth the X and Begin states (see the section on silent states on p. 71 for further

4.1 Pair HMMsinformation on how these are used). When defined this way the model allowszero-length sequences x or y, just as the pair HMM model in Figure 4.2 does,and generates a simple form for the random model distribution over sequences.The probability of a pair of sequences x and y according to this model isnmP(x, y|R) =η(1 .η)n qxi η(1 .η)m qyji=1 j=1nmm

=η2(1 .η)n+qxi qyj . (4.2)i=1 j=1We now want to allocate the terms in this expression to those that make up theprobability of the Viterbi alignment, so that the odds ratio for the whole alignmentcan be expressed as a product of odds ratios of individual terms (and, correspondingly,so that the log-odds ratio of the alignment is a sum of log-odds terms). Wedo this by allocating one factor of (1 .η) and the corresponding qa factor to eachresidue that is emitted in a Viterbi step. So the match transitions will be allocated(1 .η)2qaqb where a and b are the two residues matched, and the insert states(1 .η)qa where a is the residue inserted. Because the Viterbi path must accountfor all the residues, exactly (n +m) terms will be used, and all of (4.2) except theinitial factor of η2 is accounted for.In log-odds terms, we can now compute in terms of an additive model with log-

odds emission scores and log-odds transition scores. In practice this is normallythe most practical way to implement pair HMMs. From this, it is possible tomerge the emission scores with the transitions as shown here:pab (1 .2δ.τ)s(a, b) =log +log ,qaqb (1 .η)2δ(1 .ε.τ)

d =.log ,

(1 .η)(1 .2δ.τ)εe =.log ,

1 .ηto produce scores that correspond to the standard terms used in sequence alignmentby dynamic programming. Note that the qa contribution to d and e has vanishedbecause the factors from the Viterbi and random models cancelled. Alsoin order to absorb differences in the transitions coming from the match and gapstates, there has been a little sleight of hand in the expressions for s and d.Weintend to use s(a, b) as a score for every match, whether following another matchor an insertion. In order to make this work correctly, we have built into d an adjustmentto correct for the difference in match score when returning back from aninsertion. This means that the dynamic programming matrix terms for the insertionsno longer correspond exactly to the log-odds ratios of being in those states,although the final result will be correct.

4 Pairwise alignment using HMMsWe can now give the log-odds version of the Viterbi alignment algorithm in aform that looks like standard pairwise dynamic programming.Algorithm: Optimal log-odds alignmentInitialisation:V M(0,0) =.2log η, V X(0,0) =V Y(0,0) =.∞.All V .(i, .1), V .(.1, j) are set to .∞.Recursion: i =0, ..., n, j =0, ..., m except (0,0);.V M(i .1, j .1),

.V M(i, j) =s(xi , yj ) +max V X(i .1, j .1),.V Y(i .1, j .1);.V M(i .1, j) .d,

V X(i, j) =maxV X(i .1, j) .e;.V M(i, j .1) .d,

V Y(i, j) =maxV Y(i, j .1) .e.Termination:V =max(V M(n, m), V X(n, m) +c, V Y(n, m) +c). .These are identical to (4.1) except for the constant 2log ηin the initialisation,and the constant c =log(1 .2δ.τ) .log(1 .ε.τ) in the termination, which isneeded to correct back for our adjustment described above in d. In fact the lattercorrection is only a result of having used the same exit probability τfor matchand insert states. If the exit transition probabilities from the gap states are set to(1 .ε)τ/(1 .2δ) then c will be zero, and hence the log-odds algorithm will haveexactly the same form as our standard pairwise affine gap alignment algorithm,with a single additive constant coming from the initialisation conditions.The procedure as we have described it shows how for any pair HMM of thetype shown in Figure 4.2 we can derive an equivalent FSA for obtaining the mostprobable alignment. This allows us to see a rigorous probability-based interpretationfor the terms used in sequence alignment. To do the reverse, i.e. to go froma dynamic programming algorithm expressed as an FSA to a pair HMM, is morecomplicated. There will in general be a need for a new parameter λwhich willact as a global scaling factor for the scores, and for any given set of scores theremay be constraints on the choice of ηand τ.A pair HMM for local alignmentThe model shown in Figure 4.2 is appropriate to finding a global match betweensequences. As described in Chapter 2, many of the most sensitive pairwisesearches are local. When we introduced the local alignment algorithm, and othervariants such as the repeat and overlap algorithms, we explained them in terms

4.1 Pair HMMsMYXqxi1.ε.τδεεδ1.2δ.τqyjpxiyjτ1.ε.τ τττ1.2δ.τδδEndRX1qxi1.ηqyjBegin1.η1.ηηηηη1.ηqxi1.ηqyj1.η1.ηηηηη1.ηRY1 RY2RX2Figure 4.3 A pair HMM for local alignment. This is composed of the globalmodel (states M, X and Y) flanked by two copies of the random model(states RX1, RY1 and RX2, RY2).of changes in the update equations and boundary conditions. Both of these aremade explicit in the pair HMM formalism by adding states and transitions. Wecan therefore draw a separate pair HMM model for each variant. In Figure 4.3 weshow a model for local alignment. This looks more complicated than the globalmodel in Figure 4.2, but it is made up of simpler pieces in a straightforwardfashion.A complete probabilistic model must account for all of the sequences x andy: not only the local alignment between x and y, but also the unaligned flankingsequences. We therefore add extra model sections before and after the three-statematching segment from Figure 4.2. Each flanking segment is a copy of the completerandom background model, because the sequences in the flanking regionsare unaligned. Most terms in the likelihood contributions of these sections willcancel out with equivalent terms in the random model when calculating the log-

odds scores of a match in comparison to the random model, leaving only the localmatching score from the central part of the model, and some extra one-off terms.Similar composite models can be built for overlap and repeat models, and thevarious hybrids discussed in Chapter 2.Exercises4.1What is the probability that sequence x has length t under the full randommodel?4.2What is the expected length of sequences from the full random model?How should the parameter ηbe set?

4 Pairwise alignment using HMMs4.2 The full probability of x and y, summing over all pathsHaving a pair HMM allows us to do more than provide an alternative rationale forstandard pairwise alignment by dynamic programming. One issue that we raisedwhen discussing the significance of matches in Chapter 2 was that, when similarityis weak, it is hard to identify the correct alignment to score and test forsignificance. Now we can bypass this problem (and the approach taken throughoutthe whole of Chapter 2) by calculating the probability that a given pair ofsequences are related according to the HMM by any alignment. We do this bysumming over alignments,P(x, y) =P(x, y, π).alignments πHow do we calculate this sum? Again, there is a standard HMM algorithm, describedin Chapter 3 as the forward algorithm. The way this works out for pairHMMs is that we can again use the same dynamic programming idea that we usedfor finding the maximal scoring alignment, but add rather than take the maximumat each step. The probability version of the forward algorithms is given below,using fk (i, j) to represent the combined probability of all alignments up to (i, j)that end in state k. As before, we give this only for the global model of Figure 4.2;the extension to other types of pairwise alignment model such as the local modeldescribed above is straightforward.Algorithm: Forward calculation for pair HMMsInitialisation:f M(0,0) =1. f X(0,0) =f Y(0,0) =0.All f .(i, .1), f .(.1, j) are set to 0.Recursion: i =0, ..., n, j =0, ..., m except (0,0);f M(i, j) =pxi yj (1 .2δ.τ) f M(i .1, j .1)+(1 .ε.τ)( f X(i .1, j .1) +f Y(i .1, j .1)) ;f X(i, j) =qxi δf M(i .1, j) +εf X(i .1, j);f Y(i, j) =qyj δf M(i, j .1) +εf Y(i, j .1) .Termination:f E(n, m) =τf M(n, m) +f X(n, m) +f Y(n, m). .We can now consider the log-odds ratio of the resulting full probabilityP(x, y) =f E(n, m) to the null model probability given by (4.2). This is a measureof the likelihood that the two sequences are related to each other by someunspecified alignment, as opposed to being unrelated. In doing this we have notassumed any specific alignment. Of course, if there is an unambiguous best alignment,almost all the probability in the total sum will be contributed by the single

4.2 The full probability of x and y, summing over all pathsHBA\_HUMAN KVADALTNAVAHVD-----DMPNALSALSDLHKV + +A ++ +L+ L+++HLGB2\_LUPLU KVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHHBA\_HUMAN KVADALTNAVAHVDDM-----PNALSALSDLHKV + +A ++ +L+ L+++HLGB2\_LUPLU KVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHHBA\_HUMAN KVADALTNA-----VAHVDDMPNALSALSDLHKV ++A V V +L+L+++HLGB2\_LUPLU KVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHFigure 4.4 An example of uncertainty in positioning a gap: three signifi

cantly different gap placements in the globin alignment from Figure 2.1(b),with very similar scores.path corresponding to this best alignment. However, the full score will alwaysbe higher than that for the optimal alignment (using the same scoring scheme),and it can be significantly different when there are many comparable alternativealignments, or alignment variations.An important use of the full probability is to define a posterior distributionP(π|x, y) over alignments πgiven a pair of sequences x, y. This is given byP(x, y, π)

P(π|x, y) =. (4.3)P(x, y)If we set π=π

, the Viterbi path, in (4.3), then we obtain the posterior probabilityaccording to the model of the Viterbi path vE(n, m)/f E(n, m), which we caninterpret as the probability that the optimal scoring alignment is ‘correct’. Frequentlythis is vanishingly small! For example for the alignment of alpha globinto leghaemoglobin in Figure 2.1(b) it is 4.6 ×10.6. This observation, althoughperhaps alarming if one was hoping that the standard alignment algorithms wouldfind the ‘correct’ alignment, is not surprising. There are many small variants ofthe best alignment that have nearly the same score, or equivalently are nearlyequally likely. In particular, where there is a gap there is often a choice of wherethe gap should be placed; moving it left or right by a residue or so frequentlyleads to no change or a seemingly random fluctuation.Figure 4.4 shows an example of this behaviour with corresponding sections ofthe human alpha globin and lupin leghaemoglobin sequences. The first alignmentshown is close to the structurally verified alignment, and has score 3 (BLOSUM50,gap-open .12, gap-extend minus .2). The next has the same score, although thegap is offset by two positions. The third has score 6, although the gap is misplacedby five residues. The difference in scores of 3 coresponds to an increase in relativelikelihood of a factor of two according to the alignment model, since BLOSUM50scores are given in third-bits. It is clear that simple sequence alignment is not an

4 Pairwise alignment using HMMsaccurate way to determine the alignment in this case, which is admittedly highlydiverged.Exercise4.3The relative scores for gap position variants such as shown in Figure 4.4depend only on the substitution scores, not the gap scores. Why is this,and what are the consequences for alignment accuracy using dynamicprogramming algorithms?4.3 Suboptimal alignmentGiven that there are frequently alternative alignments with nearly the same probability(or more generally nearly the same score) as the best alignment, it is naturallyof interest to see what they are. Such alignments are known as suboptimalalignments. There are a number of different approaches to examining and characterisingsuboptimal alignments. First let us consider more carefully what wemight expect to find.One class of alignments with scores close to the optimal score will be thosementioned above that only differ in a few positions from the optimal alignment(e.g. those in Figure 4.4). Because minor variations at different places in the alignmentcan be combined independently, the number of these ‘local’ variants growsexponentially as the difference in score from the optimal score increases. It istherefore impractical to give all such variants. However, the flexibility in varyingthe alignment can vary substantially with position along the alignment. Thereare sampling methods that illustrate typical variants, and methods that show foreach cell in the dynamic programming matrix how ‘close’ it is to being in thealignment. Examples of both of these are given below.Another type of suboptimal alignment is one that differs substantially, or perhapscompletely, from the optimal alignment. Methods for finding this type ofsuboptimal alignment can be used where one suspects that more than one correctalignment may be present, for instance where there are repeats in one or both ofthe sequences. In general, this is more relevant when searching for local alignments,which only align together a part of each sequence.Probabilistic sampling of alignmentsWe first give a method for sampling alignments from the posterior distributiondefined in (4.3). Recall that this gave a probability to each possible alignment ofthe two sequences, according to its likelihood of being correct under the model.An ensemble of such samples will give a picture of the type of alignment informationthat is reliably retrievable from a given sequence pair. Any particular

