Motif finding

This exposition was developed by Knut Reinert and Clemens Gröpl. It is based on the following sources, which are all recommended reading:

1. Buhler, Tompa: Finding Motifs using Random Projections, RECOMB 2001, 69-75
2. Price, Ramabhadran, Pevzner: Finding subtle motifs by branching from sample strings, Bioinformatics, 2003, 149-155
3. Durbin, Eddy, Krogh, Mitchison: Biological sequence analysis, Probabilistic models of proteins and nucleic acids, pages 323f
4. Bailey, Elkan: The value of prior knowledge in discovering motifs in MEME, Proc. Int. Conf. Intell. Syst. Mol. Biol., 1995, 21ff
5. Mount: Bioinformatics, pages 177 ff.
6. Pevzner, Pavel: Computational Molecular Biology, 2000, chapter 8.5 (“Frequent Words in DNA”)
7. http://en.wikipedia.org/wiki/Em_algorithm

Motivation

The goal of *motif finding* is the detection of *novel, unknown* signals in a set of sequences. For example you might want to detect transcription factor binding sites in a genome, but do not know what they look like.

However, one can be quite certain that such motifs are never conserved exactly but only approximately. This fact makes the problem difficult (exercise: how can you compute a solution if no errors occur?)

Motivation (2)

Lets have a look at a toy example that already illustrates the difficulty of the problem at hand:

```
0     5     10    15    20    25    30    35    40    45
agcaatcgcccgtattccgttaaagcctgcctcgctagctcgaagctg
ggtcttgcgtgcatcgctaagctagcaaccgctagcatgcgctagcct
gattcgaataggcaaacgcacgaagtccgttaaagctagcatcgatcg
gctagctagcactattccgttttagcgatccgcctagccagagagatc
ccgctcgatcgtagcggatcgctagcatttcgttatccgtgcatagcg
```

Do you see the implanted motif of length 10 with 2 errors?

Most algorithms can find the correct motif and the correct implanted positions.

Motivation (3)

Here is the solution.

```
0     5     10    15    20    25    30    35    40    45
agcaatcgccCGTATTCCGTtaaagcctgcctcgctagctcgaagctg
GGTCTTGCGTgcatcgctaagctagcaaccgctagcatgcgctagcct
gattcgaataggcaaacgcaCGAAGTCCGTtaaagctagcatcgatcg
gctagctagcACTATTCCGTtttagcgatccgcctagccagagagatc
ccgctcgatcgtagcggatcgctagcatttCGTTATCCGTgcatagcg
```

We want of course to do this in an automated way. A paper by Pevzner sparked a lot of research in this area and many subsequent paper adopted his formal definition of the motif searching problem and use it as a benchmark problem.

Definition

Definition 1. Planted (l, d)**-motif problem:** Assume there is a fixed motif m of length l. The problem is to find m, given t sequences of length n, each containing a variant m' of m with the property that m differs from m' in exactly d positions.

In the original paper Pevzner et. al. considered it hard to solve the $(15, 4)$-motif problem in 20 sequences of length 600 and presented two algorithms WINNOWER and SP-STAR that were able to solve the problem. He failed however for the $(14, 4)$, $(16, 5)$, and $(18, 6)$ problem. (Note that we are also interested in occurrences with less then d differences. Requiring d differences is for the sake of benchmarking).

Can we get a good feeling of how difficult the different instances of the problem are?

Random background

It is quite illustrative to have a close look at the expected number of (l, d)-motifs in the problem. Lets for simplicity assume that the background sequences are i.i.d. distributed. Then the probability that a given l-mer C occurs with up to d substitutions at a given position of a random sequence is:

$$p_d = \sum_{i=0}^{d} \binom{l}{i} \left(\frac{3}{4}\right)^i \left(\frac{1}{4}\right)^{l-i}$$

The *expected* number of length l motifs that occur with up to d substitutions at least once in each of the t random length n sequences is approximately

$$E(l, d) \approx 4^l (1 - (1 - p_d)^{n-l+1})^t .$$

However this is only a heuristic estimate. . . can you see the error?

Random background (2)

The above formulas are only heuristic estimates since they do not model overlapping motifs, and – in general – the assumption of i.i.d. background distribution is not true. Nevertheless the formula gives a good estimate of the hardness of the respective problem.

