Sequence context-specific profiles for homology searching: Supplementary Information

A. Biegert and J. Söding *

*Gene Center Munich and Center for Integrated Protein Science (CIPSM), Ludwig Maximilian University of Munich, Feodor-Lynen-Str. 25, 81377 Munich, Germany

Submitted to Proceedings of the National Academy of Sciences of the United States of America

Generation of context profile library

$N = 1$ million training profiles of length $l = 2d+1$ were generated as described in the main text and Figure 2. Each training profile is represented by a count profile $c_n(j,x)$, which specifies the counts of amino acid $x \in \{1,\ldots,20\}$ at position $j \in \{-d,\ldots,d\}$. These counts are obtained by multiplying the sequence profile $t_n(j,x)$ by the effective number of sequences $N_n(j)$ at position j in the alignment from which training profile $t_n(j,x)$ was calculated: $c_n(j,x) = N_n(j)t_n(j,x)$ (see next section for details).

Here, we describe how these N profiles are clustered in order to obtain a set of K *context profiles* which recur frequently among the training profiles and which together can describe all training profiles. More precisely, we seek to determine context profiles $p = (p_1,\ldots,p_K)$ and their prior probabilities $\alpha = (\alpha_1,\ldots,\alpha_K)$ that maximize the likelihood $P(c|p,\alpha)$ that the training profile counts $c = (c_1,\ldots,c_N)$ were generated by the context profiles. We model the distribution of counts $c_n(j,x)$ in each column j by a multinomial distribution. Since $c_n(j,x)$ can be real-valued, however, we replace the factorials in the multinomial distribution by Gamma functions ($n! = \Gamma(n+1)$). The probability for context profile p_k to have emitted counts $c_n(j,x)$ ($j \in \{-d,\ldots,d\}$, $x \in \{1,\ldots,20\}$) is

$$P(c_n|p_k) = \prod_{j=-d}^{d} \left(\frac{\Gamma(N_n(j)+1)}{\prod_{x=1}^{20}\Gamma(c_n(j,x)+1)} \prod_{x=1}^{20} p_k(j,x)^{c_n(j,x)} \right)^{w_j}, \quad [1]$$

were the w_j are weights on the window positions as discussed in section 2 of the main text.

We use the Expectation Maximization (EM) algorithm to find (p^*,α^*) which optimize the likelihood $P(c|p,\alpha)$. In order to make this problem tractable, we introduce hidden variables $z = (z_1,\ldots,z_N)$. Hidden variable $z_n \in \{1,\ldots,K\}$ indicates which context profile p_k has emitted training counts c_n. With this definition, we can solve the following optimization problem:

$$(p^*,\alpha^*) = \operatorname*{argmax}_{p,\alpha} P(c|p,\alpha) = \operatorname*{argmax}_{p,\alpha} \sum_z P(c,z|p,\alpha) \quad [2]$$

Suppose we have obtained values $\tilde{p}$ and $\tilde{\alpha}_k$ in a previous iteration of the EM algorithm. Then the new values are obtained by (M-step)

$$(p,\alpha) = \operatorname*{argmax}_{p,\alpha} Q(p,\alpha)\,. \quad [3]$$

where $Q(p,\alpha)$ is the expectation value of the log likelihood of the data over all possible $z \in \{1,\ldots,K\}^N$:

$$\begin{aligned} Q(p,\alpha) &= \sum_z P(z|c,\tilde{p},\tilde{\alpha}) \log P(c,z|p,\alpha) & [4]\\ &= \mathbb{E}_z[\log P(c,z|p,\alpha)] & [5] \end{aligned}$$

Note that the probability of z is conditioned on the values $(\tilde{p},\tilde{\alpha})$ from the previous iteration and does not depend on (p,α). In order to maximize $Q(p,\alpha)$, we first apply Bayes' Theorem to calculate the probability ditribution over z:

$$P(z|c,\tilde{p},\tilde{\alpha}) = \frac{P(c|z,\tilde{p})P(z|\tilde{\alpha})}{\sum_{z'} P(c|z',\tilde{p})P(z'|\tilde{\alpha})} \propto P(c|z,\tilde{p})P(z|\tilde{\alpha})\,. \quad [6]$$