4.3 Suboptimal alignmentproperty of direct interest can be estimated by averaging over the sample, as suggestedin the section on posterior decoding of HMMs (p. 61). This is a powerfulgeneral strategy for using similarity information when the alignment is uncertainin detail; for example it is used later in the book in Chapter 8.To generate a sample alignment, we trace back through the matrix of fk (i, j)values, but instead of taking the highest scoring choice at each step, we make aprobabilistic choice based on the relative strengths of the three components. Toillustrate how this is done, let us imagine we are part way through the traceback,in state M at position (i, j), which we call cell M(i, j). We know from the forwardalgorithm thatf M(i, j) =pxi yj (1 .2δ.τ) f M(i .1, j .1)+(1 .ε.τ)( f X(i .1, j .1) +f Y(i .1, j .1)) .We choose the next step to bepxi yj (1 .2δ.τ) f M(i .1, j .1)

M(i .1, j .1) with prob. ,

f M(i, j)pxi yj (1 .ε.τ) f X(i .1, j .1)

X(i .1, j .1) with prob. ,

f M(i, j)pxi yj (1 .ε.τ) f Y(i .1, j .1)

Y(i .1, j .1) with prob. .f M(i, j)The corresponding distribution if in cell X(i, j) would be to chooseqxi δf M(i .1, j)

M(i .1, j) with prob. ,

f X(i, j)qxi εf X(i .1, j)

X(i .1, j) with prob. ,

f X(i, j)and similarly for cell Y(i, j).A set of sample global alignments from our simple example data is given here:HEAGAWGHEE HEAGAWGHE-E HEAGAWGHE-E-P-A-WHEAE -PA--W-HEAE -P--AW-HEAEHEAGAWGHEE HEAGAWGHEE HEAGAWGHEEP---AWHEAE -P--AWHEAE --PA-WHEAEYou can see that alternatives are more likely where gaps are required andevidence for the alignment is weak, as at the beginning of the sequences. Pairingsthat contribute strongly to the score, such as the Ws, or that come in blocks,as at the end of the sequence, are more stable. The frequency of a pairing in suchsamples can be used as a natural indicator of its reliability in the alignment. Below

4 Pairwise alignment using HMMswe present a direct way of calculating the expected value of this frequency, i.e.the probability that any particular pair of residues should be aligned, accordingto the model.The same type of sampling approach that we have used here will be used laterin the book when building multiple alignments (Chapter 5).Finding distinct suboptimal alignmentsAs mentioned above, a number of different methods have been given for findingalignments that are not simply minor variants of the optimal alignment. Oneapproach is to use the ‘repeat’ algorithm in Chapter 2. This found the optimalset of high-scoring matches between one sequence and multiple non-overlappingsegments of the other sequence. However, for the current purposes, this is unsatisfactorybecause it treats the two sequences differently. Also, the best singlealignment may not even be present in the set.The most widely used method for searching for distinct suboptimal alignmentsis due to Waterman & Eggert [1987], who give an algorithm to find the next bestalignment that has no aligned residue pairs in common with any previously determinedalignment. Once the top match has been obtained, the standard (Viterbi)dynamic programming matrix is recalculated, with the additional step during therecurrence that cells corresponding to residue pairs contained in the best matchare set to zero, preventing them from contributing to the next alignment. The resultingmatrix and score will therefore contain information about the second bestalignment. This procedure can be repeated, zeroing all the cells for any matchobtained so far each time, until the next score is below T (see Figure 4.5). In fact,if the matrix is stored in memory then it is not necessary to recalculate the completematrix each iteration: a marking procedure can be used to indicate whichcells need to be updated. For references to some of the other approaches to findingsuboptimal alignments, see Section 4.6.4.4 The posterior probability that xi is aligned to yjIf the probability of any single complete path being entirely correct is small,can we say anything about the local accuracy of an alignment? Often part ofan alignment is fairly clear, and other regions are less certain. The degree ofconservation varies depending on structural and functional contraints, so that coresequences may be well conserved, while loop regions are not reliably alignable.Given this situation, it can be useful to be able to give a reliability measure foreach part of an alignment.The HMM formalism allows us to do this. The idea is that we calculate thecombined probability of all the alignments that pass through a specified matched

4.4 The posterior probability that xi is aligned to yjHEAGAWGHEE

0P 0A 0W 0H 0E 0A 0E 0

000010200

H000021686

E0050082113

A0002001318

G00

5

000512

A000

20

12

4

0

4

W000

12

181040

G0 00 00 04 0

22

1418 2810 20

4 16

HE00006202726E

0 0 0 0 0 0 0 0 0 0 0P 0 0 0 0 0 0 0 0 0 0 0A 0 0 0 5 0 0 0 0 0 0 0W 0 0 0 0 2 0 0 0 0 0 0H 0 10 2 0 0 0 0 0 0 0 0E 0 2 16 8 0 0 0 0 0 0 6A 0 0 8 21 13 5 0 0 0 0 0E 0 0 6 13 18 12 4 0 0 6 6Figure 4.5 The Waterman–Eggert algorithm applied to our standard dataexample. Above, the standard local alignment matrix exactly as in Figure2.6. Below, the best local match has been zeroed out so that the secondbest alignment can be obtained.pair of residues (xi , yj ). We then compare this value with the full probability ofall alignments of the pair of sequences, calculated in the previous section. If theratio is near to one, then we can say that that match is highly reliable; if near zero,then the match is unreliable. This method used to do this is very closely relatedto the algorithm given for posterior decoding in Chapter 3.Let us introduce a new notation xi .yj to mean that xi is aligned to yj . Thenfrom standard conditional probability theory we haveP(x, y, xi .yj ) =P(x1...i , y1...j , xi .yj )P(xi+1...n, yj+1...m |x1...i , y1...j , xi .yj )=P(x1...i , y1...j , xi .yj )P(xi+1...n, yj+1...m |xi .yj )The first term is the forward probability f M(i, j) calculated above by the forwardalgorithm. The second is the corresponding backward probability bM(i, j) whichis calculated by the corresponding backward algorithm.

4 Pairwise alignment using HMMsAlgorithm: Backward calculation for pair HMMsInitialisation:bM(n, m) =bX(n, m) =bY(n, m) =τ.All b .(i, m +1), b .(n +1, j) are set to 0.Recursion: i =n, ...,1, j =m, ...,1 except (n, m);bM(i, j) =(1 .2δ.τ) pxi+1 yj+1 bM(i +1, j +1)+δqxi+1 bX(i +1, j) +qyj+1 bY(i, j +1) ;bX(i, j) =(1 .ε.τ) pxi+1 yj+1 bM(i +1, j +1) +εqxi+1 bX(i +1, j);bY(i, j) =(1 .ε.τ) pxi+1 yj+1 bM(i +1, j +1) +εqyj+1 bY(i, j +1). .There is no special termination step needed, because we only need the b .(i, j)values for i, j ≥1.We can now use Bayes’ rule to obtainP(x, y, xi .yj )

P(xi .yj |x, y) =,

P(x, y)and can also obtain similar values for the posterior probabilities of using specificinsert states. Figure 4.6 shows the results of this procedure applied to the examplesequences that we used in Chapter 2.Miyazawa [1994] describes essentially the same approach, and goes on to definewhat he calls a ‘probability alignment’. It might seem attractive to define analignment of x to y by finding for each i the j that maximises P(xi .yj ) (we dropexplicit conditioning with respect to x and y from here on, since it will always bepresent). However, this is not guaranteed to produce a well-formed alignment; itmay contain aligned pairs (i1, j1),(i2, j2) which are inconsistent with the sequenceorders, i.e. for which i2 >i1 and j1 <j2. Miyazawa pointed out that if we restrictourselves to pairs (i, j)for which P(xi .yj ) >0.5, then these will always be consistent,and will also only align each xi to at most one yj . In places where thealignment is clear, it will be covered by this condition. On the other hand, whereit is not clear, for example in corresponding loop regions of distantly related proteins,there will be gaps in both sequences where no particular pairs of residuesare strongly supported as being aligned.The expected accuracy of an alignmentMiyazawa’s approach typically gives rise to incomplete alignments, in that theremay be significant sections where no P(xi .yj ) >0.5. Although this may be whatis wanted, it is also possible to use the posterior match probabilities to give a completealignment with maximal overall accuracy, in the sense outlined below. Wefirst note that we can calculate the expected overlap A(π) between a given alignmentπand paths sampled from the posterior distribution. This is equivalently

4.4 The posterior probability that xi is aligned to yjMH E A GatchA W G H E E0 00 0 0 0 0 0 0 018 724 36 0 0 0 0 0 026 150 2 43 0 0 0 0 00 00 0 0 85 1 0 0 00 00 0 0 0 12 73 0 00 00 0 0 0 1 8 65 00 00 0 0 0 0 1 21 00 00 0 0 0 0 0 0 86X insertA G AH E W G H E E

P

A

W

H

E

A

E

P

A

W

H

E

A

E

P

A

W

H

E

A

E

87

0

0

0

0

0

0

0

00000000

7

32

200000

A0 036 028 420 00 00 00 00 0

Y insertGA00000000

W000

72

0000

G00000000

00000000

62

0000000

H2622000000

E00000000

00000000

00000000

00000000

00001000

00000000

00003000

H00000010

E00000002E

0 0 00 0 00 0 00 0 00 0 012 0 00 64 00 0 10Figure 4.6 Posterior probabilities for the example data used in Chapter 2.The three tables show the posterior probabilities of using the M, X or Ystates respectively at each (i, j) position. Values are shown as percentages,i.e. 100 times the relevant probability rounded to the nearest integer. Thepath indicated is the optimal accuracy path in the sense of (4.4).

4 Pairwise alignment using HMMsthe expected number of correct matches in π, which is a natural measure of theoverall accuracy of π.A(π) =P(xi .yj ),(i, j)∈πwhere the sum is over all aligned pairs in π. For the alpha globin/leghaemoglobinalignment of Figure 2.1(b) A(π) =16.48, or on average 0.40 per aligned residue.Given this new type of score for an alignment, can we find the alignment betweentwo sequences with the highest accuracy? We might hope that this, whileperhaps not providing the most discriminative score for use in detecting whethertwo sequences are related, would give a more accurate alignment if they are.The method required is surprisingly simple. We perform standard dynamic programmingusing score values given by the posterior probabilities of pair matches,without gap costs. The recursion equations are:..A(i .1, j .1) +P(xi .yj ),A(i, j) =max A(i .1, j), (4.4).A(i, j .1),and the standard traceback procedure will produce the best alignment. It is clearthat this procedure will optimise the sum of the P(xi .yj ) terms in a legitimatealignment. Interestingly the same algorithm works for any sort of gap score; whatwill change with different scores are the P(xi .yj ) terms themselves, which areobtained from the standard, scoring scheme-specific dynamic programming proceduresdescribed above.The optimal accuracy path for the short sequences used as examples in Chapter2 is shown in Figure 4.6. Note that it is not the same as the most likely, orViterbi path. The initial P in the shorter sequence is clearly preferably aligned tothe E and not the A of the longer sequence, although the individual scores foraligning P to E and A are the same. Intuitively, the reason for this is that aligningto the E allows more options in where the subsequent gap can be placed.4.5 Pair HMMs versus FSAs for searchingOne of the strong points of probabilistic modelling is that, if data D correspondto samples from a model M, then, in the limit of an infinitely large amount ofdata, the likelihood takes its maximum value for M,i.e. P(D|M) >P(D|M.

),where M.

is any other model. In particular, if M has a set of parameters, such asthe transition and emission probabilities of an HMM, the likelihood of the datawill be maximised by giving the model the parameter values corresponding to thesample.