For example we can estimate that 20 random sequences of length 600 are expected to contain more than one $(9, 2)$-motif *by chance*, whereas the chances of finding a random $(10, 2)$-motif are less than 10^{-7}.

This is a stunningly sharp boundary; a fact that should be kept in mind.

Random background (3)

Pevzners book contains an interesting discussion of the occurrence distribution and the "overlapping words paradox". For simplicity we consider exact matches. To avoid issues with margin effects, assume that the text is circular and has length n. Let l be the alphabet size. Let k be the size of the pattern.

Then the probability that at position i in the text there is an occurrence of the pattern is $p := 1/l^k$ and the *expected* number of occurrences is np.

However the *variance* has a much more complicated formula and *depends on the pattern*. Not all patterns have the same variance! An intuitive explanation is the following: Consider two patterns $P = AAAA$ and $P' = ACGT$. *If* there is an occurrence of P at position i, then the conditional probability for an occurrence of P at $i + 1$ is $1/l$. But for P', the conditional probability is zero – all occurrences of P' must be non-overlapping.

Random background (4)

This is formalized in the *autocorrelation polynomial* K_{PP}. For two patterns A and B, both of length k, the *correlation polynomial* is defined as

$$K_{AB}(x) = \sum_{i=0}^{k-1} c_i x^i ,$$

where the coefficient c_i is 1 if the $(k-i)$-prefix of B is identical to the $(k-i)$-suffix of A, and 0 otherwise.

Now, the variance of the number of occurrences of a pattern P in a circular text is

$$\mathrm{Var}(P) = \frac{n}{l^k} \cdot \left(2 \cdot K_{PP}(\frac{1}{l}) - 1 - \frac{2k-1}{l^k} \right) .$$

Actually, the proof is not very complicated. The analysis of word occurrences has also been generalized to approximate matches.

Types of algorithms

We can use two main components to classify motif searching algorithms. The first distinction can be made on whether the algorithms search in the *space of starting positions*, or whether they search in *motif space* starting from some suitable initial motifs. Most modern algorithms do the latter.

The second distinction can be made upon whether the algorithms work internally with *patterns* or with *profiles*. The second approach has some advantages in finding motifs with many degenerate positions but are in general somewhat more costly.

Types of algorithms

Most algorithms derive at the end of the first stage a profile as a motif (whether they use profiles directly or construct it at the end of the pattern based search).

Then they usually refine the pattern using some common local strategies like Gibbs sampling, exhaustive local search, or quite often the Expectation-Maximization algorithm.

We will present two fast and successful algorithms, PROJECTION by Buhler and Tompa which is a pattern based approach that has a refinment stage in the end using the EM algorithm, and if time allows an algorithm py Price et al. that comes in two flavors PATTERNBRANCHING and PROFILEBRANCHING.

We will first describe the Expectation-Maximization algorithm and then the two algorithms.

The expectation-maximization algorithm

The *expectation-maximization (EM)* algorithm is a general paradigm which covers many iterative algorithms for parameter estimation.

Assume that a statistical model is determined by *model parameters* θ. The model shall be adapted to explain the *observed data* with maximum likelihood. Depending on the nature of the problem, this can be a difficult task.

Sometimes it is easier to consider a related parameter estimation problem, in which the observed data are augmented by *missing data*. These are also called *missing information* or *latent data*. Changing the model this way is sometimes called *data augmentation*.

The EM algorithm (2)

The EM algorithm formalizes an intuitive idea for obtaining parameter estimates when some of the data are missing. It repeats the following two steps until convergence:

'E' step: Estimate the missing information using the current model parameters.

'M' step: Optimize the model parameters using the estimated missing information.

The EM algorithm (3)

In the motif finding example, the observed data are the input sequences and the missing information are the positions of the planted motifs. Assume that the model consists of a background nucleotide distribution and a profile matrix which consists of the nucleotide distributions for each position of the motif.

It is much easier to compute the probability that a given input sequence was generated accoring to such a model if we know the positions of the planted motifs than without this additional information.

The EM algorithm (4)

This idea has been in use for many years before Orchard and Woodbury (1972) in their *missing information principle* provided the theoretical foundation of the underlying idea. The term EM was introduced by Dempster, Laird, and Rubin (1977) where proof of general results about the behavior of the algorithm was first given as well as a large number of applications.

