Statistical Alignment

Exercise Sheet (J.Hein, A.Novak, I.Miklos, R.Lyngsø)

February 18, 2008

The traditional score based approach to alignment aims to find the one best alignment of a set of sequences, which is then used as the 'true' alignment. Statistical alignment fundamentally breaks with this view. It is based on stochastic models of sequence evolution. In general, this means that every alignment is possible. However, depending on parameters there may be a vast difference in how probable two different alignments are. In statistical alignment, the score of an alignment is the probability of obtaining the alignment under the evolutionary model, so rather than just providing a means for sorting alignments to be able to nominate one as the best, the scores of statistical alignment defines a distribution over plausible alignments. This has several advantages, among them quantifying the uncertainty of alignments and allowing joint modelling and inference of alignment, alignment parameters and phylogeny in a sound probabilistic framework.

In this tutorial you will get a brief introduction to statistical alignment with a newly developed tool based on *Markov Chain Monte Carlo* sampling. This methodology traverses the space of alignments, by repeatedly changing the current alignment in a sequence of so-called steps. Whenever a step is proposed it is either accepted or rejected depending on how the alignment probability changes – steps leading to a better alignment are always accepted while steps leading to a worse alignment may be accepted. Despite the local nature of the traversal, with two consecutive alignments usually being quite similar, MCMC sampling is guaranteed to eventually generate a true sampling according to the probability distribution on alignments. We will look at how to use the tool to sample alignments, look into statistics of the sampling, and analyse the samples produced. Time permitting, we will take a brief look at how the results compare to a traditional score based alignment.

Exercise I. (MCMC Sampling of Alignments)

In the first exercise we will look at the sampling part of MCMC based statistical alignment. This is computationally the most time consuming part of the process, so at times you may want to proceed to the following parts of the exercises while you wait for the samplings to complete.

1. First obtain the StatAlPrograms.tar.gz from the course website and unpack it with the command

   tar -xvzf StatAlPrograms.tar.gz

2. The first two files we will need are StatAlign.jar, the statistical alignment software, and glycosyl_hydrolase_family_22_lysozyme.fasta, a data set of twelve lysozyme protein sequences. This data set was retrieved from the HOMSTRAD data base at `http://www-cryst.bioc.cam.ac.uk/cgi-bin/homstrad/showpage.cgi?family=ghf22&disp=str`. The data does come in the form of a structurally curated alignment. However, this is not used by the statistical alignment program as it starts from scratch (and is looking for a homology rather than structural alignment anyway). Start the statistical alignment program with the command

   java -jar StatAlign.jar > out.txt

   Load the lysozyme data set by choosing *Add sequence(s)* in the *File* menu. Just starting the sampling with default parameters will probably mean that the sampling will outlast this tutorial. So start by choosing *Settings* under *MCMC* and set *Burn-in cycles* (number of steps before we start storing samples) to 100, *Cycles after burn-in* (number of steps where we do store samples) to 1000, and *Sampling rate* (number of steps between consecutive samples stored) to 10. Why don't we store all samples? Select *Run* to start the sampling. While the sampling is proceeding you can choose *Pause* and *Stop*; when it finishes these will become inactive and *Run* will again become active, and the last line in the out.txt file while read*Ready.* How long does it take to sample these 1,100 steps?

3. The MCMC settings used above are too small to expect the sample to give a proper reflection of the distribution over alignments. So if you want to create your own big sample, now is a good time to start it. To avoid this set of samples overwriting, or being overwritten by,

other sample sets, use the following commands to create the sample in a separate directory called longrun:

mkdir longrun

cp glycosyl_hydrolase_family_22_lysozyme.fasta longrun

cd longrun

java -jar ../StatAlign.jar > out.txt

Choose settings such that the run will finish before the end of this tutorial.

