Sequence alignment kernel for recognition of promoter regions

Leo Gordon1,*, Alexey Ya. Chervonenkis1,2, Alex J. Gammerman1, Ilham A. Shahmuradov1 and Victor V. Solovyev3

1Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK, 2Institute of Control Science, Profsoyuznaja 65, Moscow, Russia and 3Softberry Inc., 116 Radio Circle, Suite 400, Mount Kisco, NY, 10549, USA

Received on December 6, 2002; revised on March 2, 2003; accepted on April 29, 2003

ABSTRACT
In this paper we propose a new method for recognition of prokaryotic promoter regions with startpoints of transcription. The method is based on Sequence Alignment Kernel, a function reflecting the quantitative measure of match between two sequences. This kernel function is further used in Dual SVM, which performs the recognition.

Several recognition methods have been trained and tested on positive data set, consisting of 669 σ70-promoter regions with known transcription startpoints of Escherichia coli and two negative data sets of 709 examples each, taken from coding and non-coding regions of the same genome. The results show that our method performs well and achieves 16.5% average error rate on positive & coding negative data and 18.6% average error rate on positive & non-coding negative data.

Availability: The demo version of our method is accessible from our website http://mendel.cs.rhul.ac.uk/
Contact: leo@cs.rhul.ac.uk

INTRODUCTION
Promoter region is an area on the chromosome which determines where the transcription of a particular gene(s) should be initiated and on what conditions. In prokaryotic organisms the promoter region occupies several hundred base pairs upstream of Transcription Start Site (TSS) and a smaller area downstream of TSS, and may serve for transcription of a single gene as well as for a group of genes (an operon).

During the last five years many prokaryotic genomes have been sequenced, including that of Escherichia coli (Blattner et al., 1997). The gene content of these genomes was mostly computationally recognized (which is usually in good agreement with later lab experiments). However, the promoter regions and TSS are still undetermined in most cases and the software able to accurately predict promoters in sequenced genomes is not yet available in public domain. Promoter recognition, the computational task of finding the promoter regions on a DNA sequence, is very important for defining the transcription units responsible for specific pathways (because gene prediction alone cannot provide the solution) and for analysis of gene/operon regulation.

Experimental studies of prokaryotic promoter regions show that they contain a certain set of binding motifs—relatively short chunks of DNA to which the RNA polymerase and special regulatory proteins can bind in order to initiate and control the transcription (De Haseth et al., 1998). The majority of prokaryotic promoter recognition approaches use the binding motif based model of promoter region, which emphasizes the importance of binding motifs and essentially neglects the data between them (spacers). Figure 1 shows such a model for prokaryotic σ70 promoter region with two binding motifs and two spacers.

Because certain variation in the lengths of the spacers and informational content of each binding motif is allowed, the search for the ‘right’ placement of binding motifs is not completely trivial.

Different general approximate matching techniques, as well as the ones, specifically designed for this task, are used for binding motif search (Crochemore and Sagot, 2002, and references therein). Various approaches based on weight matrices (Staden, 1984; Harley and Reynolds, 1987; Mulligan and McClure, 1986), neural networks (Lukashin et al., 1989; Demeler and Zhou, 1991), (O’Neill, 1991, 1992; Horton and Kanehisa, 1992; Mahadevan and Ghosh, 4For example, so-called -10-box of σ70 bacterial promoter may look like TATAAT, CATATA, TATAAA, etc.
Our method is preferable in cases when we have a sufficient number of known promoter regions, but might not know anything about their composition.

One of the findings of this approach is that in spite of generality the developed kernel has outperformed in accuracy of the prediction several other known approaches, which suggests that the information ‘between the boxes’ might also be important for recognition.

**PROBLEM STATEMENT**

We will treat genome as a string $S$ composed of letters $\{A,C,G,T\}$. On it some specific positions called TSS are given. We assume that, given a position $p$, a region $S_{p-U} \cdots S_{p+D}$ contains enough information to distinguish whether $p$ is a TSS or not. We will call such a region a (potential) promoter region.

We are given a training set composed of positive examples (‘true’ promoter regions) and negative examples (‘false’ promoter regions). Our goal is, given an arbitrary potential promoter region to be able to find out whether it is ‘true’ or ‘false’ promoter region.

**ALGORITHM: SEQUENCE ALIGNMENT KERNEL**

Our method is based on building the kernel function $K(R, Q)$ as a quantitative measure of similarity between two sequences $R$ and $Q$. Such a function should be suitable for classification by Dual SVM (Vapnik, 1998) or any other kernel-based classification method.