The EM algorithm (5)

Let us introduce a bit of notation. Denote the *observed* part of the data by x, and the *missing* information by y. (In our case, x represents the input sequences and y the positions of the planted motifs.)

The aim is to find the model parameters θ maximizing the likelihood given the observed data or, equivalently, the log likelihood:

$$\arg\max_{\theta} \log P(x \mid \theta) .$$

The EM algorithm shall be used to solve this optimization problem. The log likelihood for the observed data might be hard to deal with directly, thus we start by considering the conditional log likelihood for the observed data given the missing information:

$$\log P(x \mid y, \theta) .$$

Note that formally the missing information acts just like additional model parameters which are excluded from optimization. The question is, what should we plug in for the missing information y?

The EM algorithm (6)

As the name says, we do not have the missing information. But of course, using the EM scheme can make sense only if the model allows us to predict the missing information in some way. If we consider $\log P(x \mid y, \theta)$ as a function of y, we obtain the posterior probability for y, given the observed data x and some model parameters θ, using Bayes' theorem as follows:

$$P(y \mid x, \theta) = \frac{P(x, y \mid \theta)}{P(x \mid \theta)} = \frac{P(x \mid y, \theta) P(y \mid \theta)}{\sum_{y'} P(x \mid y', \theta) P(y' \mid \theta)}.$$

Note that this formulation only requires knowledge of the observation likelihood given the missing information, $P(x \mid y, \theta)$, and the probability of the missing information, $P(y \mid \theta)$.

The EM algorithm works by improving an initial estimate θ^0 of the model parameters until a convergence criterion is met. In this way, we can rely on an estimate θ^t from the previous round of the main loop, and use $P(y \mid \theta^t)$ as an estimate for the distribution of the missing information according to some maximum likelihood model parameters.

The EM algorithm (7)

Since we do not know the missing information y exactly, it is not realistic to rely on $\log P(x \mid y, \theta)$ for only one particular assignment of the missing information.

Instead we consider the *expected value of the log likelihood* $\log P(x \mid y, \theta)$ of the observations x and the model parameters θ *according to the posterior distribution of the missing information y (which is estimated according to θ^t)*.

Formally this means that we replace the log likelihood

$$\log P(x \mid \theta) = \log \sum_y P(y \mid x, \theta)\, P(x \mid y, \theta)$$

by

$$\mathrm{el}(x, \theta) := \sum_y P(y \mid x, \theta)\, \log P(x \mid y, \theta)$$

which is a different function (!) but more amenable to optimization.

A greedy (short-sighted) approach would be to choose θ' such that $\mathrm{el}(x \mid \theta')$ is greater than $\mathrm{el}(x \mid \theta^t)$. Let us try this out.

The EM algorithm (8)

In the sum on the right hand side we have

$$P(x, y \mid \theta) = P(y \mid x, \theta) P(x \mid y, \theta)$$

and hence,

$$\log P(x, y \mid \theta) = \log P(y \mid x, \theta) + \log P(x \mid y, \theta)$$

or

$$\log P(x \mid y, \theta) = \log P(x, y \mid \theta) - \log P(y \mid x, \theta)$$

The EM algorithm (9)

Therefore,

$$\sum_y P(y \mid x, \theta^t) \log P(x \mid y, \theta)$$

$$= \underbrace{\sum_y P(y \mid x, \theta^t) \log P(x, y \mid \theta)}_{(I)} - \underbrace{\sum_y P(y \mid x, \theta^t) \log P(y \mid x, \theta)}_{(II)}$$

Ignore the second sum on the right hand side (II) for a moment. We denote the first sum on the right hand side (I) with

$$Q(\theta, \theta^t) := \sum_y P(y \mid x, \theta^t) \log P(x, y \mid \theta)\,.$$

Note that $Q(\theta, \theta^t)$ is the average of $\log P(x, y \mid \theta)$ over the posterior distribution of y obtained with the current parameters θ^t.

The EM algorithm (10)

$$Q(\theta, \theta^t) := \sum_y P(y \mid x, \theta^t) \log P(x, y \mid \theta) .$$

Then the (greedy) update formula for the model parameters takes the form:

$$\theta^{t+1} = \arg\max_\theta Q(\theta, \theta^t) .$$

This update formula subsumes both steps of the EM algorithm: The updated model parameters θ^{t+1} are chosen such as to *maximize (M)* the *conditional expectation (E)* of the complete data log likelihood, where the observed data are given, and the missing information is estimated according to the previous parameter values θ^t.

