SNAP: HOW SCORES ARE COMPUTED

AVIAD ROZENHEK

Contents

1. Basic notation

This document describes how SeedSearcher scores motifs, and consists mainly of definitions, which are mathematical in nature, but quite simple.

Definition: Sequence. A *sequence* S is a finite vector of characters from an alphabet which we shall keep fixed. The j-th coordinate S_j of the sequence shall be called the j-th position. We shall identify S with the set of all positions of S, thus we can (ab)use notation like $S_j \in S$.

Usually we work with gene promotor sequences, where a position is a downstream offset.

Definition: Database. An ordered set G of sequences shall be called a *database*. We shall think of G as fixed, with cardinality $N =$ number of sequences. Thus every sequence S shall be denoted by $S^i \in G$ for some i.

Usually a database is a collection of named gene promoter sequences. We are interested in scoring different "motifs" against the database. We shall identify a motif with the set of positions in the database, where the motif 'resides' (usually the starting position of the motif). We shall calls these sets Clusters:

Definition: PCluster. An *i-Position Cluster* C^i is a set of positions, all belonging to the i-th sequence: $C^i \subseteq S^i$. We shall denote by $\bar{C^i} = \{C^i : C^i \subseteq S^i\}$ the set of all i-Position Clusters.

Definition: Cluster. An ordered set (or vector) C of position clusters shall be called a *Cluster*. $C \in \prod_i \bar{C^i}$. that is, a cluster contains a (possibly empty) pcluster for every sequence in the database. We shall denote by $\bar{C} = \{C : C \in \prod_i \bar{C^i}\}$ the set of all Clusters.

We would like to associate Clusters with Scores. A score system is be a mapping of Clusters to classifications. A score function is a mapping of classifications to scores. Formally:

Definition.

A *score system* is a mapping $\psi : \bar{C} \to \left(\begin{smallmatrix} T_p & T_n & F_p & F_n \end{smallmatrix} \right)$
A *score function* is a mapping $\pi : \left(\begin{smallmatrix} T_p & T_n & F_p & F_n \end{smallmatrix} \right) \to \Re$

Where

$$\begin{aligned} T_p &= \text{True Positives} && = \text{number of correct positive classifications} \\ T_n &= \text{True Negatives} && = \text{number if correct negative classifications} \\ F_p &= \text{False Positives} && = \text{number of incorrect positive classifications} \\ F_n &= \text{False Negatives} && = \text{number of incorrect negative classifications} \end{aligned}$$

We can now define the two score functions which SeedSearcher employs:

2. Score functions

2.1. **Hyper-Geometric scoring function.**

We define the *Hyper Geometric Pvalue* P_{hg} to be:

$$P_{hg} \left(\begin{smallmatrix} T_p & T_n & F_p & F_n \end{smallmatrix} \right) = \sum_{x \geq k} D_{hg} \left(\begin{smallmatrix} x & n & K & N \end{smallmatrix} \right) \tag{1}$$

Where

$$\begin{pmatrix} x \\ n \\ K \\ N \end{pmatrix} = \begin{pmatrix} T_p & & & \\ T_p & +F_p & & \\ T_p & +F_n & & \\ T_p & +F_p & +T_n & +F_n \end{pmatrix} \tag{2}$$

and D_{hg} is the hyper-geometric distribution:

$$D_{hg}(x, k, K, N) = \frac{\begin{pmatrix} K \\ x \end{pmatrix} \begin{pmatrix} N - K \\ n - x \end{pmatrix}}{\begin{pmatrix} N \\ n \end{pmatrix}} \tag{3}$$

Thus P_{hg} is a pvalue and equals the sum of the tail of the hyper-geometric distribution.
The distribution D_{hg} is more easily understood using the following scenario: Let there be an urn with N balls, K of which are *red* and suppose we draw n balls from the urn, x of those are red. then $H(x, n, K, N)$ is the chance to draw n balls and that *at least* x of these will be red.