4.5 Pair HMMs versus FSAs for searchingSBα1.α1 1 1abac1qa

Figure 4.7 This FSA emits sequences from S with probability qa, andstrings abac from the block B of four states below. If the probability oftransition to B is low, the most probable path will never use B, even if thesequence includes the motif abac.As a consequence, if the parameters of a pair HMM describe the statisticsof pairs of related sequences well, then we should use that model with thoseparameter values for searching. If we also have a model, R, that gives a gooddescription of the generation of random sequence, then Bayesian model comparisonwith M and R is an appropriate procedure (p. 36 in Chapter 2). Accordingto this philosophy, we should be using probabilistic models for searching. However,most currently used algorithms (Chapter 2) fall short of this in two ways.First, they do not compute the full probability P(x, y|M) of the pair of sequences,summing over all alignments, but instead find the best match, or Viterbi path.Second, regarded as FSAs, their parameters may not be readily translated intoprobabilities.Consider first the effects of using Viterbi paths. It is easy to show that, inthis case, a model whose parameters match the data need not be the best searchmodel. Figure 4.7 shows a simple HMM example. A state S generates symbolswith probabilities qa; S has a transition to itself with probability αand can makea transition with probability 1 .αto a sequential block B of states that emitsafixedstring abac of length four before returning to the original state. Theprobability of emitting abac from S is PS(abac) =α4qaqbqaqc, whereas theprobability of emitting abac from B (starting at S) is 1 .α.If PS(abac) >1 .α, the most probable path for any set of data will only use S, because thetransition to B is too improbable. Nonetheless, the presence of a greater thanexpected number of strings abac in the data is what distinguishes the output ofthe model from that of the random model that emits symbols with probabilitiesqa . Model comparison using the best match rather than the total probability, willfail to detect the source of the data, even for very large datasets. We can partiallycorrect for these deficiencies by changing our parameters. For instance, the model

4 Pairwise alignment using HMMs(a)0.2ab.2.12s(a,b)a.b.Begin0a. b. a. b.0 0Ends(a,b).12 s(a,b)0 0 0 0(b)0.50.50.080.64Begin0.2End0.50.08 0.50.8 0.8qa qb 0.2 0.2 0.2 0.20.8 0.8qaqa qbqbpab0.8 0.8Begin0.2Endqa qb 0.2Figure 4.8 (a) An FSA that computes the local match algorithm. s(a, b) arethe scores for the BLOSUM50 matrix. (b) Two HMMs, an aligned sequencemodel (above) and a random model (below) whose log-odds ratio score isthe same as the score of the FSA shown in (a). The probabilities pab and qaare those used to define the BLOSUM50 matrix.will be able to detect these types of sequences if the probability of the transitionto B is increased to τwhere τ>PS(abac). However, then every abac will beclassed as coming from B, which is not correct either.Consider now the problem of turning an FSA for pairwise alignment into aprobabilistic model. Figure 4.8(a) shows an FSA for local matches; it has initialand final states that emit an unpaired sequence with zero cost. Since the length

4.6 Further readingof this unpaired sequence can be arbitrary, and since a probabilistic model willalways have a non-zero cost for each emission, no fixed rescaling procedure canmake the scores of this model into the log probabilities of an HMM. On theother hand, if we are doing Bayesian model comparison, and if we define a randommodel R that emits an unpaired sequence with the same probabilities usedby the local alignment model M for its inital and final unaligned regions, thenthe log-odds for the unpaired sequence will be zero. We may then be able tofind two pair HMMs whose log-odds ratios match the FSA scores, for exampleFigure 4.8(b). Note that the transition probabilities here are not very plausible,since they imply very short sequences. Yet the parameters assumed for the FSAare known to work well. Based on this, we suspect that the standard parametershave been empirically set to ‘unconsciously’ compensate for the same failingof Viterbi as a search method as is illustrated in the simple case of Figure 4.8.This leads us to suggest that probabilistic models may underperform standardalignment methods if Viterbi is used for database searching, but if the forwardalgorithm is used to provide a complete score independent of specific alignment,then probabilistic models like pair HMMs may improve upon the standardmethods.Exercises4.4Show that using the full probabilistic model with the example in Figure4.7 allows discrimination between model and random data.4.5Compare this with using the Viterbi path in the model where the transitionprobability to B has been raised to τsuch that τ>PS(abac).4.6We can modify the model further by setting all the emission probabilitiesat S to the same value, 1/A, where A is the alphabet size. The differencebetween this model and a random model with the same emission probabilitiesis then precisely the number of strings abac in the data. Doesthis discriminate as well as the full probabilistic model?4.6 Further readingAlthough the explicit formulation of pairwise alignment in terms of pair hiddenMarkov models that we have given here is not standard, several authors have consideredan equivalent full probabilistic model. Bucher & Hofmann [1996] discusssearching with a local probabilistic model normalised via a partition function.Bishop & Thompson [1986] introduced a related model in the context of evolutionaryanalysis, a strand that has been developed more recently by Thorne,Kishino & Felsenstein [1991; 1992], who have developed parameter estimation

4 Pairwise alignment using HMMsmethods for probabilistic models of gapped alignment of DNA sequences. Wediscuss some of these evolutionary motivated models further in Chapter 8.Zuker [1991] and Barton [1993] describe methods for finding suboptimal alignmentsthat differ from the method of Waterman & Eggert [1987]. Mevissen &Vingron [1996] give an alternative approach to quantifying the reliability of adynamic programming alignment, and Vingron [1996] provides a good recentreview of methods for finding and assessing the significance of suboptimalalignments.

5 Profile HMMs for sequence families

So far we have concentrated on the intrinsic properties of single sequences, suchas CpG islands in DNA, or on pairwise alignment of sequences. However, functionalbiological sequences typically come in families, and many of the mostpowerful sequence analysis methods are based on identifying the relationship ofan individual sequence to a sequence family. Sequences in a family will havediverged from each other in their primary sequence during evolution, having separatedeither by a duplication in the genome, or by speciation giving rise to correspondingsequences in related organisms. In either case they normally maintainthe same or a related function. Therefore, identifying that a sequence belongs toa family, and aligning it to the other members, often allows inferences about itsfunction.If you already have a set of sequences belonging to a family, you can performa database search for more members using pairwise alignment with one of theknown family members as the query sequence. To be more thorough, you couldeven search with all the known members one by one. However, pairwise searchingwith any one of the members may not find sequences distantly related to theones you have already. An alternative approach is to use statistical features of thewhole set of sequences in the search. Similarly, even when family membership isclear, accurate alignment can be often be improved significantly by concentratingon features that are conserved in the whole family.How, in brief, do we identify such features? Just as a pairwise alignment capturesmuch of the relationship between two sequences, a multiple alignment canshow how the sequences in a family relate to each other. Figure 5.1 shows amultiple alignment of seven sequences from the large globin family (hundredsof globin sequences are available in the protein sequence databases). The threedimensional structure has been obtained for each protein in the alignment shown,and the sequences have been aligned on the basis of aligning the eight alphahelices of the conserved globin fold, and also on the basis of aligning certainkey residues in the sequences, such as two conserved histidines (H) which arethe residues which interact with an oxygen-binding heme prosthetic group in theglobin active site.It is clear that some positions in the globin alignment are more conserved thanothers. In general the helices are more conserved than the loop regions between101

5 Profile HMMs for sequence familiesHelix AAAAAAAAAAAAAAAA BBBBBBBBBBBBBBBBCCCCCCCCCCCHBA\_HUMAN ---------VLSPADKTNVKAAWGKVGA--HAGEYGAEALERMFLSFPTTKTYFPHFHBB\_HUMAN --------VHLTPEEKSAVTALWGKV----NVDEVGGEALGRLLVVYPWTQRFFESFMYG\_PHYCA ---------VLSEGEWQLVLHVWAKVEA--DVAGHGQDILIRLFKSHPETLEKFDRFGLB3\_CHITP ----------LSADQISTVQASFDKVKG------DPVGILYAVFKADPSIMAKFTQFGLB5\_PETMA PIVDTGSVAPLSAAEKTKIRSAWAPVYS--TYETSGVDILVKFFTSTPAAQEFFPKFLGB2\_LUPLU --------GALTESQAALVKSSWEEFNA--NIPKHTHRFFILVLEIAPAAKDLFS-FGLB1\_GLYDI ---------GLSAAQRQVIAATWKDIAGADNGAGVGKDCLIKFLSAHPQMAAVFG-FConsensus Ls.... vaWkv. . g.L..f.P. F FHelix DDDDDDDEEEEEEEEEEEEEEEEEEEEE FFFFFFFFFFFFHBA\_HUMAN -DLS-----HGSAQVKGHGKKVADALTNAVAHV---D--DMPNALSALSDLHAHKLHBB\_

HUMAN GDLSTPDAVMGNPKVKAHGKKVLGAFSDGLAHL---D--NLKGTFATLSELHCDKLMYG\_

PHYCA KHLKTEAEMKASEDLKKHGVTVLTALGAILKK----K-GHHEAELKPLAQSHATKHGLB3\_

CHITP AG-KDLESIKGTAPFETHANRIVGFFSKIIGEL--P---NIEADVNTFVASHKPRGGLB5\_

PETMA KGLTTADQLKKSADVRWHAERIINAVNDAVASM--DDTEKMSMKLRDLSGKHAKSFLGB2\_

LUPLU LK-GTSEVPQNNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKGGLB1\_

GLYDI SG----AS---DPGVAALGAKVLAQIGVAVSHL--GDEGKMVAQMKAVGVRHKGYGNConsensus . t ...v..Hgkv.a a...l d .al.l H .Helix FFGGGGGGGGGGGGGGGGGGG HHHHHHHHHHHHHHHHHHHHHHHHHHHBA\_HUMAN -RVDPVNFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSKYR-----HBB\_

HUMAN -HVDPENFRLLGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKYH-----MYG\_

PHYCA -KIPIKYLEFISEAIIHVLHSRHPGDFGADAQGAMNKALELFRKDIAAKYKELGYQGGLB3\_CHITP --VTHDQLNNFRAGFVSYMKAHT--DFA-GAEAAWGATLDTFFGMIFSKM------GLB5\_

PETMA -QVDPQYFKVLAAVIADTVAAG---------DAGFEKLMSMICILLRSAY------LGB2\_

LUPLU --VADAHFPVVKEAILKTIKEVVGAKWSEELNSAWTIAYDELAIVIKKEMNDAA--GLB1\_

GLYDI KHIKAQYFEPLGASLLSAMEHRIGGKMNAAAKDAWAAAYADISGALISGLQS----Consensusv. f l... .... f .aa.k.. lskyFigure 5.1 An alignment of seven globins from Bashford, Chothia &Lesk [1987]. To the left is the protein identifier in the SWISS-PROTdatabase [Bairoch & Apweiler 1997]. The eight alpha helices are shown asA–H above the alignment. A consensus line below the alignment indicatesresidues that are identical among at least six of the seven sequences in uppercase, ones identical in four or five sequences in lower case, and positionswhere there is a residue identical in three sequences with a dot.them, and certain residues are particularly strongly conserved. When identifyinga new sequence as a globin, it would be desirable to concentrate on checking thatthese more conserved features are present. How to obtain and use such informationwill be the subject of this chapter.As might be expected, our approach to consensus modelling will be to makea probabilistic model. In particular, we will develop a particular type of hiddenMarkov model well suited to modelling multiple alignments. We call these profileHMMs after standard profiles, which are closely related non-probabilistic structuresintroduced previously for the same purpose by Gribskov, McLachlan &Eisenberg [1987]. Profile HMMs are probably the most popular application ofhidden Markov models in molecular biology at the moment [Eddy 1996].We will assume for the purposes of this chapter that we are given a correctmultiple alignment, from which we will build a model that can be used to find andscore potential matches to new sequences. The multiple alignment could be built

5.1 Ungapped score matricesfrom structural information, like the globin alignment shown here, or it couldcome from a sequence-based alignment procedure, such as those discussed inChapter 6.Much of this chapter makes use of the theory presented in Chapter 3 for generalHMMs. The most important algorithms will be presented again in the specificform relevant to profile HMMs. There is also an extensive discussion of how toestimate optimal probability parameters from multiple sequence alignments.5.1 Ungapped score matricesOne general feature of protein family multiple alignments, which can be seenin Figure 5.1, is that gaps tend to line up with each other, leaving solid blockswhere there are no insertions or deletions in any of the sequences. We will startby considering models for these ungapped regions.As an example, consider the E helix of Figure 5.1. A natural probabilisticmodel for such a region would be to specify independent probabilities ei (a)ofobserving amino acid a in position i (we use letter e because these will turn outto be the emission probabilities of the hidden Markov model when we introducegaps). The probability of a new sequence x according to this model is thenLP(x|M) =ei (xi ),i=1where L is the length of the block, 21 in this case. As usual, we are in fact moreinterested in the ratio of this probability to the probability of x under a randommodel, and so to test for membership in the family we evaluate the log-odds ratioL ei (xi )

S =log .qxi

i=1The values log ei (a) behave like elements in a score matrix s(a, b), where theqasecond index is position i, rather than amino acid b. For this reason, such anapproach is known as a position specific score matrix (PSSM). A PSSM can beused to search for a match in a longer sequence x of length N by evaluating thescore Sj for each starting point j in x from 1 to N .L +1, where L is the lengthof the PSSM.5.2 Adding insert and delete states to obtain profile HMMsAlthough a PSSM captures some conservation information, it is clearly an inadequaterepresentation of all the information in a multiple alignment of a protein