4. The reason for the burn-in cycles is that we will usually not start from a random sample from the distribution over alignments. So the burn-in stage is to allow the MCMC process to converge to this distribution. There is no sure-fire way of saying when the process has converged. However, one good indicator is the probability of the alignments sampled. Usually this will initially be increasing until it reaches a plateau it fluctuates around. So as long as the alignment probability shows an increasing trend, we should not be storing samples. To check whether your run had reached convergence before you started storing samples, run the following commands:

   python get_lh.py out.txt > lh.dat

   gnuplot plot_lh.gp

   As the increase in probability usually slows down gradually, the probabilities are plotted against the logarithm of the sample index, so the last sample of the burn-in stage should be plotted at 2 on the x-axis. Was it safe to start storing samples at this early point?

5. A prepared set of samples from a longer run (5,000 burn-in cycles, 40,000 cycles after burn-in, and a sampling rate of 200) is available in the files prepared_alignments.txt and prepared_out.txt. Is it safe to use this set of samples?

6. As mentioned at the beginning, each proposed step in the MCMC process can either be accepted or rejected. A reasonably high level of acceptance is desirable for exploring the full space of alignments. Determine how many alignment change steps that were accepted, respectively rejected, with the following commands:

   grep Alignment: prepared_out.txt | grep accepted | wc

grep Alignment: prepared_out.txt | grep rejected | wc

How about the phylogeny change steps (*Edge:* for branch lengths and *Topology:* for structure)? You can also check whether there is a difference between the acceptance rate for the first 1,000 steps and the last 1,000 steps with the following commands:

grep Alignment: prepared_out.txt | head -1000 | grep accepted | wc

grep Alignment: prepared_out.txt | tail -1000 | grep accepted | wc

7. To get an idea of how the time it takes to sample alignments depends on the data set size, create data sets with 3, respectively 6, of the sequences in the original data set and check how long it takes to create samples from these. You may want to run the statistical alignment just with

   java -jar StatAlign.jar

   This way, the output is not written to the file out.txt but to the window you are running the program in, making it easier to spot when the *Ready.* appears. Does the time depend on sequence similarity? Also try deleting the last half of the positions in the 12 sequence data set and see how that affects sampling time. You can do this with the command

   python half_sequence.py glycosyl_hydrolase_family_22_lysozyme.fasta > half_ghf22l.fasta

Exercise II. (Analysis of Sampled Alignments)

So far we have created data files with sets of sampled alignments. Though they do represent the distribution over alignments, they are not very informative without post-processing to extract useful summaries.

1. Despite the hallmark of statistical alignment being not to aim to find the one true alignment, in many situations it is still useful to have an alignment representing the inferred homology between positions in sequences. Each alignment visited in the MCMC sampling process has an associated probability, and by remembering the one with highest

probability we could produce the alignment most likely to be 100% correct among the alignments visited. However, often it as is preferable to find an alignment likely to have as many positions correct as one that is most likely to be 100% correct; in particular as even the alignment most likely to be 100% correct has a vanishingly small probability of this. The *maximum posterior decoding* (MPD) method achieves this by essentially computing the alignment where the sum of the probabilities of the alignment columns (i.e. the frequency with which they are seen in the sample) is maximised. Compute the MPD alignment from your sample with

java -jar MPDAlignment.jar alignments.txt

How does the alignment compare with the original alignment from the HOMSTRAD data base? Beneath the alignment the probability of each column in the alignment is listed. Is there a correlation between low probability columns and disagreements with the HOMSTRAD alignment? Is there a correlation between column probability and variation within the column? How do gaps influence column probability?

2. Alignment column probabilities can generally be used to assess the detailed variation in the certainty of the homology statements of an alignment. We will explore that using another program based on the statistical alignment methodology, but not applying the MCMC approach. Compile and run this program with the following commands:

   tar -xvzf StructureDecoder.tar.gz

   cp Data/blosum62.bla .

   javac -O *.java

   java Main

   For pairwise alignment it is feasible to compute exact column probabilities rather than approximate them from a set of sampled alignments. This program uses column probabilities in pairwise alignments with all close relatives in a data set to assess how well defined amino acid homology is through a particular sequence. For each column in the most probable alignment, the probability of that column occurring summed over all possible alignments is computed. For each position this is averaged over all close relatives and plotted.