The first term on the right is obtained by applying eq. (1) to $P(c_n|\tilde{p}_{z_n})$ for all n,

$$P(c|z,\tilde{p}) = \prod_{n=1}^{N}\prod_{j=-d}^{d} \left(\frac{\Gamma(N_n(j)+1)}{\prod_{x=1}^{20}\Gamma(c_n(j,x)+1)} \prod_{x=1}^{20} \tilde{p}_{z_n}(j,x)^{c_n(j,x)} \right)^{w_j}, \quad [7]$$

while the second term on the right side is simply

$$P(z|\tilde{\alpha}) = \prod_{n=1}^{N} \tilde{\alpha}_{z_n}\,. \quad [8]$$

Therefore, we can calculate the probability that training counts c_n were emitted by context profile p_k (E-step):

$$\begin{aligned} &P(z_n = k|c,\tilde{p},\tilde{\alpha})\\ &= \frac{\tilde{\alpha}_k \prod_{j=-d}^{d} \left(\frac{\Gamma(N_n(j)+1)}{\prod_{x=1}^{20}\Gamma(c_n(j,x)+1)} \prod_{x=1}^{20} \tilde{p}_k(j,x)^{c_n(j,x)} \right)^{w_j}}{\sum_{k'=1}^{K} \tilde{\alpha}_{k'} \prod_{j=-d}^{d} \left(\frac{\Gamma(N_n(j)+1)}{\prod_{x=1}^{20}\Gamma(c_n(j,x)+1)} \prod_{x=1}^{20} \tilde{p}_{k'}(j,x)^{c_n(j,x)} \right)^{w_j}}\\ &= \frac{\tilde{\alpha}_k \prod_{j=-d}^{d} \left(\prod_{x=1}^{20} \tilde{p}_k(j,x)^{c_n(j,x)} \right)^{w_j}}{\sum_{k'=1}^{K} \tilde{\alpha}_{k'} \prod_{j=-d}^{d} \left(\prod_{x=1}^{20} \tilde{p}_{k'}(j,x)^{c_n(j,x)} \right)^{w_j}}\,. \end{aligned} \quad [9]$$

Let us now take a closer look at $Q(p,\alpha)$:

$$\begin{aligned} &Q(p,\alpha) = E_z[\log P(c,z|p,\alpha)]\\ &= E_z[\log(P(z|z,p,\alpha)P(z|\alpha))]\\ &= \sum_{n=1}^{N}\sum_{j=-d}^{d} w_j \left(\log\Gamma(N_n(j)+1) - \sum_{x=1}^{20}\log\Gamma(c_n(j,x)+1) \right.\\ &\qquad \left. + \sum_{x=1}^{20} c_n(j,x)\mathbb{E}_z[\log p_{z_n}(j,x)] \right) + \sum_{n=1}^{N}\mathbb{E}_z[\log\alpha_{z_n}]\,. \end{aligned} \quad [10]$$

Here,

$$\mathbb{E}_z[\log p_{z_n}(j,x)] = \sum_{k=1}^{K} P(z_n = k|c,\tilde{p},\tilde{\alpha}) \log p_k(j,x) \quad [11]$$

and

$$\mathbb{E}_z[\log\alpha_{z_n}] = \sum_{k=1}^{K} P(z_n = k|c,\tilde{p},\tilde{\alpha}) \log\alpha_k\,. \quad [12]$$

To maximize $Q(p,\alpha)$ under the constraints

$$\sum_{k=1}^{K}\alpha_k = 1 \quad \text{and} \quad \sum_{x=1}^{20} p_k(j,x) = 1\ \forall k,j \quad [13]$$

we define Lagrange multipliers $\lambda, \mu_{kj} \in \mathbb{R}$ for these constraints:

$$\frac{\partial Q}{\partial\alpha_k} = \sum_{n=1}^{N} \frac{P(z_k|c,\tilde{p},\tilde{\alpha})}{\alpha_k} = \lambda \cdot \underbrace{\frac{\partial}{\partial\alpha_k}\left(\sum_{k=1}^{K}\alpha_k - 1 \right)}_{1}, \quad [14]$$

$$\frac{\partial Q}{\partial p_k(j,x)} = \sum_{n=1}^{N} \frac{w_j c_n(j,x) P(z_n = k|c,\tilde{p},\tilde{\alpha})}{p_k(j,x)}$$
$$= \mu_{kj} \cdot \underbrace{\frac{\partial}{\partial p_k(j,x)} \left(\sum_{x=1}^{20} p_k(j,x) - 1 \right)}_{1} . \quad [15]$$