The factors $P(y \mid x, \theta^t)$ are computed in the E-step, e. g. using Bayes' theorem as shown above. They are excluded from the optimization in the M-step. Thus we maximize a weighted sum of log likelihood functions, one for each outcome of the hidden variables y.

Next we look at the other summand (II) in the objective function.

The EM algorithm (11)

We have

$$\sum_y P(y \mid x, \theta^t) \log P(y \mid x, \theta^t) - \sum_y P(y \mid x, \theta^t) \log P(y \mid x, \theta)$$

$$= \sum_y P(y \mid x, \theta^t) \log \frac{P(y \mid x, \theta^t)}{P(y \mid x, \theta)}$$

Observe that this is just the relative entropy of $P(y \mid x, \theta^t)$ with respect to $P(y \mid x, \theta)$. It is well-known that the relative entropy is always non-negative (exercise) and zero if and only if $\theta = \theta^t$.

Hence it holds that

$$0 \leq Q(\theta^{t+1} \mid \theta^t) - Q(\theta^t \mid \theta^t) \leq \mathrm{el}(x, \theta^{t+1}) - \mathrm{el}(x, \theta^t)$$

with equality only if $\theta^{t+1} = \theta^t$. It follows that we will always make the difference positive and thus increase the likelihood of x under the new model, unless we have $\theta^{t+1} = \theta^t$.

The EM algorithm (12)

One can show that under reasonable assumptions the EM algorithm will approach a local optimum of the objective function. This is not guaranteed for the parameter set θ, though.

The MEME models

The different types of sequence model supported by MEME make differing assumptions about how and where motif occurrences appear in the dataset.

- The simplest model type is called *OOPS*, since it assumes that there is exactly One Occurrence Per Sequence of the motif in the dataset.
- A generalization is called *ZOOPS*, which assumes Zero Or One Motif Occurrences Per Sequence.
- Finally, *TCM* (Two-Component Mixture) models assume that there are zero or more non-overlapping occurrences of the motif in each sequence in the dataset.

The MEME models (2)

A motif is modeled by a sequence of discrete random variables parameters give the probabilities of each of the different letters (4 in the case of DNA, 20 in the case of proteins) occurring in each of the different positions in an occurrence of the motif.

The background positions in the sequences are modeled by a single discrete random variable for each of the different letters. Hence the complete model can be see as a $4 \times (l + 1)$ profile in the case of nucleotides.

The MEME models (3)

Consider searching for a single motif in a set of sequences by fitting one of the three sequence model types to it. The dataset consists of n sequences, each of length l.

The starting point(s) of the occurrence(s) of the motif (if any) in each of the sequences are unknown and are the ”missing information”.

The EM algorithm tries to optimize the chosen model for the (M-step) as well as the estimated starting positions (E-step). For details consider the MEME paper (reference in the beginning).

Since the EM algorithm only converges to a local maximum, it is important to have good starting positions. To compute these we can use algorithms like *PROJECTION*, *PATTERNBRANCHING*, or *PROFILEBRANCHING*, described later.

Example

Lets look at a small example in the context of motif finding. Assume we are given the data $x = x_1, x_2, x_3$ as follows. It is the observed data.

	1	2	3	4	5	6
x_1	A	**C**	**A**	**G**	C	A
x_2	A	G	G	**C**	**A**	**G**
x_3	T	**C**	**A**	**G**	T	C

We are missing the start positions z_{ij} of the hidden motif and want to represent them by a matrix w where w_{ij} is the probability that the pattern starts at position j in sequence i.

Assume that a motif finding algorithm resulted in the following model parameters θ which in our case is a $4 \times (l+1)$ matrix p describing in the 0th column the background probabilities of the 4 nucleotides and in the other l positions the probabilities that a certain letter is in the motif.

Example (2)

Assume that our motif has length three and is

	0	1	2	3
A	0.25	0.1	0.5	0.2
C	0.25	0.3	0.2	0.1
G	0.25	0.3	0.1	0.4
T	0.25	0.3	0.2	0.3

We use this initial guess now to estimate the missing data w.