5 Profile HMMs for sequence familiesfamily. We have to find some way to take account of gaps. It is possible to combinethe scores of multiple ungapped block models, and this is the approach takenby Henikoff & Henikoff [1991] in the BLOCKS database. However, we will pursuehere the aim of developing a single probabilistic model for the whole extentof the alignment.One approach is to allow gaps at each position in the alignment, using thesame gap score γ(g) at each position, as in pairwise alignment. However, thisis also ignoring information, because the alignment gives us explicit indicationsof where gaps are more and less likely. We want to capture this information togive us position sensitive gap scores, just as the emission probabilities gave usposition sensitive substitution scores.The approach we take is to build a hidden Markov model (HMM), with a repetitivestructure of states, but different probabilities in each position. This will providea full probabilistic model for sequences in the sequence family. We start offby observing that the PSSM can be viewed as a trivial HMM with a series ofidentical states that we will call match states, separated by transitions of probability1.BeginEnd

MjAlignment is trivial because there is no choice of transitions. We rename theemission probabilities for the match states to eMi (a).The next step is to deal with gaps. We must treat insertions and deletions separately.To handle insertions, i.e. portions of x that do not match anything in themodel, we introduce a set of new states Ii , where Ii will be used to match insertionsafter the residue matching the ith column of the multiple alignment. TheIi have emission distribution eIi (a), but these are normally set to the backgrounddistribution qa, just as for seeing an unaligned inserted residue in a pairwise alignment.We need transitions from Mi to Ii , a loop transition from Ii to itself, to accommodatemulti-residue insertions, and a transition back from Ii to Mi+1.Hereis a single insert state of this kind:MjIjBegin EndWe denote insert states in our diagrams by diamonds. The log-odds cost of aninsert is the sum of the costs of the relevant transitions and emissions. Assumingthat eIi (a) =qa as described above, there is no log-odds contribution from theemission, and the score of a gap of length k islog aMj Ij +logaIj Mj+1 +(k .1)logaIj Ij .

5.2 Adding insert and delete states to obtain profile HMMs 105From this you can see that the type of insert state shown corresponds to an affinegap scoring model.Deletions, i.e. segments of the multiple alignment that are not matched byany residue in x, could be handled by forward ‘jump’ transitions between nonneighbouringmatch states:However, to allow arbitrarily long gaps in a long model this way would require alot of transitions. Instead we introduce silent states Dj as described in Section 3.4:DjBeginEnd

MjBecause the silent states do not emit any residues, it is possible to use a sequenceof them to get from any match state to any later one, between two residues in thesequence. The cost of a deletion will then be the sum of the costs of an M →Dtransition followed by a number of D →D transitions, then a D →M transition.This is at first sight exactly analogous to the cost of an insert, although the paththrough the model looks different. In detail, it is possible that the D →D transitionswill have different probabilities, and hence contribute differently to thescore, whereas all the I →I transitions for one insert involve the same state, andso are guaranteed to have the same cost.The full resulting HMM has the structure shown in Figure 5.2. This form ofmodel, which we call a profile HMM, was first introduced in Haussler et al.[1993] and Krogh et al. [1994]. We have added transitions between insert anddelete states, as they did, although these are usually very improbable. Leavingthem out has negligible effect on scoring a match, but can create problems whenbuilding the model.Begin EndMjDjIjFigure 5.2 The transition structure of a profile HMM. We use diamonds toindicate the insert states and and circles for the delete states.

5 Profile HMMs for sequence familiesProfile HMMs generalise pairwise alignmentWe have seen how the costs of using gap states in a profile HMM mirror thoseused in pairwise alignment with affine gaps. To help make clear the relationship,it is useful to consider the degenerate case where the multiple alignment fromwhich we build the HMM contains just one sequence.Let us compare Figure 5.2 with Figure 4.2. If we call the example sequence y,then Figure 5.2 is an unrolled version of Figure 4.2, with the yj emissions eachcoming from a separate copy of the pair HMM. The states Mj correspond to asequence of match states M, the Ij to corresponding incarnations of X, and the Djto incarnations of Y. To achieve as close a correspondence as possible, the naturalvalues for the match emission probabilities eMi (a)are pyi a /qyi , the conditionalprobabilities of seeing a given yi in a pairwise alignment, and for the transitionprobabilities aMi Ii =aMi Di+1 =δand aIi Ii =aDi Di+1 =εfor all i.In formal terms our profile HMM is effectively the hidden Markov model obtainedby conditioning the pair HMM of Figure 4.2 on emitting sequence y asone of the sequences in its alignment. Because of this, the Viterbi equations forfinding the most probable alignment of x to our profile HMM are essentially thesame as those for the most probable alignment of x and y to the pair HMM describedin Chapter 4. If we convert them into log-odds ratio form we recover ourstandard affine gap cost pairwise alignment equations of (2.16), as we will seebelow. Any differences are due to slightly different Begin and End arrangements.5.3 Deriving profile HMMs from multiple alignmentsAlthough it is nice to see that the profile HMM is doing the same sort of dynamicprogramming as we have used before for pairwise alignment, this is not why weintroduced them. The key idea behind profile HMMs is that we can use the samestructure as shown in Figure 5.2, but set the transition and emission probabilitiesto capture specific information about each position in the multiple alignment ofthe whole family. Essentially, we want to build a model representing the consensussequence for the family, not the sequence of any particular member.There are a number of different ways to derive the parameter values from amultiple alignment of the sequences in the family. To provide an example forillustrating these methods, Figure 5.3 shows a short section of the globin alignmentshown in Figure 5.1.Non-probabilistic profilesA model similar to the profile HMM was first introduced by Gribskov, McLachlan& Eisenberg [1987] who coined the name ‘profile’ (see also Gribskov, Lüthy &Eisenberg [1990]). However, they did not have an underlying probabilistic model,

5.3 Deriving profile HMMs from multiple alignmentsHBA\_HUMAN ...VGA--HAGEY...HBB\_HUMAN ...V----NVDEV...MYG\_PHYCA ...VEA--DVAGH...GLB3\_CHITP ...VKG------D...GLB5\_PETMA ...VYS--TYETS...LGB2\_LUPLU ...FNA--NIPKH...GLB1\_GLYDI ...IAGADNGAGV...\*\*\* \*\*\*\*\*Figure 5.3 Ten columns from the multiple alignment of seven globin proteinsequences shown in Figure 5.1. The starred columns are ones that will betreated as ‘matches’ in the profile HMM.but rather directly assigned position specific scores for each match state and gappenalty, for use in standard ‘best match’ dynamic programming. They set thescores for each consensus position to the averages of the standard substitutionscores from all the residues seen in the corresponding multiple alignment column.For example, they would set the score for residue a in column 1 of our exampleto be57 s(V, a) +17 s(F, a) +17 s(I,a)where s(a, b) is the standard substitution matrix. They also set gap penalties foreach column using a heuristic equation that decreased the cost of a gap (eitherinsertion or deletion) according to the length of the longest gap observed in themultiple alignment spanning the column.Although this seems an intuitively obvious way to combine information, and ithas been used effectively by many people for finding new members of families,it does produce anomalies. For example, column 1 is much more strongly conservedthan column 2 in the example shown in Figure 5.3, but the informationin column 1 will be smeared out just as much by the substitution matrix as thatin column 2. If we had an alignment with 100 sequences, all with a cysteine (C)at some position, then the implicit probability distribution for that column for an‘average’ profile would be exactly the same as would be derived from a singlesequence. This does not correspond to our expectation that the likelihood of acysteine should go up as we see more confirming examples.In addition to these observations about substitution scores, the scores for gapsdo not behave as expected. For example, from the alignment in Figure 5.3 thescore for a deletion would be set to be the same in column 2, where there is adeletion in one sequence, HBB\_HUMAN, as in column 4, where there is a deletionopening in five of the seven sequences. It would be more reasonable to set theprobability of a new gap opening to be higher in column 4.

5 Profile HMMs for sequence familiesChanges have been made to non-probabilistic profiles to address these andother problems [Thompson, Higgins & Gibson 1994b; Gribskov & Veretnik1996], and we shall return to some of these later.Basic profile HMM parameterisationLet us turn back to hidden Markov model profiles. Like all HMMs, these haveemission and transition probabilities. Assuming that these probabilities are nonzero,a profile HMM can model any possible sequence of residues from the givenalphabet. It therefore defines a probability distribution over the whole space ofsequences. The aim of the parameterisation process it to make this distributionpeak around members of the family.The parameters we have available to control the shape of the distribution arethe values of the probabilities, and also the length of the model. There is a lot tosay about setting these optimally. We give here the basic methods from Krogh etal. [1994]. After sections on database searching and variants for local alignment,we will return to an extended discussion of alternative parameter estimation techniques.The choice of length of the model corresponds more precisely to a decision onwhich multiple alignment columns to assign to match states, and which to assignto insert states. The profile HMM we derived above from the single sequence yhad a match state for each residue yi . However, looking at Figure 5.3 it seemsclear that the consensus sequence for this region should only have eight residues,and that the two non-starred residues in GLB1\_GLYDI should be treated as aninsertion with respect to the consensus. For the time being we will use a heuristicrule to decide which columns should correspond to match states, and which toinserts. A simple rule that works well is that columns that are more than half gapcharacters should be modelled by inserts.The second problem is how to assign the probability parameters. We regard thealignment as providing a set of independent samples of alignments of sequencesx to our HMM. Since the alignments are given, we can estimate the parametersdirectly using equations (3.18) from Section 3.3. We just count up the number oftimes each transition or emission is used, and assign probabilities according toAkl Ek (a)akl =.and ek (a) =.l.Akl.a.Ek (a )where k and l are indices over states, and akl and ek are the transition and emissionprobabilities, and Akl and Ek are the corresponding frequencies.In the limit of having a very large number of sequences in our training alignment,this will give an accurate and consistent estimate of the probabilities. However,it has problems when there are only a few sequences. A major difficultyis that some transitions or emissions may not be seen in the training alignment,

5.4 Searching with profile HMMs33BEGIN33YWVTSRQPNMLKIHGFEDCA133YWVTSRQPNMLKIHGFEDCA240YWVTSRQPNMLKIHGFEDCA333YWVTSRQPNMLKIHGFEDCA433YWVTSRQPNMLKIHGFEDCA533YWVTSRQPNMLKIHGFEDCA633YWVTSRQPNMLKIHGFEDCA750YWVTSRQPNMLKIHGFEDCA8ENDFigure 5.4 A hidden Markov model derived from the small alignmentshown in Figure 5.3 using Laplace’s rule. Emission probabilities are shownas bars opposite the different amino acids for each match state, and transitionprobabilities are indicated by the thickness of the lines. The I →Itransition probabilities times 100 are shown in the insert states. (Figuregenerated automatically using the SAM package.)and so would acquire zero probability, which would mean they would never beallowed in the future. More broadly, we are not using any previous knowledgeabout protein alignments, as the earlier non-probabilistic methods did implicitly,by using an independently derived substitution matrix. As a minimal approachto avoid zero probabilities, we can add pseudocounts to the observed frequencies(as in Chapters 1 and 3). The simplest pseudocount method is Laplace’s rule: toadd one to each frequency. We discuss better ways to choose the pseudocount values,and other approaches to estimating the parameters, at greater length belowin Section 5.6.Example: Parameters for an HMM based on Figure 5.3Let us assume that we use Laplace’s rule to obtain parameters for an HMM correspondingto the alignment in Figure 5.3. Then eM1(V) =6/27, eM1(I) =eM1(F) =2/27, and eM1(a) =1/27 for all residue types a other than V, I, F. Similarly,aM1M2 =7/10, aM1D2 =2/10 and aM1I1 =1/10 (following column 1 there are sixtransitions from match to match, one transition to a delete state, in HBB\_HUMAN,and no insertions). Figure 5.4 shows the complete set of parameters for the HMMin diagrammatic form.5.4 Searching with profile HMMsOne of the main purposes of developing profile HMMs is to use them to detect potentialmembership in a family by obtaining significant matches of a sequence tothe profile HMM. We will assume for now that we are looking for global matches.