Suppose we are given a matrix Swap$(x, y)$ which defines the score corresponding to a single point mutation of letter $x$ into letter $y$ or vice versa (the matrix is symmetric). We are also given a vector Gap$(x)$ which defines the score corresponding to a single point deletion or insertion of letter $x$.

One of the schemes for simultaneous generation of two sequences over a given alphabet was proposed by Watkins (2000). The generative model may emit either two letters (one into each sequence), only one letter into the first sequence (which corresponds to a gap into the second one), or only one letter into the second sequence (which corresponds to a gap into the first one). The model is completely defined by the probabilities for each pair it may emit. For any two non-empty sequences there are several ways (or paths) to generate them using this model. For every such path the corresponding probability is the product of probabilities along the path. The total probability $P(x, y)$ that the sequences $x$ and $y$ will be generated by the model is the sum of probabilities of all the paths that lead to generating the given pair. Watkins (2000)

---

There are still not enough examples from other classes of *E. coli* promoters like $\sigma^{30}, \sigma^{54}$, etc.—for statistical analysis.

Theoretically, they do not have to belong to the genome and can be any strings of $U + D$ letters from $\{A,C,G,T\}$.
has proven that the function $P(x, y)$ is symmetric and positively definite, and so may be used as a kernel for SVM and other kernel-based algorithms.

If we take the Swap$(x, y)$ matrix and Gap$(x)$ vector to be the logarithms of the probabilities from Watkins’ model, then the classical Global Alignment algorithm by Needleman–Wunsch (Needleman and Wunsch, 1970) can be regarded as a method to calculate the probability of the most probable path to generate the two sequences. However, it has not been proven that the alignment score it provides is non-negatively definite [one of the necessary conditions for a kernel function to be valid, see Vapnik (1998)].

Straightforward summation of all paths’ probabilities would need exponential time. We would also need to add together a big number of very small values, which might suffer from floating point arithmetic underflow. Following the dynamic programming ideas used in Global Alignment, an algorithm was proposed (Surkov et al., 2001) for fast and precise calculation of the kernel function $P(x, y)$.

The algorithm

Suppose we are given two sequences to align, $Q = ‘ACGT’$ and $R = ‘ACGT’$. Let us write them along the two dimensions of an empty matrix (Fig. 2).

In each cell $p_{i,j}$ of the matrix we will be keeping the probability that $Q_{1..j}$ aligns with $R_{1..j}$. It is convenient to start the calculations from the bottom left corner, which is initialized with the value of 1. Then, we fill all the other cells using the recursive formula:

$$
\begin{align*}
    p_{0,0} & = 1, \\
    p_{i,0} & = p_{0,j} = 0, \quad \text{for } i > 0 \text{ and } j > 0, \\
    p_{i,j} & \leftarrow \text{Swap}(R_i, Q_j) \cdot p_{i-1,j-1} \\
    & + \text{Gap}(Q_j) \cdot p_{i,j-1} \\
    & + \text{Gap}(R_i) \cdot p_{i-1,j},
\end{align*}
$$

where the Swap$(x, y)$ matrix and the Gap$(x)$ vector of probabilities are given as parameters to the algorithm. The kernel value we are looking for is the probability $K = p_{|R|,|Q|}$ in the top right corner of the matrix.

Note, that to calculate values on any ‘backslash’ diagonals of the $p$ matrix $(i + j = D)$ we only need to know the values on the two preceding diagonals: $i + j = D - 1$ and $i + j = D - 2$. This property is used to speed up the calculations. If we ‘turn’ the matrix $p$ by 45°, the diagonals become rows and columns, yielding $O(\max(|R|, |Q|)^2)$ time complexity and $O(\max(|R|, |Q|))$ space complexity.

Allowing for affine gaps

A more complicated version of this algorithm accounts for ‘affine gaps’ (Gotoh, 1982)—it means that in a run of gaps, the one starting the run may be given a different probability than the gaps extending the run. This is attained by using a more sophisticated computational scheme, a three-layer matrix (Fig. 3, for a close-up see Fig. 4) instead of one layer described above.