By our assumption that each sequence contains exactly one occurrence of the motif, we can write

$$w'_{ij} = P(z_{ij} = 1 \mid x, p) = \frac{P(x \mid z_{ij} = 1, p)}{\sum_{k=1}^{4} P(x \mid z_{ik} = 1, p)}.$$

Example (3)

This yields the following matrix w:

1	2	3	4
0.0520	**0.7790**	0.0130	0.1558
0.1108	0.0416	0.0166	**0.8390**
0.0170	**0.8547**	0.0427	0.0855

Now we estimate the missing data using our initial model. We can then refine the model by assuming the probabilities for the motif starting positions are correct.

Example (4)

If we ask now about the probability of each letter we can re-estimate the new model by updating the frequencies of each letter with the weights given by w. For example for the first pattern position being a C we add $w_{1,2} + w_{2,4} + w_{3,2}$ to the previous frequency, that is $p'_{C,1} = 0.7790 + 0.8390 + 0.8547 + 0.3$ and so on. Then the new frequencies need to be normalized, that is $p_{C,1} = \frac{p'_{C,1}}{\sum_{i=A,C,G,T} p'_{i,1}}$. This results in:

	0	1	2	3
A	....	0.079	0.647	...
C	....	0.692	0.198	...
G	....	0.150	0.093	...
T	....	0.079	0.062	...

As one can see the new model tends to model the motif *CAG* quite well. Now we will discuss algorithms that produce a good starting model.

PROJECTION

PROJECTION is a probabilistic algorithm that iteratively increases the chance that it finds the correct motif.

It has three key parameters:

- the *projection size* k.
- the *bucket threshold* s
- the number of *independent trials* m

Given these parameters, which we will explain later, the algorithm is quite simple.

PROJECTION (2)

(1) PROJECTION($k, s, m, s_1, \ldots, s_t$);
(2) for $i = 1$ to m do
(3) choose randomly k different positions $l_k \subset \{1, \ldots, l\}$;
(4) for each l-mer $x \in s_1, \ldots, s_t$ do
(5) // $h(l_k, x)$ *is a hash function using the positions in* l_k;
(6) compute $h(l_k, x)$;
(7) store x in the hash bucket $h(l_k, x)$;
(8) od
(9) for each bucket with $y \geq s$ elements do
(10) refine bucket with EM algorithm;
(11) od
(12) od
(13) choose consensus pattern of best bucket;

PROJECTION (3)

What is the key idea of the algorithm?

For a fixed choice of I_k positions and the unknown (planted) motif M, let us call the bucket $h(M)$ the *planted bucket*. If $k < l - d$, then, because of the randomness of I_k, there is a good chance that a number of the t planted instances of M will hash to the *same* bucket.

If we choose the number of buckets large enough, then the planted bucket should have a significantly higher count than a bucket containing random l-mers.

PROJECTION (4)

The outcome of the PROJECTION algorithm is dependent on the distribution of the errors in the copies of the motif and the choices of I_k. To increase the chance of success, we repeat the hashing using different random selections of I_k for *m* times.

We do not know which bucket is the planted bucket. Therefore we inspect each bucket that has more than *s* elements.

Each of those buckets is then *refined* using the EM algorithm (other local searches are possible).

Finally, the consensus sequence of the bucket with the highest expected value is chosen as the motif.

Next we discuss how the parameters *k* and *s* should be chosen.

Choice of k and s

For the above argument to work we obviously have to choose $k < l - d$, since only then do we have a chance that a number of the planted projections hashes into the same bucket. In particular the planted instances for which the d mutated positions are disjoint from the k hash positions will hash to the planted bucket.

On the other hand we do not want to choose k too small to avoid the contamination of the planted bucket by random background sequences. Since we are hashing $t(n - l + 1)$ l-mers into 4^k buckets, the average bucket will contain less than one random l-mer if we choose $4^k > t(n - l + 1)$.

If we can do this, then we are able to choose a relatively low bucket size threshold $s \approx 3$ or 4.