Remark. Naturally, values of P_{hg} are numbers $\in [0, 1]$ and can be very close to 0. So for both numerical stability and ease of use we shall define and use:

$$h \left(\begin{smallmatrix} T_p & T_n & F_p & F_n \end{smallmatrix} \right) = -\log_{10} Pval_{hg} \left(\begin{smallmatrix} T_p & T_n & F_p & F_n \end{smallmatrix} \right) \tag{4}$$

This h shall be called the *hyper-geometric scoring* function

2.2. **Exploss scoring function.**

This is a much simpler score function, which only "rewards" (or "punishes") according to correct (or incorrect) positive classifications: The score is increased when a correct positive classification is made (T_p), and is decreased when an incorrect positive classification is made (F_p). negative classifications are ignored (T_n, F_n). This is still a discriminative function because it relies on classifications of both the positive and negative sets, unlike non-discriminate functions which will rely only on classifications of the positive set, e.g. T_p and F_n.

Definition.
Let $1 < \alpha, \beta$ be any fixed constansts. we define the exploss function E to be:

$$E_{\alpha,\beta}(T_p, F_p) = \frac{\beta^{F_p}}{\alpha^{T_p}} \tag{5}$$

This function receives values $\in (0, \infty)$ and it is easily seen that the better the positive classification, the closer the score is to 0. so we again take:

$$e_{\alpha,\beta}(T_p, T_n, F_p, F_n) = -\log_{10} Exploss_{\alpha,\beta}(T_p, F_p) \tag{6}$$

This e shall be called the *exploss scoring* function

3. Score systems

We shall now recall the definitions in section 1, and define several ways to map Clusters to $\left({\scriptstyle T_p\ T_n\ F_p\ F_n} \right)$ classifications. We shall assume to have a constant database of size N, with weights $W^i \in [0, 1]$ on the sequences $s^i \quad i = 1...N$ and also positional weights $W^i_j \in [0, 1]$ on any position in any sequence.

Remark. In the simplest case, the prior on the regulation of s^i is discrete, and there is no positional weight prior. thus:

$$\forall i \quad W^i = \begin{cases} 1 & \text{if The sequence } s^i \text{ is positively labeled} \\ 0 & \text{if The sequence } s^i \text{ is negatively labeled} \end{cases}$$

and

$$\forall i \forall j \quad W^i_j = 1$$

To gain some intuition, let us retreat from the abstract and focus our attention on the specific problem which we are trying to solve. We are trying to find a motif (short IUPAC sequence) which will correctly separate a database of gene promoters (sequences) into regulated (positively labeled) and non-regulated (negatively labeled) sets, which are known in advance. Now suppose we have such a motif m of length l. m infers a Cluster c which contains all the positions in the promoters where m starts. we are interested in defining several ρ classification functions such that:

$$\rho(c) = \begin{pmatrix} T^\rho_p(c) \\ T^\rho_n(c) \\ F^\rho_p(c) \\ F^\rho_n(c) \end{pmatrix} = \sum_i \rho_i(c^i) = \sum_i \begin{pmatrix} T^{\rho_i}_p(c^i) \\ T^{\rho_i}_n(c^i) \\ F^{\rho_i}_p(c^i) \\ F^{\rho_i}_n(c^i) \end{pmatrix}$$

Thus a classification of a Cluster c is the sum of the classifications on c^i PClusters. How do we classify a PCluster? there are a couple of questions involved, with several good answers:

- *ω pweight functions:* how much does a position p^i_j (the j-th coordinate of sequence i) weigh? we can define several different weight functions on positions. we call these pweight functions, and they are denoted with the letter ω
  - *Discrete pweight*: given a threshold t (usually $t = 0.5$) we define the ω^t_d discrete pweight function with threshold t function to be:

$$\omega^t_d(p^i_j) = \begin{cases} 1 & \text{if } t < W^i \text{ (The sequence is positively labeled)} \\ 0 & \text{if } W^i <= t \text{ (The sequence is negatively labeled)} \end{cases}$$