5 Profile HMMs for sequence familiesIn practice, as for pairwise alignment, one of the local alignment methods may bemore sensitive for finding distant matches. We discuss these in the next section.We have a choice of ways to score a match to a hidden Markov model. Wecan either use the Viterbi equations to give the most probable alignment π.of asequence x together with its probability P(x, π.|M), or the forward equations tocalculate the full probability of x summed over all possible paths P(x|M).In either case, for practical purposes the result we want to consider when evaluatingpotential matches is the log-odds ratio of the resulting probability to theprobability of x given our standard random modelP(x|R) =qxi .iWe therefore show here versions of the Viterbi and forward algorithms that aredesigned specifically for profile HMMs, and which result directly in the desiredlog-odds values. Note that changing to log-odds does not change the result; wecould have subtracted the random model log score afterwards. However, it iscleaner and more efficient. Another practical reason for working in log-odds unitsis to avoid problems of underflow when working with raw probabilities, as wediscussed in Section 3.6.Viterbi equationsLet VjM(i) be the log-odds score of the best path matching subsequence x1...i tothe submodel up to state j, ending with xi being emitted by state Mj . SimilarlyVjI(i) is the score of the best path ending in xi being emitted by Ij , and V D(i)forjthe best path ending in state Dj . Then we can write..VjM.1(i .1) +logaMj.1Mj ,.

eMj (xi )

VjM(i) =log +max VjI.1(i .1) +logaIj.1Mj ,.

qxi.VjD.1(i .1) +logaDj.1Mj ;..VjM(i .1) +log aMj Ij ,.

eIj (xi )

VjI(i) =log +max VjI(i .1) +log aIj Ij , (5.1).

qxi.V D(i .1) +log aDj Ij ;

j..VjM.1(i) +log aMj.1Dj ,.VjD(i) =max VjI.1(i) +log aIj.1Dj ,..VjD.1(i) +log aDj.1Dj .These are the general equations. In a typical case, there is no emission scoreeIj (xi ) in the equation for VjI(i) because we assume that the emission distributionfrom the insert states Ij is the same as the background distribution, so the probabilitiescancel in the log-odds form. Also, the D →I and I →D transition termsmay not be present, as discussed above.

5.4 Searching with profile HMMsWe need to take a little care over initialisation and termination of the dynamicprogramming. We want to allow the alignment to start and end in a delete or insertstate, in case the beginning or end of the sequence does not match the first or thelast match state of the model. The simplest way to ensure this mechanistically isto rename the Begin state as M0 and set V M(0) =0 (as we did in Chapter 3). We0then allow transitions to I0 and D1. Similarly, at the end we can collect togetherpossible paths ending in insert and delete states by renaming the End state toML+1 and using the top relation without the emission term to calculate VLM+1(n)as the final score.If these recurrence equations are compared with those for standard gapped dynamicprogramming in (2.16), it can be seen that apart from renaming of variablesthis is the same algorithm, but with the substitution, gap-open and gap-extendscores all depending on position in the model, j.Forward algorithmThe recurrence equations for the forward algorithm are similar to the Viterbiequations, but with the max() operation replaced by addition. We define variablesFM(i), FjI(i) and FjD(i) for the partial full log-odds ratios, corresponding tojV M(i), VjI(i) and V D(i). The recurrence equations are then:jj(xi ). .FjM(i) =logeMj +log aMj.1Mj exp FjM.1(i .1)qxi .

.+aIj.1Mj exp FjI.1(i .1) +aDj.1Mj exp FjD.1(i .1) ;(xi ) ..FjI(i) =logeIj +log aMj Ij exp FjM(i .1)qxi .

.+aIj Ij exp FjI(i .1) +aDj Ij exp FjD(i .1) ;. . .FjD(i) =log aMj.1Dj exp FjM.1(i) +aIj.1Dj exp FjI.1(i)+aDj.1Dj exp FjD.1(i).Initialisation and termination conditions are handled as for the Viterbi case, withFM(0) being initialised to 0.0Although these appear a little complicated, in a practical implementation theoperation log(ex +ey) can be performed efficiently to adequate accuracy by functionlookup and interpolation; see Section 3.6.Alternatives to log-odds scoringIn some of the earlier papers on HMMs, rather than calculating the log-odds scorerelative to a random model, the logarithm of the probability of the sequence giventhe model was used directly. This was called the LL score for ‘log likelihood’:LL(x) =log P(x|M). The LL score is strongly length dependent, so for searching

112 5 Profile HMMs for sequence families-6-5-4-3-2-1

00 50 100 150 200 250 300LL/lengthprotein lengthnon-globinstraining dataother globins-200-100

01002003004005000 50 100 150 200 250 300log-oddsprotein lengthnon-globinstraining dataother globinsFigure 5.5 To the left the length-normalized LL score is shown as a functionof sequence length. The right plot shows the same for the log-odds score.it is not good enough to use a simple threshold. It is better to use LL divided bythe sequence length, but even that is not always perfect, because the dependencebetween LL and sequence length is not linear (see example below).A way to get around this is to estimate an average score and a standard deviationas a function of length and then use the number of standard deviationseach sequence is away from the average. This is called the Z-score, and is alsoillustrated in the example below.Example: Modelling and searching for globinsFrom 300 randomly picked globin sequences a profile HMM was estimated fromscratch, i.e. starting from unaligned sequences using procedures we will explainin Chapter 6. A simple pseudocount regulariser was used. The estimation wasdone several times and the model with the highest overall LL score was picked.(We used the default settings of the SAM package, version 1.2; Hughey & Krogh[1996]).With this model a database of about 60 000 proteins (SWISS-PROT release 34;Bairoch & Apweiler [1997]) was searched using the forward algorithm. The LLand log-odds scores were found for each sequence. For the null model we usedthe amino acid frequencies of the 300 sequences in the training set. In Figure 5.5the length-normalised scores are shown for all the globins in the training set, allthe other globins in the database and all the rest of the proteins with lengths upto 300 amino acids.1 The globin sequences are clearly separated from the nonglobinsapart from a few in the ‘twilight zone.’The main difference between the two is in the variance of the score for nonglobins,which is lower for the log-odds score, and therefore the separation isclearer. However, just choosing a cut-off of zero for the log-odds would miss a1 A few dubious globins and other strange sequences were removed from these data.

5.4 Searching with profile HMMs 1130

510152025

50 100 150 200 250 300Z-score from LLprotein lengthnon-globinstraining dataother globins0102030405060708090

50 100 150 200 250 300Z-score from log-oddsprotein lengthnon-globinstraining dataother globinsFigure 5.6 The Z-score calculated from the LL scores (left) and the log-odds (right).lot of real globins in the search. This is because the profile HMM is not broadenough: it is too concentrated on a subset of the globins. Although there are waysto address this problem directly that we will return to later in the chapter, it isalso possible to take a pragmatic approach to the separation of signal from noisegiven the results of the search, and calculate Z-scores for each hit.To calculate Z-scores, a smooth curve is fitted to the LL or log-odds score of thenon-globin sequences (a method is outlined in Krogh et al. [1994]). A standarddeviation is then estimated for each length (or rather a little interval around it),and for each score the distance from the smooth curve is calculated in units of thestandard deviation. This is the Z-score. The result (still as a function of sequencelength) is shown in Figure 5.6.2It is evident that it is now possible to find a threshold which will separate mostglobins from all other sequences. It is also clear that the score based on log-oddsis much better for discrimination, with approximately three times the signal tonoise ratio of the LL score. The reason for this is that dividing by the probabilityof the random model adjusts for the residue composition of the sequence.Withoutdoing that, sequences with similar residue compositions as globins will tend toscore more highly than sequences containing different residues, increasing thevariance of the noise.AlignmentAside from finding matches, the other principal use of profile HMMs is to givean alignment of a sequence to the family, or more precisely to add it into themultiple alignment of the family. This is primarily the subject of the next chapter,2 There is no analytical result about the shape of these score distributions. The global alignmentdistribution is probably not exactly a Gaussian [Waterman 1995], but it appears to bea good approximation. For local alignments the extreme value distribution may be morereasonable, as discussed in Chapter 2.

5 Profile HMMs for sequence familieson multiple alignment methods, which covers alignment with profile HMMs atlength. For now, we will just point out that the natural solution is to take the highestscoring, or Viterbi, alignment. This is obtained by tracing back on the Viterbivariables Vj.(i), exactly as with pairwise alignment. Beyond this, all the methodsof Chapter 4 can be applied, to explore variants, and to assess the reliability ofthe alignment.5.5 Profile HMM variants for non-global alignmentsWe have seen that there is a very close relationship between the Viterbi alignmentof a sequence to a profile HMM and the global dynamic programming comparisonbetween two sequences using affine gap penalties, which we described inChapter 2. It is therefore possible to generalise all the variations of dynamic programming,such as those that find local, repeat and overlap matches, to use profileHMMs.However, we have developed probabilistic models much more fully sinceChapter 2, and this time we want to take more care to ensure that the result ofconverting to a local algorithm remains a proper probabilistic model, i.e. thatwe assign each sequence a true probability so that the sum over all sequencesP(x|M) =1. Our approach to doing this is to specify a new model for thexcomplete sequence x, which incorporates the original profile HMM together withone or more copies of a simple self-looping model that is used to account for theregions of unaligned sequence. These behave very like the insert states that weadded to the profile itself. We call them flanking model states, because they areused to model the flanking sequences to the actual profile match itself.The model for local (Smith–Waterman style) alignment is shown here:BeginQEndQThe flanking model states are shown as shaded diamonds. Notice that as wellas specifying the emission probabilities of the new states, which will normallyof course be qa, we must specify a number of new transition probabilities. Thelooping probability on the flanking states should be close to 1, since they must

5.5 Profile HMM variants for non-global alignmentsaccount for long stretches of sequence. Let us set these to (1 .η). Note also thatwe have made use of silent states, shown as shaded circles, as ‘switching points’to reduce the total number of transitions.The next issue is how to set all the transition probabilities from the left flankingstate to different start points in the model. One option is to give them equal probabilities,η/L. Another is to assign more probability to starting at the beginningof the model. The default option in the HMMER package for profile HMMs [Eddy1996] assigns probability η/2 to the start of the profile, and η/(2(L .1)) to theother positions, favouring matches that start at the beginning of the model.If all the probability is assigned to the first model state, then it forces this modelto match only complete copies of the profile in the searched sequence, ensuringa type of ‘overlap’ match constraint. This can be appropriate when, for example,the HMM represents a protein domain that you expect to find either present as awhole or absent. However, to allow for rare cases where the first residue might bemissing, it may be wise in such cases to allow a direct transition from the flankingstate into a delete state, as shown here:EndBeginQ QIt is clear that by tinkering with the transition connections and probabilities awide variety of different models can be produced, each potentially useful in differentcircumstances. A final example similar to the first model for local matchesisBegin Q Endwhich allows repeat matches to subsections of the profile model, like the repeatalgorithm variant in Chapter 2.Note that all these variants of transition connectivity and probability assignmentaffect not only the types of match that are allowed, but also the score. More

5 Profile HMMs for sequence familiesrestrictive transition distributions will give higher match scores if a good match isfound, so are preferable if they can be designed to represent the types of correctmatches that are expected.Exercises5.1Show that if the random model is the same as that described in Chapter 4(a succession of two states looping on themselves with probability (1 .η)), with ηthe same as in the flanking models, the local alignment modelgives update equations like those of equation (2.9).5.2Explain the reasons for any differences.5.6 More on estimation of probabilitiesAs promised above, we now return to the subject of parameter estimation atgreater length. Although our discussion for most of this section will be focusedon the emission probabilities, analogous methods can be used for the transitionprobabilities. The aim here is to introduce methods that can be used. A more detailedmathematical discussion about the estimation of probabilites from samplecounts is given in Chapter 11 (p. 312).The most straightforward approach to parameter estimation would be to givethe maximum likelihood estimates for the parameters. We will change notationslightly from that used before. Given observed frequencies cja of residue a inposition j of the alignment, maximum likelihood estimates for eMj (a), the correspondingmodel parameters, arecja

eMj (a) =..(5.2)a.cja.As we described above, a clear problem with this is that if there are no observedexamples of a particular outcome then its probability is estimated as zero. Thiswill frequently occur. For example, in the alignment of Figure 5.3 only V, IandFare present in the first column. However, it is quite likely that other amino acidswill occur in that position amongst all the other globin sequences in biology. Theeasiest way to deal with this problem is to add pseudocounts to the observedcounts cja. Below, we first discuss the pseudocount approach at greater length,then give some more complex alternatives.Simple pseudocountsA very simple and much-used pseudocount method is to add a constant to all thecounts, which prevents the problem with zero probabilities. When the constant isone, as we used above in our example, this is called ‘Laplace’s rule’. A slightly