Choice of m

Finally we come to the determination of the number m of independent trials to run. In the experiments they choose m such that the probability is at least $q = 0.95$ that the planted bucket contains s or more planted motif instances in at least one of the m trials. We determine m with the following argument:

Let $p(l, d, k)$ be the probability that a given planted motif instance hashes to the planted bucket, that is, $p(l, d, k) = \binom{l-d}{k} / \binom{l}{k}$. (For example $p(15, 4, 8) = 165/6435 \approx 0.026$ or $(P(19, 6, 8) = 715/92378 \approx 0.0077$.)

Let $\hat{t}$ be an estimate of the number of sequences containing a planted motif instance ($\hat{t} = t$ in the challenge problem).

Then the probability that fewer than s planted instances hash to the planted bucket in a given trial is $B_{\hat{t},p(l,d,k)}(s)$, where $B_{t,p}(s)$ is the probability that there are fewer than s successes in t independent Bernoulli trials, each trial having probability p of success.

Choice of m (2)

Thus the probablity that s or more planted instances hash to the planted bucket in at least one trial is

$$1 - (B_{\hat{t},p(l,d,k)}(s))^m$$

We want this entity to be larger or equal to q. Hence it follows:

$$m \geq \lceil \frac{\log(1-q)}{\log(B_{\hat{t},p(l,d,k)})(s))} \rceil$$

Given the choices of s and k usually a couple of hundred trials suffices.

Motif refinement

Now we have found a set of buckets that needs to be explored to recover the planted motif.

PROJECTION starts the refinement using the EM algorithm based on the following, simple probabilistic model.

- An instance of some length l motif occurs only once per input sequence.
- Motif instances are generated from a $4 \times l$ weight matrix model W whose (i, j)th entry gives the probability that base i occurs at position j of an instance, independent of its other positions.
- The remaining $n - l$ residues in each sequence are choosen randomly and independently according to some background nucleotide distribution.

Motif refinement (2)

Now let S be the set of t input sequences and let P be the background distribution.

The EM-based refinement seeks a weight matrix model W^* that maximizes the likelihood ratio $Pr(S \mid W^*, P)/Pr(S \mid P)$, that is, a motif that explains the observed sequences much better than the background model alone.

Starting from some initial guess W_0 the EM algorithm will converge against a local maximum of the above ratio.

Motif refinement (3)

PROJECTION performs EM refinement on every bucket with at least s elements. It forms an initial guess W_h from a bucket h as follows:

Set $W_h(i, j)$ to be the frequency of base i among the jth positions of all l-mers in h. This guess forms a *centroid* for h. In order to avoid zero entries in W_h a Laplace correction b_i is added to $W_h(i, j)$, where b_i is the background probability of nucleotide i in the input.

Motif refinement (4)

If W_h^* is the candidate motif model refined from W_h, then we compute the motif from the model as follows:

- From each input sequence choose the l-mer sequence x with the *largest* likelihood ratio $Pr(x \mid W_h^*)/Pr(x \mid P)$. The multiset T of l-mers represents the motif in the input that is most consistent with W_h^*. Let C_T be the consensus of T, and let $s(T)$ be the number of elements of T whose Hamming distance to C_T exceeds d.

- For biological examples PROJECTION outputs the T that minimizes $s(T)$ over all buckets and trials.

Motif refinement (5)

For the challenge problem, T is further refined heuristically. PROJECTION computes the consensus C of the sequences in T, and defines the score of T to be the number of sequences in T whose Hamming distance to C is at most d.

Now let T' be the set that contains the l-mer from each input sequence that is closest in Hamming distance to C. If the score of T' is greater than the score of T, they replace T by T' and repeat. This procedure converges usually after a few iterations.

Performance

The following table gives an overview of the performance of PROJECTION compared to other motif finders on the (l, d)-motif problem. The measure is the *average performance* defined as $| K \cap P | / | K \cup P |$ where K is the set of the $l \times t$ residue positions of the planted motif instances, and P is the corresponding set of positions predicted by the algorithm.

l	d	Gibbs	WINNOWER	SP-STAR	PROJECTION
10	2	0.20	0.78	0.56	**0.82**
11	2	0.68	**0.90**	0.84	**0.91**
12	3	0.03	0.75	0.33	**0.81**
13	3	0.60	**0.92**	**0.92**	**0.92**
14	4	0.02	0.02	0.20	**0.77**
15	4	0.19	**0.92**	0.73	**0.93**
16	5	0.02	0.03	0.04	**0.70**
17	5	0.28	0.03	0.69	**0.93**
18	6	0.03	0.03	0.03	**0.74**
19	6	0.05	0.03	0.40	**0.96**

PATTERNBRANCHING

PATTERNBRANCHING is a deterministic algorithm that is pattern based and uses a greedy local search to find the correct motif. It considers each *l-mer* as a potential candidate for the pattern $P = p_1, \ldots, p_l$.