On the $d$-level only diagonal transitions are allowed, which correspond to substitutions. $H$-level is used for ‘horizontal’ transitions only, it accounts for gaps in $Q$. $V$-level is used for ‘vertical’ transitions only, it accounts for gaps in $R$. The end result is the probability $K = p_{|R|,|Q|}^d$. The core idea of this stratification is to multiply each transition from the main, $d$-layer to any of the two ‘gap layers’ by an additional probability coefficient, StartGap. If StartGap = 1, there is no difference with the original scheme. But if StartGap < 1, it is equivalent to paying an additional penalty in order to start the gap.

$$
\begin{align*}
    p_{0,0}^d & = 1, \\
    p_{i,0}^d & = p_{i,0}^b = 0, \quad \text{for } i > 0 \text{ and } j > 0, \\
    p_{i,j}^d & \leftarrow \text{Gap}(Q_j) \cdot (p_{i,j-1}^d \cdot \text{StartGap}), \\
    p_{i,j}^b & \leftarrow \text{Gap}(R_i) \cdot (p_{i-1,j}^d + p_{i-1,j-1}^d \cdot \text{StartGap}), \\
    p_{i,j}^h & \leftarrow \text{Swap}(R_i, Q_j) \cdot (p_{i-1,j}^d + p_{i-1,j-1}^d + p_{i-1,j-1}^d \cdot \text{StartGap}).
\end{align*}
$$

The stratification has a slow-down impact on the performance,7 but gives more flexibility to the system.

In addition to what is described above, we propose the following:

1. Take the root of power $|R| + |Q|$ from each kernel value—this gives us a close approximation to the similarity measure ‘per symbol’ (or rather ‘per gap’), and makes the kernel values relatively independent of sequence lengths $|R|$ and $|Q|$.

7In fact, both the execution time and the memory needs increase in a constant number of times—this does not affect the complexity of the algorithm.
Sequence alignment kernel for pro-motor regions

A

C

C

C

T

T

A

A

C

C

G

G

T

T

C

C

1

K(R,Q)

h-layer

d-layer

v-layer

Fig. 3. ‘3D’ version of sequence alignment, which accounts for affine gaps.

Fig. 4. A local fragment of the ‘3D’ version of sequence alignment.

2. After the whole matrix of kernel values \( K \) has been found, normalize the matrix:

\[
K'(X, Y) = \frac{K(X, Y)}{\sqrt{K(X, X) \cdot K(Y, Y)}}
\]

Because a kernel corresponds to dot product in some imaginary feature space between two vectors, by this normalization we get rid of the ‘lengths’ of the vectors, so that only the cosine of the angle between them remains.

3. Then take each element to some fixed power \( \alpha > 1 \); this gives us one more convenient parameter to tune.\(^8\)

The need for this arises when the vectors are ‘too much different’ from each other, so that after the normalization we get ones on the diagonal, \( K'(X, X) = 1 \), and nearly zeros everywhere else.

\(^8\)We are unaware of a theoretical proof that this operation is valid for kernel functions in general, but in our experiments involving generating hundreds of kernel matrices every single one was positively definite.

To test this method on the problem of prokaryotic promoter recognition, the following parameter values were found to be optimal: \( \text{StartGap} = 0.05, \alpha = 1.3 \). We obtained \( \text{Swap}(x, y) \) matrix and \( \text{Gap}(x) \) vector by exponentiation \( \text{Sankoff-76 transition/transversion score matrix} \) (Sankoff et al., 1976), which happened to work well applied to our problem (see Table 1).

### COMPARISON OF DIFFERENT METHODS

Sequence Alignment Kernel was used in conjunction with Dual Support Vector Machine (Vapnik, 1998). This method was tested among several different promoter region predicting methods.

#### The data

As the primary source of data we used the sequenced genome of \( E.\text{coli} \) (strain K-12, substrain MG1655) Blattner et al. (1997). A list of 669 experimentally confirmed \( \sigma^{70} \) promoters with known TSS positions was put together from RegulonDB [Salgado et al. (2000); http://www.cifn.unam.mx/Computational_Genomics/regulondb/ and PromEC (Hershberg et al., 2001); http://bioinfo.md.huji.ac.il/marg/promec/] databases. Then promoter regions \( [\text{TSS} – 60 \cdots \text{TSS} + 19] \) were taken as the positive examples.