5.6 More on estimation of probabilitiesmore sophisticated method is to add a quantity proportional to the backgrounddistribution, givingcja +AqaeMj (a) =., (5.3)a.cja +Awhere cja are the real counts, and A is the weight put on the pseudocounts ascompared to the real counts. Values of A of around twenty seem to work well forprotein alignments.This form of regularisation has the appealing feature that eMj (a) is approximatelyequal to qa if very little data is available, i.e. all the real counts are verysmall compared to A. At the other extreme, where a large amount of data is available,the effect of the regulariser becomes insignificant and eMj (a) is essentiallyequal to the maximum likelihood solution. So, at this intuitive level, pseudocountsmake a lot of sense.Adding pseudocounts amounts to adding some fake imagined data into thealignment, based on our general knowledge of proteins, to represent all the otherthings that might happen. They thus correspond to prior information about proteinfamilies, before having seen the specific data for the family in the form of thealignment. This statement can be formalised in a Bayesian framework. Bayes’equation tells us how to combine data, D, with a prior probability distributionover the parameters P(θ) to give a posterior distribution over θ, from which wecan take either the maximum or the mean as our best estimate,P(D|θ)P(θ)

P(θ|D) =.P(D)In our case the parameters θare our model probabilities. The pseudocountmethod given above corresponds in this Bayesian framework to assuming aDirichlet prior distribution with parameters αa =Aqa over the probabilities; seeChapter 11 for mathematical details.Dirichlet mixturesThe problem with the simple pseudocounts, as compared to the substitution matrixbased methods, is that only the most rudimentary prior knowledge can becontained in a single pseudocount vector. For this reason we need a lot of exampledata in the alignment to get good estimates of the parameters. Experiencesuggests that to achieve good discrimination typically fifty or more examples aredesirable when modelling proteins.In order to include better prior information, it was therefore suggested byBrown et al. [1993] that one should use a mixture of Dirichlet distributions asthe prior. The idea is that there might be several different sets of pseudocount priorsα1., ..., αK corresponding to different types of alignment environments, where.

5 Profile HMMs for sequence familiesαk corresponds to Aqa in the example above. One set might be relevant for ex-

aposed loop environments, one for buried small residue environments, etc. Givenour counts cja we first estimate how likely each prior distribution k is (based onhow well it fits the observed data), then combine their effects according to theseposterior probabilities:cja +αakeMj (a) =P(k|cj ).,a.(cja +αak.)

kwhere the P(k|ci )are the posterior mixture coefficients. We calculate these byBayes’ rule,pk P(cj |k)

P(k|cj ) =.k.pk.P(cj |k )where the pk are the prior probabilities of each mixture component, and P(cj |k)is the probability of the data according to Dirichlet mixture k. The equation forP(cj |k) has a frightening looking form, which is in fact fairly simple to calculate:a cja ! a

(cja +αak) .a αakP(cj |k) =. ..,cja! .cja +αk

(αak )

a aaawhere

(x) is the gamma function, a standard function over the reals related tothe factorial function on the integers. For further details and an explanation of thisequation, see Chapter 11, where we also describe how the mixture componentdistributions αk.are obtained.Using this type of approach, it seems that good profile HMMs can be fit toalignments with as few as ten or twenty examples [Sj.lander et al. 1996].Substitution matrix mixturesAn alternative approach to using a mixture of Dirichlets is to adjust the pseudo-

counts in a single Dirichlet formulation, using information from the observedcounts and a substitution matrix. This is not a theoretically well-founded approach,but it makes intuitive sense as a heuristic, combining features of the non-

probabilistic profile methods and the Dirichlet pseudocount methods.The first step is to convert the matrix entries s(a, b) into conditional probabilitiesP(b|a). If we assume that the substitution matrix entries are derived aslog-odds ratios, as in Chapter 2, then s(a, b) =log(P(a, b)/qaqb), which is thesame as log(P(b|a)/P(b)), so P(b|a) =qbes(a,b). We can in fact derive P(b|a)values from an arbitrary score matrix s(a, b) given background probabilities qa;see below.Given conditional probabilities P(b|a) we can generate pseudocounts as follows.Let fja be the maximum likelihood probabilities derived from the counts,

5.6 More on estimation of probabilitiesso fja =cja/a.cja.. Using these we set pseudocount values withαja =A fjbP(a|b),bwhere A is a positive constant comparable to the one we used with simple pseudo-

counts [Tatusov, Altschul & Koonin 1994; Claverie 1994; Henikoff & Henikoff1996]. We then use essentially the same equation as (5.3) to obtain the modelparameters:cja +αja

eMj (a) =..a.cja +αja.There is no obvious statistical interpretation for this type of pseudocount, butthe idea is quite natural: amino acid i contributes to pseudocount j in proportionto its abundance in the column and the probability of its changing to amino acidj. The formula interpolates between the treatment of pairwise alignments and themaximum likelihood solution. The substitution matrix term dominates if thereare small numbers of sequences (especially if A .1), and values close to themaximum likelihood estimate are obtained when the number of counts is large(more precisely when the total number of counts Cj .A).There are various choices for the scaling constant A of the pseudocounts. Forinstance A =1 was used in Lawrence et al. [1993], but this appears to be too weakin practice. Claverie [1994] suggests A =min(20, N ), and Henikoff & Henikoff[1996] suggest A =5R, where R is the number of different residue types observedin the column (i.e. the number of a for which cja >0).Deriving P(b|a) from an arbitrary matrixEven if a score matrix s(a, b) was not derived as a log-odds matrix, as long as certainconditions are fulfilled it is possible to find a scale factor λsuch that λs(a, b)will behave correctly when interpreted as a log-odds matrix [Altschul 1991]. Theconditions are that the matrix is negatively biased, i.e. ab qaqbs(a, b) <0, andthat it contains at least one positive entry.What we want is a set of values rij for which1 rabs(a, b) =log ,λqaqbwhere rab can be interpreted as the probability for the pair a, b. This equation iseasily inverted, so we get the pair probabilities expressed in terms of the substitutionmatrix rab =qaqb exp(λs(a, b)). To be legitimate probabilities the rab haveto sum to one. We therefore need to find a λsuch thatf (λ) =qaqbeλs(a,b) .1 =0. (5.4)a,bOne such value is λ=0, but clearly this is not what we want. The two conditions

5 Profile HMMs for sequence familieswe gave above turn out to be sufficient to ensure there is another, positive solutionto this equation; see the exercises below.The resulting value of λis called the natural scaling factor of the substitutionmatrix. This probabilistic interpretation of the substitution matrix leads to an en

tropy measure for the matrix of ab rab log(rab/qaqb), which is a useful quantityfor characterising and comparing substitution matrices [Altschul 1991].Exercises5.3Use the negative bias condition to show that f (λ) is negative for smallenough λ. Hint: calculate f (0), the derivative of f (λ)at λ=0.5.4Use the second condition, that there is at least one positive s(a, b), toshow that f (λ) becomes positive for large enough λ.5.5Finally, show that the second derivative of f (λ) is positive, and from thisand the results of the previous two exercises that there is one and onlyone positive value of λsatisfying (5.4).Estimation based on an ancestorThere is a more principled and direct way to use the information in substitutionmatrices for estimating the HMM probabilities than that described above. Thisapproach does not use pseudocounts. Instead, it assumes that all the observed sequenceshave been derived independently from a common ancestor, and generatesan estimate of the residue present in a given position in that common ancestor (orrather a posterior probability distribution for what that residue was). From thiswe can estimate the probability of seeing each residue in a new descendant of theancestor, different from those in the sample.k

Assume we have example sequences xk with residues xj in column j of thealignment (we have adjusted our notation slightly; this xjk is not the jth residuein sequence xk if there are gaps, but it is a convenient notation for what we needhere). Once again, we need the conditional probabilities P(b|a) derived from thesubstitution matrix. Let the residue in the common ancestor be yj . Then we canuse Bayes rule to calculate the posterior probability that yj =ak

qa kP(xj |a)

P(yj =a|alignment) = .. (5.5)ka.qa.kP(xj |a )Note that we needed a prior distribution for residues at the common ancestor,which we set to qa because that is our background probability for amino acids inthe absence of further information.We can now calculate the HMM emission probabilities as the predicted probabilitiesfor a new sequenceeMj (a) =P(a|a )P(yj =a |alignment). (5.6)a.

5.6 More on estimation of probabilitiesOne problem with this approach is that, as we noticed above, different columnsvary widely in their degree of conservation. Indeed, that is one of the propertiesthat we wanted to exploit when using alignments to estimate profile HMMs.However, using a single substitution matrix implies assuming a fixed degree ofconservation. As we discussed in Chapter 2, matrices typically come in familiesvarying in their level of implied conservation. Examples are the PAM [Dayhoff,Schwartz & Orcutt 1978] and the BLOSUM [Henikoff & Henikoff 1992] series ofmatrices. We can therefore significantly improve the approach in (5.5) and (5.6) ifwe optimise over choice of matrix from a family. This way, a very conserved columnmight use a conservative matrix, such as PAM30, and a very varied columnwould use a divergent matrix, such as PAM500.How do we choose the optimal matrix? A natural approach is to maximise thelikelihood of the observed data1 Nk

P(xj , ..., xj |t) =qa P(xj |a, t) (5.7)akwhere t is the matrix family parameter (t for evolutionary time). It would also bepossible to use a Bayesian approach here, proposing a prior distribution over t,then combining this with (5.7) in Bayes’ rule to obtain a posterior distribution fort, and summing over this in (5.6). However, that would require signficantly morecomputation.The maximum likelihood time-dependent approach is closely related to the‘evolutionary weights’ method in the PROFILE package [Gribskov & Veretnik1996]. However, that method estimates different evolutionary times t for eachpossible ancestral amino acid, and also adjusts the resulting weights with respectto a set of baseline probabilities; for details see Gribskov & Veretnik [1996].There are also strong connections between the methods of this subsection andthose discussed later in Chapter 8 when building phylogenetic trees using maximumlikelihood methods.Testing the pseudocount methodsAll the methods mentioned above have been tested in various ways. Direct tests,in which profiles were constructed and used for searching, were carried out extensivelyby Henikoff & Henikoff [1996]. The best method turned out to be thesubstitution matrix based method (5.6), with A =5R as described above; theDirichlet mixture regulariser came a reasonably close second. Other tests gavedifferent results [Tatusov, Altschul & Koonin 1994; Karplus 1995], so it is notclear which method is best, and it is likely that this will depend on the applicationand the details of the mixture components or substitution matrix used.An interesting method was for testing various regularisers was given byKarplus [1995]. Instead of performing a huge number of database searches, he

5 Profile HMMs for sequence familiesasked the following question:3 How well can an amino acid distribution be approximatedfrom a small sample? Columns were extracted from a large set ofdeep alignments (the BLOCKS database; Henikoff & Henikoff [1991]). Imaginewe take a small sample of size n with counts sa from a column with completecounts Ca. From the sample counts sa we can estimate the probabilities es (a)ofother symbols that might occur in the same column, using one of the methodsdescribed above (we use a subscript s to remind ourselves that this estimation isdependent on the sample counts). We can also estimate the probabilities of othersymbols directly from the frequencies with which they occur in all columns ofthe database together with the probability P(s|C) of drawing s from a column C(given by the multinomial distribution). This estimate is given by:columns CP(s|C)Ca

P(a|s) =.,columns CP(s|C)|C|

where |C|denotes the number of symbols in the column C. P(a|s) can onlybe calculated up to a sample size of n =5, but this is also the most interestingregime, because it is for small sample sizes that regularisation is most crucial.We can now use the relative entropy .P(a|s)log es (a) to compare the ‘ideal’aprobability P(a|s) with that given by the regulariser. Summing over all sampless of size n gives a measureEn =P(s) .P(a|s)log es (a) , (5.8)s,|s|=nawhere P(s) is the probability of drawing the sample s averaged over all columnsin the database. This can be calculated using P(s) =CP(s|C)|C|/C |C|.Karplus proposed that a good regulariser should minimise En. He showed thatseveral of the more complex regularisers described above resulted in estimatorsthat were very close to optimal, in the sense that En was very small up to n =5.Of course, we are ultimately interested in database searches, and it is not evidentthat the regulariser obtaining the lowest value of En will actually be best forsearching. It is likely that the typical similarities in the source alignment databaseare not the same as the ones that we will be searching for with our HMM.As well as evaluating methods, Karplus’ approach can also be used to set thefree parameters in the various methods described above, for example the totalnumber of pseudocounts A to use in (5.3). For any value of A we can calculateEn from our database of columns, either directly or by some sort of randomsampling, and in fact we can also calculate the gradient of the relative entropywith respect to A. We can therefore find the value of A that minimises this averagerelative entropy, using gradient descent methods [Press et al. 1992], or by3 This page has been rewritten for the second printing.