Solving for α_k and $p_k(j,x)$ and using the normalization constraints in eq. (13) to eliminate λ and the μ_{kj} yields

$$\alpha(k) = \frac{\sum_{n=1}^{N} P(z_n = k|c,\tilde{p},\tilde{\alpha})}{\sum_{k'=1}^{K} \sum_{n=1}^{N} P(z_n = k'|c,\tilde{p},\tilde{\alpha})} \quad [16]$$

$$p_k(j,x) = \frac{\sum_{n=1}^{N} P(z_n = k|c,\tilde{p},\tilde{\alpha}) c_n(j,x)}{\sum_{y=1}^{20} \sum_{n=1}^{N} P(z_n = k|c,\tilde{p},\tilde{\alpha}) c_n(j,y)} \quad [17]$$

These two equations give the recipe for updating model parameters p and α to new values in the M-step of the EM algorithm. In the E-step we use eq. (9) to estimate hidden variables z.

Note that our clustering approach corresponds to a soft clustering of training profiles. Each training profile can be generated by any of the context profiles (see eq. (2)), and hence each context profile has contributions from all training profiles, as can be seen in eq. (17). This kind of clustering is adapted to the intended use of our context profile library, in which we *mix* context profiles according to their posterior probabilities in order to generate pseudocounts, instead of deriving the pseudocounts from the most similar context profile.

In practice, we have found that the EM algorithm converges reliably after 25 iterations for $K = 4000$, running about 100h on a single core of a 2.2 GHz 64-bit Quad-Core AMD Opteron processor. Several independent EM runs yield context profiles whose CS-BLAST sensitivity was within less than 2% of each other.

Effective number of sequences

The effective number of seqences at column i of a multiple alignment is calculated on the subalignment M_i formed by all sequences with a residue in column i and by all columns with at most 10% terminal gaps in these sequences. A terminal gap is a gap that lies either to the left or to the right of the entire sequence. For each column j of M_i we calculate amino acid frequencies $p(j,x)$, using the Hennikoff sequence weighing scheme. Then the number of effective sequences

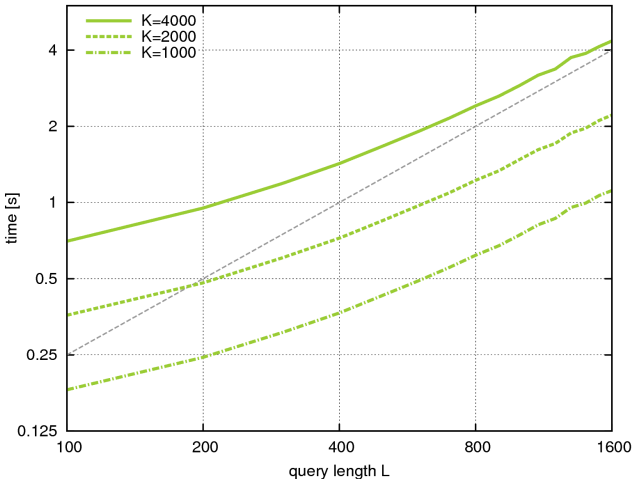

Fig. 1. Time required for generating the context-specific profile in CS-BLAST depending on query length L and size of context library $K = 1000, 2000, 4000$ (window length $l = 13$). The plot verifies that the runtime T scales roughly with $T \propto KlL$. To limit the runtime of the profile generation in CS-BLAST for a typical protein with 200 residues to $\sim 1s$, we choose $K = 4000$.