  - We define the *real pweight* ω_r function to be:

$$\omega_r(p^i_j) = W^i$$

    Using this function a position weighs the same as the sequence it belongs to.
  - We define the positional pweight ω_p function to be:

$$\omega_p(p^i_j) = W^i \cdot W^i_j$$

    Using this function the prior on the coordinates is taken into account.
- *ρ pcluster classification functions:* How do we handle several classifications of same the same sequence? that is, how do we classify a c^i cluster which contains several positions $p^i_{j_1}, ..., p^i_{j_n}$ all belonging to the same sequence s^i? given some pweight function ω we can define several ρ_ω *pcluster classification* functions:
  - We define the ρ^1 *single position classification* function to be:

$$\rho^1_\omega(c^i) = \begin{pmatrix} \max\limits_{p^i_j \in c^i}\{\omega(p^i_j)\} \\ 1 - \max\limits_{p^i_j \in c^i}\{\omega(p^i_j)\} \end{pmatrix}$$

    Using this function only the "best" position in each pcluster is taken into accout.
  - We define the ρ^∞ *total positions classification* function to be:

$$\rho^\infty_\omega(c^i) = \begin{pmatrix} \sum\limits_{p^i_j \in c^i} \omega(p^i_j) \\ \sum\limits_{p^i_j \in c^i} 1 - \omega(p^i_j) \end{pmatrix}$$

    Using this function all the positions of a pcluster are taken into account.

Let ω be a pweight function $\in \{\omega^t_d, \omega_r, \omega_p\}$
and let ρ be a counting function $\in \{\rho^1_\omega, \rho^\infty_\omega\}$. We shall define the *score system* ρ to be

$$\begin{pmatrix} T^\rho_p(c) \\ F^\rho_p(c) \end{pmatrix} = \sum_i \begin{pmatrix} T^\rho_p(c^i) \\ F^\rho_p(c^i) \end{pmatrix} = \sum_i \rho(c^i)$$

$$\begin{aligned}\begin{pmatrix}F_n^\rho(c)\\T_n^\rho(c)\end{pmatrix} &= \sum_i \begin{pmatrix}F_n^\rho(c^i)\\T_n^\rho(c^i)\end{pmatrix}\\ &= \sum_i \left(\rho(s^i)-\rho(c^i)\right)\\ &= \sum_i \left(\rho(s^i)-\begin{pmatrix}T_p^\rho(c^i)\\F_p^\rho(c^i)\end{pmatrix}\right)\\ &= \left(\sum_i \rho(s^i)\right)-\begin{pmatrix}T_p^\rho(c)\\F_p^\rho(c)\end{pmatrix}\end{aligned}$$

We have successfully defined several scoring systems, which map clusters into $\left({\scriptstyle T_p\ T_n\ F_p\ F_n} \right)$ classifications. Summing up our method: we classify each pcluster using either the "best" position in the cluster (single position classification) or using all the positions in the cluster (total positions classification). each position is weighed using a discrete/real/positional weight function. this system maps a motif (with its corresponding cluster) to a $\left({\scriptstyle T_p\ T_n\ F_p\ F_n} \right)$ classification of the database. We use either an exploss or hyper-geometrical score function to map these classification vectors into real-valued scores.

4. REMARKS

Several issues still need further refining:

4.1. **removing overlaps in total position classifications.** When pclusters reflect the starting positions of a motif m with length l it often make sense not to allow overlaping positions. position p_j^i overlaps with p_k^i iff (assuming $w.l.g \quad j < k$) $k \in [j, j+l)$. Thus $\rho(s^i)$ now stands for a classification of the maximum overlap-free set of positions in s^i.

4.2. **DNA sequences: using the reverse strand.** When the sequences in concern are DNA, it often makes sense to "look" for motifs in both the positive and reverse strands. This affects scoring in a simple way: The reverse strands are not added as sequences, but rather they have the effect of doubling the length of all sequences. However, when a cluster c^i (or even s^i for that matter) is cleaned off overlaps the overlaps are removed separately from the positive and reverse strands.

4.3. **future changes.** The positional pweight function will change to reflect the positional bias along the entire motif (and not just the starting position)