You may run into complications using the lysozyme data set with this program. If this is the case, use the pfcp.fsa file (papain family cystein proteinase, again retrieved from the HOMSTRAD data base) in the Data subdirectory unpacked from the StructureDecoder.tar.gz archive. Load it by choosing *Open* from the *File* menu and choose *Settings...* from the *Run* menu. Set the *Minimum similarity* value to 300 to start with (this determines what is considered a close relative), the *Max number of steps in maximizing (10-200)* to 10, and *Precision (decimal places)* to 3. Now choose *Preprocess...* from the *Run* menu. Select a sequence with many close relatives (the number in parenthesis) and choose *Run...* from the *Run* menu. When the analysis finishes, click *OK* to see the results. Repeat with a sequence with few close relatives. Are there systematic differences between the plots? Try lowering the *Minimum similarity* value. How does this affect the two plots? Try to find the family in the HOMSTRAD data base. This provides information about structure conservation throughout the sequence. There is also a snapshot of the three dimensional structure in the pfcp.jpg file. Are there correlations between the homology plots and structure variation?

3. The model behind the statistical alignment program not only defines a distribution over alignments, but a joint distribution over alignments, phylogenies and parameters. The probability of an alignment depends not only on the alignment but also on the phylogeny assumed to relate the sequences as well as model parameters. So the method also allows us to explore uncertainties about these entities. The file glycosyl_hydrolase_family_22_lysozyme.tree.nexus contains 1,000 trees sampled from a long MCMC run on the lysozyme sequences. We will use the program SplitsTree4 to analyse this set.

   If SplitsTree4 is not installed, you will need to go to `www-ab.informatik.uni-tuebingen.de/software/splitstree4/welcome.html`, register for a license, and download the installation.sh file (for Unix). Running the command

   installation.sh

   should install SplitsTree4 with the executable SplitsTree file in the splitstree4 directory. Go to this directory and start the SplitsTree4 program with the command

   SplitsTree

Load the glycosyl_hydrolase_family_22_lysozyme.tree.nexus file by choosing *Open* in the *File menu*, and choose *ConsensusNetwork* in the *Networks* menu. With the default threshold of 0.33 the sampled tree topologies do not exhibit any uncertainties. Is this expected given our previous analyses? Try lowering the threshold to 0.2. What changes? What does the change signify? Can you find evidence for this uncertainty in the sequence data? What is the meaning of the threshold value? What happens when you lower the threshold even more?

Exercise III. (Comparison to Score Based Alignment)

In the final exercise we will reanalyse our data set with a commonly used score based alignment program.

1. Start the ClustalX program by the following commands:

   cd clustal

   clustalx

   Load the lysozyme data set by choosing *Load Sequences* in the *File* menu. In the *Alignment* menu, choose *Multiple Alignment Parameters* under *Alignment Parameters* and change the Protein Weight Matrix to *BLOSUM series*. What should the *Gap Opening* and *Gap Extension* parameters be set to? Experiment with a few different values and see how it affects the alignment.

2. An alignment is constructed by choosing *Do Complete Alignment* in the *Alignment* menu. Once you have found an alignment you are happy with, compare it to the MPD alignment we constructed with the statistical alignment programs. Are there significant differences? What does the histogram in the bottom frame of the window indicate? How does it correlate with the secondary structure annotation of the HOMSTRAD data base? How does it compare to the column probabilities of the MPD alignment?

3. ClustalX uses a guide tree to build a multiple alignment, that is determined from pairwise alignments of the sequences in the data set. How does this correlate with the consensus tree we found using SplitsTree4 (before lowering the threshold)? Can you recreate the uncertainty

found with SplitsTree4 on the sampled trees (experiment with *Exclude Positions with Gaps*, *Correct for Multiple Substitutions* and *Clustering Algorithm* in the *Trees* menu)? Is this surprising?