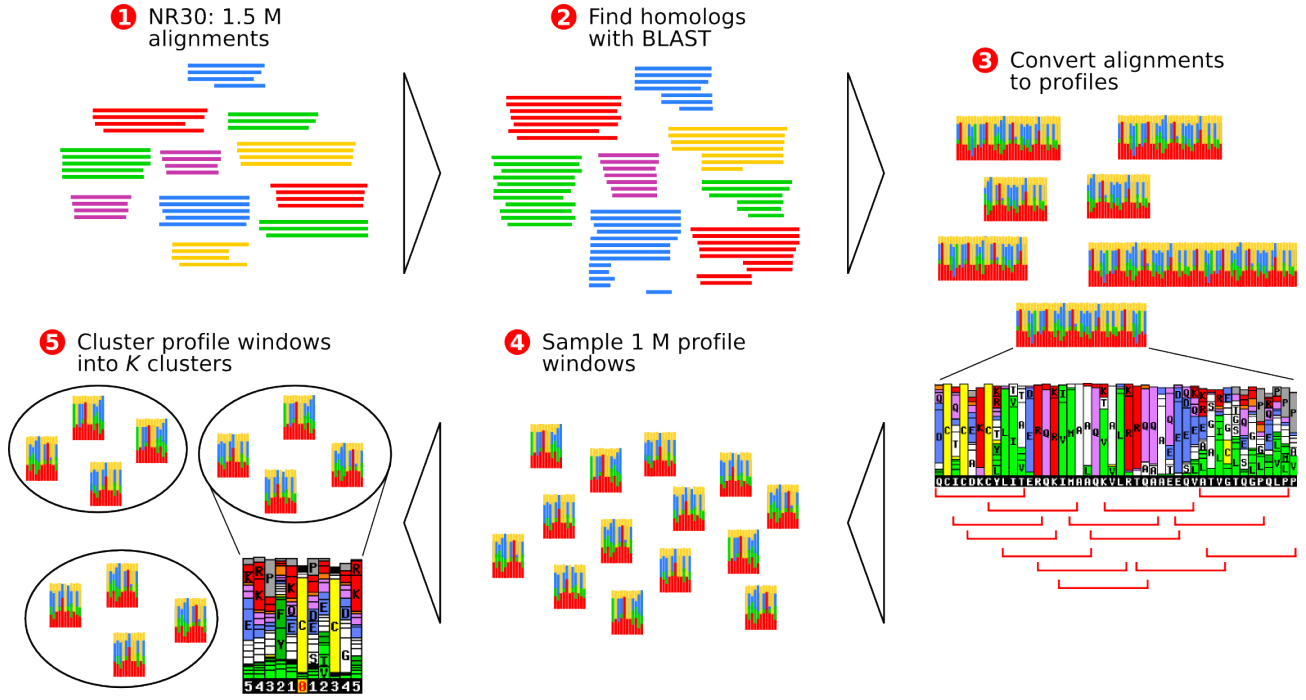

Fig. 2. Computation of the library of context profiles representing local sequence contexts. From a database (NR30) of 1.5M groups of aligned sequences covering the NR database, we select the 50 000 most diverse alignments and enrich these with homologs from a BLAST search. The alignments are converted to sequence profiles and 1 Mio. profile windows are randomly sampled and used to train K context profiles ($K = 500, 1000, 2000, 4000$) with the expectation maximization algorithm.

is

$$N_n(i) = \exp\left(-\frac{1}{L_i}\sum_{j\in M_i}\sum_{x=1}^{20} p(j,x)\log\ p(j,x)\right) . \quad [18]$$

Here, L_i is the number of columns in M_i.

Parameter optimization

Our context-specific search tools CS-BLAST and the EM clustering procedure for generating the library of context profiles (see section about generation of context profiles) contain several adjustable parameters: The number of context profiles K, the profile window length l, the positional weight parameters w_{center} and β, and the pseudocount admixture τ. We optimize these parameters using a homology detection benchmark that is similar to the benchmark utilized to compare CS-BLAST with BLAST (see main text for details). The optimization runs, however, are performed on a small *optimization set* (1329 domains) that is distinct from the comprehensive *test set* (5287 domains) with no optimization set member sharing a common fold with any test set member. For a given parameter configuration ($K, l, w_{\text{center}}, \beta$, and τ) we first generate a library of K context profiles by EM clustering, then we perform an all-against-all comparison of the training-set domains and count the true positive (TP) and false positive (FP) hits at various E-value thresholds. From this we can infer a ROC5 score ($\in [0,1]$) for each query. The ROC5 score is defined as the area under the TP-versus-FP ROC curve (receiver operating statistic) up to the fifth false positive hit, divided by the area under the optimal ROC curve. To assess the overall homology detection performance, one can plot the fraction of queries with ROC5 scores above a variable ROC threshold ($\in [0,1]$). The area under such a curve, the mean ROC5 score, conveniently captures the homology detection performance in a single value and is therefore used as performance index in the parameter optimization. Optimization results by mean ROC5 score proved to be more robust and less noisy than results obtained by optimizing the overall sensitivity at a given error rate.