Starting from an initial guess A_0 it constructs a path of patterns $A_0 \rightarrow A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_k$ with the property that pattern A_i is in the Hamming distance 1-neighborhood $D_{=1}(A_{i-1})$ of pattern A_{i-1}. ($D_{=k}(A)$ is the set of patterns of distance exactly k of A.)

The path is constructed by iteratively applying the function `BestNeighbor` which maps pattern A to the best neighbor in $D_{=1}(A)$. Finally A_k is scored and compared to the outcome computed with other starting positions.

PATTERNBRANCHING (2)

Two questions need to be adressed.

1. Given a pattern A, how do we score it?

2. How do we compute `BestNeighbor`(A)?

First, we *score* a pattern A using its *total* distance from the sample. For each sequence S_i in the sample $\mathcal{S} = \{S_1, \ldots, S_n\}$, let $d(A, S_i) = \min\{d(A, P) \mid P \in S_i\}$, where P denotes an l-mer. Then the total distance of A from the sample $\mathcal{S}$ is $d(A, \mathcal{S}) = \sum_{S_i \in \mathcal{S}} d(A, S_i)$.

Second, `BestNeighbor(A)` is the pattern $B \in D_{=1}(A)$ with the lowest total distance $d(B, \mathcal{S})$.

PATTERNBRANCHING (3)

```
(1)  PATTERNBRANCHING(S, l, k);
(2)  bestMotif = arbitrary motif pattern;
(3)  bestScore = d(bestMotif, S)
(4)  for each l-mer A_0 in S do
(5)      for j = 0 to k do
(6)          if d(A_j, S) < bestScore
(7)              bestMotif = A_j;
(8)              bestScore = d(bestMotif, S);
(9)          fi
(10)         A_{j+1} = BestNeighbor(A_j);
(11)     od
(12) od
(13) output bestMotif;
```

PATTERNBRANCHING (4)

If we want to do a more thorough search of $D_{=k}(A_0)$, we can keep a set $\mathcal{A}$ of r patterns at each iteration instead of a single pattern and define `bestNeighbor`($\mathcal{A}$) to be the set of r pattern $B \in D_{=1}(\mathcal{A})$.

The algorithms can be speeded up by approximating the total distance $d(B, \mathcal{S})$ of each pattern $B \in D_{=1}(A_j)$ by estimating $d(B, S_i) = \min\{d(B, P) \mid P \in S_i\}$ using only patterns $P \in S_i$ which satisfy two conditions:

PATTERNBRANCHING (5)

- $d(A_j, P) \leq 2k - j$, which is especially satisfied for the case when P is an occurrence of the correct motif M with at most k mutations and the path of best neighbors $A_j \rightarrow ... \rightarrow A_k$ leads to M

- P agrees with B at the nucleotide changed from A_j, which is likely to be true for the pattern $P \in S_i$ minimizing $d(B, P)$.

By storing the values $d(A_j, P)$ we can quickly compute the estimate of $d(B, S_i)$ for all $B \in D_{=1}(A_j)$.

Another possible speedup is to adaptively change the number of neighbors kept by limiting the total distance to $\mathcal{S}$.

PROFILEBRANCHING

The PROFILEBRANCHING algorithm is similar to the PATTERNBRANCHING algorithm, with the following changes:

1. convert each sample string A_0 to a profile $X(A_0)$
2. generalize the scoring method to score profiles
3. modify the branching method to apply to profiles
4. use the top scoring profile as a seed to the EM algorithm

PROFILEBRANCHING (2)

To convert a sample string to a profile we define $X(A_0)$ as a $4 \times l$ profile matrix (x_{vw}) which in column w has probability $x_{wv} = 0.5$ for nucleotide $v = a_w$ and $x_{wv} = 1/6$ for each other nucleotide. The total distance score for patterns is replaced by an *entropy score* for profiles: Given a profile $X = (x_{vw})$, and a pattern $P = p_1 \ldots p_l$, let $e(X, P)$ be the log probability of sampling P from X, i. e. $e(X, P) = \sum_{w=1}^{l} \log(x_{p_w w})$.