5.7 Optimal model constructionother optimisation methods. In principle this can be done for any sample size n,yielding parameters dependent on n.5.7 Optimal model constructionWhen we first discussed the parameterisation of profile HMMs, we pointed outthat as well as estimating the probability parameters, it is necessary to decidewhich columns of the alignment should be assigned to insert states, and which tomatch states. We call this process model construction. At the time we proposed asimple heuristic, but we can do better than that. There is an efficient dynamic programmingalgorithm which can find the column assignments that maximise theposterior probability of the model, at the same time as fitting optimal probabilityparameters.In the profile HMM formalism, it is assumed that an aligned column of symbolscorresponds either to emissions from the same match state or to emissionsfrom the same insert state. It therefore suffices to mark which columns comefrom match states to specify a profile HMM architecture and the state paths forall the sequences in the alignment, as shown in Figure 5.7. In a marked column,symbols are assigned to match states and gaps are assigned to delete states. Inan unmarked column, symbols are assigned to insert states and gaps are ignored.State transition and symbol emission counts are obtained from the state paths,and these counts can be used to estimate probability parameters by one of themethods in the previous section. In passing, we note that this model estimationprocedure implicitly assumes that the multiple alignment is correct, i.e. that theimplied state paths have probability one and all other state paths have probabilityzero, which is akin to a Viterbi assumption. The next chapter addresses issues ofsimultaneous alignment and model estimation.There are 2L combinations of markings for an alignment of L columns, andhence 2L different profile HMMs to choose from. There are at least three waysto determine the marking. In manual construction, the user marks alignmentcolumns by hand. This is perhaps the simplest way to allow users to manuallyspecify the model architecture to use for a given alignment. In heuristic construction,a rule is used to decide whether a column should be marked. For instance,a column might be marked when the proportion of gap symbols in it is below acertain threshold. In MAP construction, a maximum a posteriori choice is determinedby dynamic programming. A description of this algorithm follows.MAP match–insert assignmentThe MAP construction algorithm recursively calculates a number Sj , which is thelog probability of the optimal model for the alignment up to and including column

5 Profile HMMs for sequence families(a) Multiple alignment:EndBeginM M MI I I ID D Dxx.. .x(c) Observed emission/transition countsbat AG ---Cmodel positionrat A-AG -C0123cat AG-AAA -400gnat --AAAC match C -0 0 4emissionsG -030T -000goat AG ---C12. . . 3A 0060insert C 0000(b) Profile-HMM architecture: emissions G 0010T 0000M-M4 3 2 4M-D1 1 0 0M-I 0010state I-M 0020transitions I-D 0 0 1 0I-I 0040D-M-001D-D-100D-I -020000 1234Figure 5.7 As an example of model construction from an alignment, a smallDNA multiple alignment is given (a), with three columns marked above withx’s. These three columns are assigned to positions 1–3 in the model architecture(b). The assignment of columns to model positions determinesthe symbol emission and state transition counts (c) from which probabilityparameters would be estimated.j, assuming that column j is marked. Sj is calculated from smaller subalignmentsending at a marked column i (i <j) by incrementing Si with the summed logprobability of the transitions and emissions for the columns between i and j.Therelevant probability parameters are estimated ‘on the fly’ from the counts thatare implied by marking columns i and j while leaving unmarked the interveningcolumns (if any).Transition and emission counts for a section of alignment bounded by markedcolumns i and j are independent of how columns are marked before i and after j,thus making a dynamic programming recursion possible. Only marked columnsare considered in the recursion, because transition and emission counts for unmarkedcolumns are not independent of the assignment of neighbouring columns;a single insert state may account for more than one column in the alignment.For instance, let Tij be the summed log probability of all the state transitionsbetween marked columns i and j. We can determine Tij from the observed statetransition counts cxy and the probabilities axy:Tij =cxy log axy.x,y∈M,D,I

5.8 Weighting training sequencesTransition counts cxy are obtained from the partial state paths implied by markingi and j. For instance, if in one sequence we see a gap in column i, five residuesin columns i +1to j .1, and a residue in column j, we would count one delete–insert transition, four insert–insert transitions, and one insert–match transition.The transition probabilities axy are estimated from the cxy in the usual fashion,possibly including Dirichlet prior terms αxy (or indeed, any form of prior that isindependent of the marking outside of i, ..., j):cxy +αxy

axy =..y cxy +αxyLet Mj be the analogous log probability contribution for match state symbolemissions in column j, and Ii+1, j.1 be the same for the insert state emissions forcolumns i +1, ..., j .1 (for j .i >1). We can now give the algorithm:Algorithm: MAP model constructionInitialisation:S0 =0, ML+1 =0.Recurrence: for j =1, ..., L +1:Sj =max Si +Tij +Mj +Ii+1, j.1 +λ;0≤i<jσj =argmax Si +Tij +Mj +Ii+1, j.1 +λ.0≤i<jTraceback: From j =σL+1, while j >0:Mark column j as a match column;j =σj . .A profile HMM is then built from the marked alignment. The extra term λisa penalty used to favour models with fewer match states. In Bayesian terms, λisthe log of the prior probability of marking each column, implying a simple butadequate exponentially decreasing prior distribution over model lengths.With some care in implementation, this algorithm is O(L) in memory andO(L2) in time for an alignment of L columns.5.8 Weighting training sequencesOne issue that we have avoided completely so far is that of weighting sequenceswhen estimating parameters. In a typical alignment, there are often some sequencesthat are very closely related to each other. Intuitively, some of the informationfrom these sequences is shared, so we should not give them each thesame influence in the estimation process as a single sequence that is more highlydiverged from all the others. In the extreme that two sequences are identical, itmakes sense that they should each get half the weight of other sequences, so that

5 Profile HMMs for sequence families2t= 2674t= 83t= 5t= 36t= 35I2I 3I1V5II+12V6V7I II ++1 2 3I4t=2 11 234 Figure 5.8 On the left, a tree of sequences with branch lengths. On theright, the corresponding ‘current’ and ‘voltage’ values used in the ‘Kirch

hoff’s law’ approach to sequence weighting (see text).the net effect is of having only one of them. Statistically, the problem is that typicallythe examples we have do not constitute a good random sample from all thesequences that belong to the family; the assumption of independence is incorrect.To deal with this sort of situation, there have been a large number of proposals fordifferent ways to assign weights to sequences. In principle, any of these can beused in combination with any of the methods of the preceding sections on fittingmodel parameters and model construction.Simple weighting schemes derived from a treeMany weighting approaches are based on building a tree relating the sequences.Since sequences in a family are related by an evolutionary tree, a very natural approachis to try to reconstruct this tree and use it when estimating the independentcontribution of each of the observed sequences, downweighting sequences thathave only recently diverged. We discuss phylogenetic tree construction at lengthlater in Chapters 7 and 8, as well as in the next chapter on multiple sequencealignment. For our current purposes, the fine details of the method are probablynot too important, and we will assume that we are given a tree connecting thesequences, with branch lengths indicating the relative degrees of divergence foreach edge in the tree.One of the intuitively simplest weighting schemes [Thompson, Higgins & Gibson1994b] can be expressed nicely as follows. We are given a tree made of aconducting wire of constant thickness and apply a voltage V to the root. All theleaves are set to zero potential and the currents flowing from them are measuredand taken to be the weights. Clearly, the currents will be smaller in the highly dividedparts of the tree so these weights have the right qualitative properties. They

5.8 Weighting training sequencescan be calculated by applying Kirchhoff’s laws. For instance, in the tree shown inFigure 5.8, let the current and voltage at node n be In and Vn, respectively. Sinceconstant factors do not affect the calculation, we can set the resistance equal tothe edge-time. We then find V5 =2I1 =2I2, V6 =2I1 +3(I1 +I2) =5I3, andV7 =8I4 =5I3 +3(I1 +I2 +I3). There are therefore three equations relating thefour currents, and these give I1: I2: I3: I4 =20 : 20 : 32 : 47.Another attractively simple idea was proposed by Gerstein, Sonnhammer &Chothia [1994]. Their algorithm works up the tree from the leaves, incrementingthe weights. Initially the weight of a sequence is set equal to the edge-time ofthe edge immediately above it. Now, suppose node n has been reached. The edgeabove n has edge-time tn, and this is shared out amongst the weights of all thesequences at the leaves below n, incrementing them by a fraction proportional totheir current weight values. Formally, the increase wi in a weight wi is givenby wi =tn .wi . (5.9)wk

leaves k below nThe same operation is carried out up to the root.This is clearly an easy and efficient algorithm. For instance, the weights in thetree of Figure 5.8 are computed as follows: Initially the weights are set to theedge lengths of the leafs, w1 =w2 =2, w3 =5, and w4 =8. At node 5 the edgelength of 3 above node 5 is shared out equally to w1 and w2, giving them 3/2each, so now w1 =w2 =2 +3/2 =3.5. At node 6 we find the edge of length 3above node 6 is shared out to nodes 1, 2 and 3 in the ratio 3.5 : 3.5 : 5, makingw1 =w2 =3.5 +3 ×3.5/12, and w3 =5 +3 ×5/12. With w4 =8, this givesw1: w2: w3: w4 =35 : 35 : 50 : 64. Even though these weights are close to thosegiven by the Kirchhoff rule, the methods are in a sense opposed, for in a tree withtwo leaves and one edge longer than the other, the longer edge is down weightedby Kirchhoff and up weighted by (5.9).Root weights from Gaussian parametersOne view of weights is that they should represent the influence of leaves on theroot distribution. It is possible to make this idea precise, as Altschul, Carroll &Lipman [1989] showed. They built on the version of Felesenstein’s ‘pruning’ algorithmwhich applies to continuous parameters [Felsenstein 1973]. Instead ofdiscrete members of an alphabet we have a continuous real-valued variable, likethe weight of an organism. In place of a substitution matrix we have a probabilitydensity that defines the probability of substituting one value, x, of this variableby another, y. A simple example of such a density is a Gaussian, where the probabilityof x →y along an edge with time t is exp(.(x .y)2/(2σ2t). The pruning

5 Profile HMMs for sequence familiest 3t 2t1x1x2 x3Figure 5.9 The tree described in the text when deriving Gaussian weights.algorithm now proceeds exactly as for a finite alphabet, but with integrals replacingdiscrete sums [Felsenstein 1973].4Felsenstein’s algorithm yields a Gaussian distribution for the parameter inquestion at the root whose mean μdepends linearly on the values xi of the param

eters at the leaves, so μ=wi xi . Altschul, Carroll & Lipman [1989] proposedthat these wi should be used as weights. They represent the influence of each leafat the root.Example: Altschul–Carroll–Lipman weights for a three-leaf treeTo illustrate how the weights are derived, consider the simple three-leaf treeshown in Figure 5.9, where leaf i takes the value xi . The probability distributionat node 4 is given by.(x.x1)2 .(x.x2)2P(x at node 4 |L1, L2) =K1e e

2t12t2where K1 is a normalising constant. One can rewrite this as.(x.v1 x1.v2 x2)2P(x at node 4 |L1, L2) =K1e 2t12where v1 =t2/(t1 +t2), v2 =t1/(t1 +t2) and t12 =t1t2/(t1 +t2). If we were consideringonly the two-leaf tree with root at node 4, the mean of the root distributionwould be given by μ=v1x1 +v2x2, and the weights would be v1 and v2. Continuingwith our three-leaf tree, however, we find next that the distribution at node 54Historically, the continuous case came first, and Felsenstein defined the pruning algorithmfor Gaussian distributions of real-valued parameters. In the cited paper he takes account ofthe distribution of the parameters at each leaf, e.g. the mean and variance of the weight ofan organism. Puzzlingly, he also introduces covariances between values for different leaves.It is not clear how to calculate a covariance between, say, the weights of cows and cats. Forproteins, having multiple corresponding sites in an alignment would allow correlations to beconsidered in principle.

