

Pattern Search in a Single Genome

1.7.09

Background and Motivation. The haploid human genome is 23 pieces of text of total length $3 \cdot 10^9$. (the fact that the text is spread on 23 pieces is a detail that can be ignore for now) It has a lot of structure in the sense that it has protein genes, RNA genes, regulatory signals in it. Additionally, it has structural structure related to packing into chromosomes, duplication and more. This complexity makes it a challenging task to analyse by computational and statistical methods. The comparison of different related genomes is a major contributor to the understanding of genomes.

Search for single patterns is still important and challenging. Signal search can be split into cases: Is the signal known? Is it partially known? Do we accept error? If we accept error, then what kind of error? Substitutions or also insertion-deletions of single letters? If we don't know it, we are searching for common instances of an unknown signal and the same questions pertain concerning errors. Searching for patterns can be complicated by the structure of the human genome since it makes it relevant to search conditionally on features in the human genome, such as "500 base pairs before a protein gene" or similar criteria. We could also have a battery of signals we are searching for simultaneously and it could be that the search could be rationalized relative to first searching for one signal, then the second etc. . To all these search scenarios are additionally added the question of statistical significance: is the observation (found configurations of signals) statistically significant? It could it as well have been observed in a random string. Thus we now need to define a random string. This can sound easy – just flip a four sided (A,C,G,T) coin $3 \cdot 10^9$ times. This is good for many purposes, but can also fail seriously due to the inherent structure of the human genome. For instance CG rich regions are present in the genome and the naïve model of the genome, would make "CGCGACGCGC" have a lower probability than a model that incorporates regions of CG richness in the random model of the human genome.



Above: a single signal (red) is to be search for in one sequence (blue) that contains a single instance of it. More instance could be envisaged. How is this done most efficiently?

Below: In general four cases can be described: i. we look for a perfect match. ii. we allow errors due to substitutions iii. We allow errors due to substitutions and insertion-deletions. Iv. We rank the possible matchings according to a weight function and keep matches above a certain threshold or the best match. $p1$ and $p2$ are here the two patterns of length 5, W the weight of the complete patterns defined via a nucleotide-nucleotide weight function $w(\cdot)$.

CGTA	CGTA	C-GTA	CTGTA	$W(p1,p2) = \sum_{i=1}^5 w(p1[i],p2[i])$
CGTA	CGGA	CCGGA	CCGGA	

Searching for a single signal without error. The simplest way to do this is to scan the sequence starting at all conceivable positions and then scan both sequence and pattern searching for mismatches. If there is no mismatch a match has been found. Algorithms faster than this has been devised (Knuth et al., 1973; Boyer et al.,1974).

Searching for a single signal allowing substitution errors. Again we can start at each position in the sequence and search for mismatches with the pattern, but this time only discarding a position as a math if the number of mismatches exceeds the number allowed. This is actually into the realm of local alignment (Smith and Waterman, 1981), the problem of trying to identify regions in two sequences that are similar. Here we would require the region of the pattern to be the entire pattern - which is also sometimes called glocal alignment - and furthermore disallow insertions and deletions when comparing the regions, a feature usually allowed for alignment. With mismatches we can no longer use the fast methods for searching without errors, as they rely on knowing exactly what we have seen so far when the first mismatch occurs. Still, faster methods than the naive approach are possible (Chan et al., 2006).

Searching for a single signal allowing substitution and insertion-deletions errors. This is a special case of local alignment (Smith and Waterman, 1981). (Global) Alignment finds the best way of putting "-" into two strings so they become equally long and many identical nucleotides are matches in the first sequence with nucleotides in the second sequence. Local alignment solves the same problem, but now we are looking for sub-segments in the two sequences that align well. Signal search with substitutions and insertion-deletions is

a special case of local alignment because now we allow a segment in the second sequence, but the segment in the first sequence must be the complete sequence.

Searching for signals based on nucleotide-nucleotide matrix. This is done via a position specific weight matrix (PWM) – (Wasserman and Sandelin, 2004). The PWM is made by having observed a series of previous instances of the pattern. The frequency of each nucleotide is calculated in each position in the observed pattern. The frequency of nucleotides in a background model (not pattern) is also given. A score is given to how well the pattern at hand fits the pattern distribution relative to the background distribution. This is in principle a multiplicative score, but is made additive by taking the logarithm. A nucleotide N would score $\log[f_i(N)/f(N)]$, where $f_i(N)$ is the probability of seeing N at the i 'th position of the pattern and $f(N)$ is probability of seeing N in the background distribution. Databases with PWM corresponding to different classes of patterns are available (for instance JASPAR and TRANSFAC).