Then for each $S_i \in \mathcal{S}$ let $e(X, S_i) = \max\{e(X, P) \mid P \in S_i\}$.

PROFILEBRANCHING (3)

For a profile X we define $\mathcal{D}_{=1}(X)$ to be the set of profiles obtained from X by amplifying a single nucleotide in a single position w of X to create a profile $\bar{X} = (\bar{x}_{vw})$ with *relative entropy* equal to ρ.

The relative entropy is defined as $\sum_v x_{vw} \log(\bar{x}_{vw}/x_{vw})$. We use $\rho = -0.3$ (the parameter was not optimized).

For example, given the probabilities $(\frac{1}{2}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6})$, we obtain $(0.27, 0.55, 0.09, 0.09)$ by amplifying the second nucleotide (exercise: verify this (the ratios of the non amplified nucleotides are fixed to a constant d)). At a given position w, we only amplify a nucleotide v if $x_{vw} < 0.5$.

PROFILEBRANCHING (4)

The resulting algorithm is quite similar to PATTERNBRANCHING. For each l-mer A_0 in $\mathcal{S}$, we let $X_0 = X(A_0)$ and construct a path of profiles $X_0 \rightarrow X_1 \rightarrow ... \rightarrow X_k$ by iteratively applying `BestNeighbor` for profiles, which maps a profile X to a local improvement of its best neighbor in $\mathcal{D}_{=1}(X)$. The best neighbor Y is the profile with the highest entropy $e(Y, \mathcal{S})$.

After branching for k iterations from each l-mer A_0 in the sample, we run the EM algorithm to convergence on the top scoring profile we have found.

PROFILEBRANCHING (5)

```
(1)  PROFILEBRANCHING(S, l, k);
(2)  bestProfile = arbitrary motif profile;
(3)  bestScore = e(bestProfile, S);
(4)  for each l-mer A_0 in S do
(5)      X_0 = X(A_0);
(6)      for j = 0 to k do
(7)          if e(X_j, S) > bestScore
(8)              bestProfile = X_j;
(9)              bestScore = e(bestProfile, S);
(10)         fi
(11)         X_{j+1} = BestNeighbor(X_j);
(12)     od
(13) od
(14) run EM algorithm on bestProfile;
(15) output resulting profile;
```

Performance

The PROFILEBRANCHING algorithm can be speeded up with similar ideas as the PATTERNBRANCHING algorithm.

In summary the PROFILEBRANCHING algorithm takes about 5 times as long as the PATTERNBRANCHING algorithm. On the motif challenge problem PATTERN-BRANCHING performs almost as well as other methods (PROJECTION, MITRA, MULTIPROFILER), but runs significantly faster (a factor of 20-100 times).

Performance (2)

On the motif challenge problem PATTERNBRANCHING outperforms PROFILE-BRANCHING, but on motifs that are degenerate it is the other way around. For example consider a $(15, 5)$ problem in $n = 20$ sequences of length $N = 600$ with the restriction that all mutations of a position mutate to a fixed second nucleotide.

This is a hard implanted problem where pattern based approaches have an average performance coefficient of 0.1, while that of MEME is 0.63 and that of PROFILEBRANCHING is 0.99. Also for even more subtle patterns PROFILEBRANCHING outperforms MEME.

Performance (3)

For example, if we allow in each occurrence of the motif one of five mutations to mutate to a third nucleotide value for that position, the average performance coefficients decline to 0.03 for PATTERNBRANCHING, 0.03 for MEME, and only 0.62 for PROFILEBRANCHING.

The authors unfortunately do not give the exact model used in their EM algorithm.

Summary

- Motif search algorithms fall in basically two classes. The ones that consider the space of all starting positions of the motif, and the ones that consider motif space and search for the motif given a suitable start motif.

- The algorithms we proposed conduct a local search and end up with a proposal of the motif. This proposed motif is then refined using Expectation Maximization (EM).

- The EM algorithm is an iterative procedure that estimates missing values by estimated values and then uses this guess to estimate the parameters (maximize the expected value with repsect to the parameter set). The EM algorithm converges to a local maximum.