5.8 Weighting training sequencesis given by.(y.x3)2.(x.v1 x1.v2x2)2 .(x.y)2P(y at node 5 |L1, L2, L3) =K2e 2t3e 2t12 e 2t4 dxwhere K2 is a normalising constant, and the integral is taken over all possiblevalues of x at node 4 (and is the exact equivalent of the sum over all possibleancestral assignments of residues in the case of a discrete alphabet). This is astandard Gaussian integral, and boils down to the following.(y.w1 x1.w2x2.w3 x3)2P(y at node 5 |L1, L2, L3) =K3e 2t123where K3 is a new normalising constant and t123 =t3{t1t2 +t4(t1 +t2)}/ , with.=t1t2 +(t3 +t4)(t1 +t2). The mean of the distribution of y, i.e. of the rootdistribution, is given byμ=w1x1 +w2x2 +w3x3with w1 =t2t3/ , w2 =t1t3/ , and w3 ={t1t2 +t4(t1 +t2)}/ . These are thereforethe Altschul–Carroll–Lipman weights for a tree with three leaves.Voronoi weightsThere are also weighting schemes not based on trees. One approach is based onan image of the sequences from a family lying in ‘sequence space’. In general,some will lie in clusters and others will be widely separated. The philosophy ofthe Voronoi scheme [Sibbald & Argos 1990] is to assume that this unevennessrepresents effects of sampling, including the ‘sampling’ performed by naturalselection in favouring certain phyla. A more thorough trawl through all eligiblesequences of the protein family, or perhaps a multitude of reruns of evolution,should produce a flat distribution within some region. To compensate for the gaps,we want to give sequences a weight proportional to the volume of empty spacearound them.If sequence space were two-dimensional, or even low-dimensional, we coulduse standard methods from computational geometry to divide up space into regionsaround each example point. The standard approach is to take lines joiningneighbouring pairs of points and draw their perpendicular bisectors, extendingthem till they join up. This produces a partitioning into polygons (in two dimensions)called a Voronoi diagram [Preparata & Shamos 1985], which has theproperty that the polygon around each point is the set of all points closer to thatpoint than any other.Sequence space is of course a high-dimensional construct in which the Voronoigeometry is hard to picture or calculate. However, we can implement the underlyingprinciple of it by sampling sequences randomly from sequence space andtesting to see which of the family sequences each sequence lies closest to. The

5 Profile HMMs for sequence familiestrick is in the sampling. This is accomplished by choosing, at each position ofthe alignment, uniformly from those residues which occur at that position in anysequence. If we count ni such sample sequences closest to the ith family member(dividing up the counts if there is a tie), then we can define the ith weight to beni /k nk .Maximum discrimination weightsAnother approach to weighting comes indirectly, from focusing initially on a reformulationof the primary goal in building the model [Eddy, Mitchison & Durbin1995]. Rather than maximising the likelihood of sequences in the family, or eventheir posterior probability derived from Bayesian priors, we are normally interestedin making the correct decision on whether sequences are members of thefamily or not. We are therefore interested in the probabilityP(x|M)P(M)

P(M|x) =,

P(x|M)P(M) +P(x|R)(1 .P(M))where x is a sequence from the family, M is the model for the family that weare fitting, R is our alternative, random model for sequences not in the family,and P(M) is the prior probability of a new sequence belonging to the family.Given example training sequences xk , we would like to maximise the probabilityof classifying them all correctly, which isD =P(M|xk ),knot P(xk |M) as usual with maximum likelihood based approaches. We call Dthe discrimination of the model on the set of sequences xk . Maximising D willhave the effect of emphasising performance on distant or difficult members of thefamily. Sequences that are easily classified will have P(M|x) values very closeto one; changing parameters to increase their likelihood P(x|M) will have verylittle effect on D. On the other hand, increasing the likelihood of sequences forwhich P(M|x) is small can potentially have a big effect.It turns out that the parameter values that maximise D can be shown to be theones that maximise a weighted version of the likelihood, where the weights areproportional to 1 .P(M|xi ), i.e. the probability of misclassifying sequence i.This can be seen from the observation that if y =ex /(K +ex ), then.log yK==(1 .y).

.xK +exP(x|M)

If we set x =log , which is the log likelihood ratio for sequence x, thenP(x|R)y =P(M|x). So at a maximum of log D we will also be at a maximum of theweighted sum of log likelihood ratios, with weights 1 .P(M|xi ), and since the

5.8 Weighting training sequencesrandom model is fixed this is equivalent to a maximum of the weighted log likelihoodof the model M. The maximum discrimination criterion therefore amountsto another sequence weighting system.One difference from previous systems, however, is that these weights are definedin a somewhat circular fashion; they depend upon the model that is beingfit. When using maximum discrimination weighting as a method, an iterative approachmust be used; an initial set of weights gives rise to a model, from whichposterior probabilities P(M|x) can be calculated, giving rise to new weights, andhence a new model, and so on until convergence is achieved. This iterative re-

estimation procedure is analogous to the versions of the EM algorithm used to fitHMM parameters to sets of unlabelled sequences (p. 64 and p. 324).Maximum discrimination training has a big advantage in that it is directly optimisingperformance on the type of operation that the model will be used for,ensuring that the most effort is applied to recognising the most distant sequences.On the other hand, exactly the same point can lead to problems. If there is anytraining sequence that has been misclassified, then the distortion needed to give ita good score can damage performance for correct members of the class. To someextent, though, this same problem occurs with all weighting schemes: incorrectlyassigned sequences will be the most distant ones in any tree that gets built fromthe examples.Maximum entropy weightsFinally, we describe two weighting methods based on the idea of trying to makethe statistical spread of the model as broad as possible.Assume column i of a multiple alignment has kia residues of type a and atotal of mi different types of residues. To make a distribution as uniform as possiblefrom these counts by weighting each sequence, we can choose a weightfor sequence k of 1/(mikixk ). Maximum likelihood estimation will then yield aidistribution pia =kia/(mi kia) =1/mi , i.e. all the residues appearing in the columnwill have the same probability. To illustrate the idea, suppose we have tensequences with residue Aat a site, and one sequence with a B, so the unweightedfrequencies of A and B are cA =10 =1 . The weights of the ten sequences11, cB 11are w1 =w2 =...=w10 =1/(2 ×10) =0.05, and w11 =1/(2 ×1) =0.5, whichhave the effect of making the overall weighting for each of Aand Bequal.The preceding paragraph only considered one column. With just one weightper sequence, it is of course not possible to make the distribution uniform for allcolumns in an alignment. However, by averaging over all columns, one may hopeto obtain reasonable weights. That is, the weights are calculated as1wk =,mikixkii

5 Profile HMMs for sequence familiesand then normalised to sum to one. This weighting scheme was proposed by[Henikoff & Henikoff 1994].Instead of averaging, there is another approach to combining the informationfrom the different columns that has a simple theoretical justification. A standardmeasure of the ‘uniformity’ of a distribution is the entropy (11.8), which is largerthe more uniform the distribution is. Indeed, it is easy to see that the weights chosenabove based on a single column maximise the entropy of the distribution piafor that column. An HMM defines a probability distribution over sequences, andtherefore a natural extension of the single column weighting to full sequences isto maximise the entropy of the complete HMM distribution [Krogh & Mitchison1995]. We will see that, perhaps surprisingly, this is closely related to maximumdiscrimination weighting.Let us consider all the sites in an alignment with no gaps. We then sum theentropies from each site, and choose the weights to maximise this sum; that iswe maximise i Hi (w.) +λk wk , where Hi (w.) =.pia log pia, and pia

ais the weighted frequency of residue a at the ith site, computed as above.Suppose for instance that we have the sequences x1 =AFA, x2 =AAC, andx3 =DAC. Giving them weights w1, w2 and w3, respectively, the entropies ateach site areH1(w.) =.(w1 +w2)log(w1 +w2) .w3 log w3,H2(w.) =.w1 log w1 .(w2 +w3)log(w2 +w3),H3(w.) =.w1 log w1 .(w2 +w3)log(w2 +w3).We assume that the weights sum to one, and therefore we have to use a Lagrangemultiplier term λk wk , when differentiating and finding the maximum of theentropy. Setting the derivatives of H1(w.) +H2(w.) +H3(w.) +λk wk to zero2

gives (w1 +w2)w=(w1 +w2)(w2 +w3)2 =w3(w2 +w3)2, which implies w1 =1w3 =0.5, w2 =0. This makes the frequencies in each column equal, which wasour goal. If it seems odd to give a sequence zero weight, note that the residue at2

each site in x2 is always present in one of the other two sequences. Intuitively, x1 lies ‘between’ x1 and x3, (in fact, it would be a possible ancestral sequence of xand x3 in an evolutionary reconstruction based on parsimony; see Chapter 7).Another way to view the result of this example is that if we set the model probabilitiesto be the weighted counts frequencies, as a weighted maximum likelihoodprocedure would, the resulting model assigns an equal probability to all 1 of the original sequences, x, x2 and x3. This seems very reasonable, accordingto the view that all the example sequences should be treated as equally goodmembers of the family for which we are building the model. In fact, Krogh &Mitchison [1995] show that the maximum entropy procedure assigns weights tothe example sequences so that some subset of the sequences (perhaps all of them)have non-zero weight and equal probabilities under the resulting model, or they

5.9 Further readinghave a higher probability, in which case they have zero weight. The former canbe thought of as boundary points for the region of sequence space occupied bythe whole sequence set, while the latter are internal points.Furthermore, empirical tests indicate that the maximum entropy weights areoptimal in the sense that they maximise the minimum score assigned to anyof the example sequences [Krogh & Mitchison 1995]. This is an absolute versionof the criterion specified in the previous section on maximum discriminationweights; rather than simply weighting the weakest match most strongly, all theparameter-fitting effort is applied to increasing its score, until it reaches that ofthe other non-zero-weighted sequences. Although satisfying an attractive goal,maximum entropy weighting suffers from the same problems as maximum discrimination:if a sequence is an outlier that should not be a full member of thefamily, the method will force it in, possibly at a substantial cost in performanceon all other sequences. In addition, the rejection of all information from some ofthe sequences may seem intuitively undesirable.Exercise5.6Compute the weights for the following sequence set, using each of theweighting methods described above except Voronoi weights (which requiresrandom sampling of sequences): AGAA, CCTC, AGTC.5.9 Further readingPSSM methods were introduced during the 1980s for finding new members ofsequence families, although the matrix values were not always obtained usingan explicit probability-based derivation. They are also known by other names,such as weight matrices [Staden 1988]. More recent papers using related methodsinclude those by Stormo [1990]; Henikoff & Henikoff [1994]; Tatusov, Altschul& Koonin [1994].The non-probabilistic versions of profiles already have a long history, andmany variants of the profile method have been suggested and tested. Thompson,Higgins & Gibson [1994b] and Luthy, Xenarios & Bucher [1994] report an improvementwhen the sequences are weighted using one of the BLOSUM matrices[Henikoff & Henikoff 1992] instead of a PAM matrix. In Thompson, Higgins &Gibson [1994b] the treatment of gaps is also improved.Several ways have been suggested for incorporating structural information intoprofiles. In Luthy, McLachlan & Eisenberg [1991] substitution matrices were estimatedfor six different structural environments: the three secondary structureelements α-helix, β-sheet, and ‘other’ combined with an outside/inside classification,which was based on the exposure of an amino acid to solvent. Other 5 Profile HMMs for sequence familiesvariations of structural profiles can be found in Bowie, Luthy & Eisenberg [1991];Wilmanns & Eisenberg [1993].Early on, profile HMMs were adopted by Baldi et al. [1994], who used them tomodel globins, immunoglobulins and kinases. In this work a different estimationmethod was also introduced, which was based on gradient descent, see also Baldi& Chauvin [1994]. The same basic structure of profile HMMs has since been usedin several different areas. A library of HMMs for all the big protein families hasbeen established under the name of PFAM [Sonnhammer, Eddy & Durbin 1997].The library of regular expressions called PROSITE [Bairoch, Bucher & Hofmann1997] is being extended to something essentially like profile HMMs [Bucher etal. 1996]. Profile HMMs also have several uses for DNA. For instance they canbe used to find DNA repeat family members in large-scale genomic sequence.