Ideally, we would like to compute mean ROC5 scores for a wide range of different parameter configurations. However, due to the considerable runtime of the EM clustering algorithm we have to restrict our optimization runs to a selected set of reasonable parameter settings. Parameter values for window size l and library size K result from a compromise between sensitivity and time efficiency in CS-BLAST. To limit the runtime of the profile generation in CS-BLAST for a typical protein to about $1s$, we choose $K = 4000$ and $l = 13$ (see Figure 1). With K and l fixed, we benchmark all 60 possible parameter combinations for $w_{\text{center}} \in \{1.3, 1.6, 2.0, 2.5\}$, $\beta \in \{0.8, 0.85, 0.9\}$, and $\tau \in \{0.8, 0.85, 0.9, 0.95, 1.0\}$. The optimization runs reveal that parameter setting $w_{\text{center}} = 1.6$, $\beta = 0.85$, and $\tau = 0.9$ gives the best mean ROC5 score of 0.2374. Around this optimal configuration we test the effect of different values for cluster size ($K = 500, 1000, 2000, 4000$) and window length ($l = 9, 11, 13, 15$). The results suggest that the homology detection performance improves not only with the number of context profiles K, as expected, but also with the length of the context window l. However, a subsequent evaluation of window lengths parameters $l = 13$ and $l = 15$ on the benchmark set revealed almost identical performance for both settings. We therefore chose $l = 13$ for the benefit of faster runtime. Figure 3A-E illustrates the effect of varying one parameter at a time while keeping the others fixed at the optimum configuration ($K = 4000, l = 13, w_{\text{center}} = 1.6, \beta = 0.85, \tau = 0.9$).

In addition to CS-BLAST parameters ($K, l, w_{\text{center}}, \beta, \tau$), we also need to optimize pseudocount admixture parameters a and b for CSI-BLAST (see methods section in main text). We employ the same benchmark procedure as described above, but perform three rounds of CSI-BLAST instead of CS-BLAST. The last search is performed against our training database; all previous iterations use the full NR database. Since we already know the optimal pseudocount admixture for the one-sequence case from the optimization of τ in CS-BLAST, we set a to 0.9. Benchmark runs for $b = 6, 8, 10, 12, 16, 20$ reveal that $b = 12$ gives the best results and is therefore the default setting in CSI-BLAST (Figure 3F).

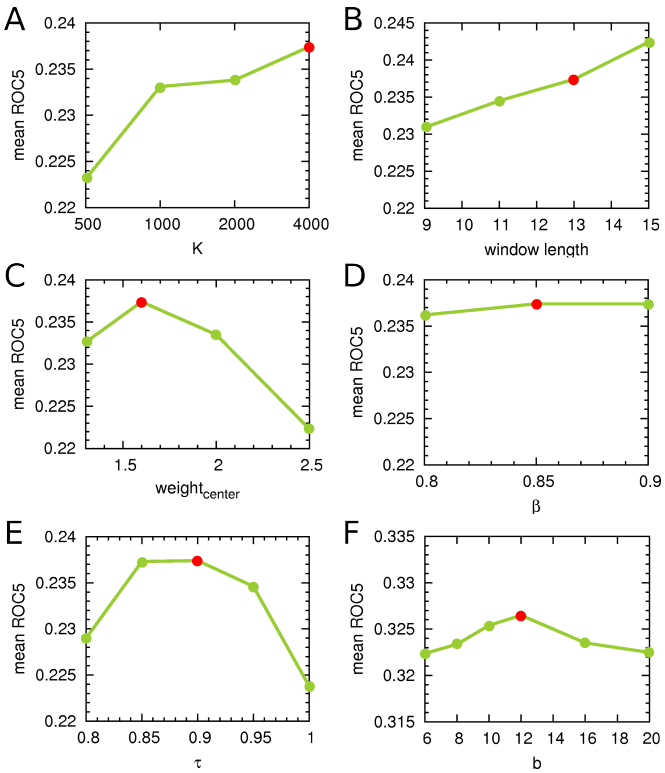

Fig. 3. Optimization of parameters $K, l, w_{\text{center}}, \beta, \tau$ and CSI-BLAST parameter b. The plots illustrate the effect of varying one parameter at a time while keeping the others fixed at the optimal configuration $K = 4000, l = 13, w_{\text{center}} = 1.6, \beta = 0.85$, and $\tau = 0.9$. Red circles indicate the optimal parameter value that is chosen as default in CS-BLAST/CSI-BLAST. **A** Number of context profiles K. **B** Window length l. **C** Weight of central profile column w_{center}. **D** Weight decay parameter β. **E** Pseudocount admixture τ. **F** Pseudocount extinction parameter b in CSI-BLAST ($a = 0.9$). Three iterations of CSI-BLAST were performed for this optimization. Note the different y-scale.