Searching for common unknown exact signals. One clever way to approach would be via suffix trees, that is a compact way to represent all segments of a sequence in a compact way (McCreight, 1976). More compact generalisations have been found.

The above problems vary much in computational hardness and how well defined they are. They are all relevant for analyzing the human genome.

Describing context dependent conditions and evaluating statistical significance using stochastic null models for genomes is often done via methods used to *annotate* genomes. Instead of modelling the probability of a genome directly, it is done via hidden states describing the properties of interest (CG-islands, protein coding, RNA folding, signals,..). Given the hidden states the nucleotides of the genomes are emitted probabilistically. The favourite models for the hidden states are stochastic grammars (Durbin et al., 1998 chapt. 9)

Plan:

Read the first seven chapters of the Python tutorial available in the documentation section at www.python.org. While you are reading the tutorial, start working on a program that determines whether a pattern occurs in a string by naively checking for occurrence at each position in the string.

Get hold of chromosome 21 of the human genome, e.g. from genome.ucsc.edu. A common promoter signal is the TATA box that will most often consist of the string TATAAA. How many times does this string occur in chromosome 21? You may want to start just looking for it in a small part of chromosome 21 to get an idea of how long time it will take to search the entire chromosome.

Write a program that uses the faster method of Knuth et al. or Boyer et al. How much faster is it to search for TATA boxes with this program? Access the one of the promoter data bases (TRANSFAC or JASPAR). Can you transform a position weight matrix to its most probable string? Search for the most probable string for a couple of promoters. Write a program that for each position computes the probability of obtaining the substring starting at that position from a PWM. What is a high probability? How many patterns with high probability occur in chromosome 21?

Read Lawrence et al., 1993. Try to modify the method to find N occurrences in a single string (rather than 1 occurrence in each of N strings), and write a program that does this. Can you find long patterns that occur many times in chromosome 21? You may also be able to get data from some of the other students working in our group this summer that allows you to search for one pattern repeated in a number of sequences.

Comments. If the above is done, one could pursue: Access promoter database at TRANSFAC or JASPAR. Search the human genome for occurrences of 10 of these promoters. Plot these relative to the start of a gene just following them. Find the longest string occurring k times on the human genome. $k=2, \dots, 100$. Plot the length(k). Find the longest string occurring k times in the human genome allowing for e errors. Plot the length(k, e).

References:

- Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990). "Basic local alignment search tool". *J Mol Biol* **215** (3): 403–410.
 - Boyer and Moore, *A fast string searching algorithm*, Carom. ACM 20, (10), 262–272(1977)
 - Burge and Karlin (1997) Prediction of complete gene structures in human genomic DNA. *J. Mol. Biol.* 268, 78-94.
 - Ho-Leung Chan, Tak-Wah Lam, Wing-Kin Sung, Siu-Lung Tam and Swee-Seong Wong, Compressed Indexes for Approximate String Matching ESA 2006.
 - CHURCHILL, G. A. 1989. Stochastic models for heterogenous DNA sequences. *Bull. Math. Biol.* 51:79-94.
 - Durbin et al. (1998) *Biological Sequence Analysis* CUP
 - Gusfield (1998) *Strings, Trees, and Sequences: Computer Science and Computational Biology* CUP
 - Knuth, Morris & Pratt (1977). "Fast pattern matching in strings". *SIAM Journal on Computing* 6 (2): 323–350.
 - Lawrence (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science* 262(5131):208-14.
 - McCreight (1976). "A Space-Economical Suffix Tree Construction Algorithm". *Journal of the ACM* **23** (2): 262–2
 - Smith TF, Waterman MS (1981). "Identification of Common Molecular Subsequences". *Journal of Molecular Biology* 147: 195–197
 - Wasserman & Sandelin (2004) Applied bioinformatics for the identification of regulatory signals *Nature Reviews Genetics* 5, 276-287
 - Wikipedia: Sequence motif, Position-specific scoring matrix, Approximate string matching, BLAST, Knuth–Morris–Pratt algorithm, String searching algorithm
- <http://jaspar.genereg.net/>