As there is no experimentally confirmed negative data (i.e. the positions that are confirmed not to be TSS), we had to take the risk and choose the negative examples randomly from the same chromosome. Approximately 81% of known TSS are located in the intergenic non-coding regions and 19% in the coding regions. So two different negative example sets were prepared:

(a) coding negative example set containing 709 sub-sequences, 80 letters each, from the coding regions (genes) and

(b) non-coding negative example set containing 709 sub-sequences, 80 letters each, from the non-coding regions (intergenic spacers).

The hypothetical non-TSS in both sets of examples is located in the 61st position, so the examples have the same format as the positive ones: \([\text{nonTSS} − 60 \cdots \text{nonTSS} + 19] \).

The same data points were used when testing all the other methods.

<table>
<thead>
<tr>
<th>Swap(x, y)</th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1.0000</td>
<td>0.1738</td>
<td>0.3679</td>
<td>0.1738</td>
</tr>
<tr>
<td>C</td>
<td>0.1738</td>
<td>1.0000</td>
<td>0.1738</td>
<td>0.3679</td>
</tr>
<tr>
<td>G</td>
<td>0.3679</td>
<td>0.1738</td>
<td>1.0000</td>
<td>0.1738</td>
</tr>
<tr>
<td>T</td>
<td>0.1738</td>
<td>0.3679</td>
<td>0.1738</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

| Gap(x)    | 0.1054 | 0.1054 | 0.1054 | 0.1054 |

Table 1. Optimal \( \text{Swap}(x, y) \) matrix and \( \text{Gap}(x) \) vector used for \( \sigma^{70} \) prokaryotic promoter recognition.

1967
Other methods

Here we briefly list the methods that we used to compare with our Sequence Alignment Kernel based method. In all methods except the first two we used SVM in the last stage, simple or kernel-based.

- BLAST-based Nearest Neighbours method (Altschul et al., 1997) is simply a Nearest Neighbours classifier where the distance is defined by pairwise BLASTn E-value. On each iteration the training set is converted into BLAST-compatible database, then each test example is looked up in the database to find the nearest match.

- Boxes + threshold method (Staden, 1984; Harley and Reynolds, 1987) is probably the best known technique for automatic motif discovery. Here a hypothesis is made that σ−motif promoter regions contain but two important binding motifs on relatively fixed positions (see Fig. 1). Every potential promoter area is converted into four numerical features (matching the scores for the −10′ and −35′ binding motifs and likelihoods of the two distances, from TSS to −10′ motif and −10′ motif to −35′ motif). Then the four are added together, forming the ‘general’ likelihood. The optimal threshold value for it is found on the training set, and then used to classify the test set examples.

- Boxes + SVM method is very similar to the previous one, but the four likelihoods are not added. They are used as four independent features in the standard SVM routine. Simply speaking, SVM finds the best combination of those features, of which the sum is but one, so it is easily explainable, why SVM-based methods generally perform better.

- Boxes and regulatory sites + SVM method (Bailey and Elkan, 1995) is based on the previous method, where three additional features are generated to each example from CRP, IHF and LexA regulatory sites.

The resulting 4 + 3 features were used in the standard SVM routine.

- Zone likelihood + SVM method (Oppon and Hide, 1998) uses the hypothesis that promoter region has a distribution of oligonucleotides (short substrings of a given length) which is different from the overall distribution in the DNA. Unlike Oppon and Hide (1998), we suspected that different zones inside the promoter region have different distributions. We have found five zones, whose local distributions well predicted the promoter regions.\(^\text{10}\)

The score was computed for every such zone and the resulting five features were used in the standard SVM routine.

- Locality-improved kernel + SVM method (Schölkopf et al., 1998) was originally applied to Translation Initiation Sites prediction (Zien et al., 2000, locality-improved kernel), but as it suits relatively general purpose of classifying DNA regions, we applied it to prediction of promoter regions.\(^\text{11}\)

This method directly yielded the kernel function which was used in the kernel-based SVM routine.

Criteria and results of comparison

In every experiment the positive examples and the negative examples were mixed together, 1/2 of them were randomly chosen to serve as training and the other 1/2—as test examples. Then the recognition program was executed. After each execution four numbers were calculated:

- TP, true positives, #(correctly recognized positives),
- TN, true negatives, #(correctly recognized negatives),
- FN, false negatives, #(positives recognized as negatives),
- FP, false positives, #(negatives recognized as positives).

For every method, 100 executions were performed. Then the average TP, TN, FN and FP were found and the following relative measures were calculated:

\[
\begin{align*}
\text{TP} &= \frac{TP}{TP + FN} \times 100, \\
\text{FP} &= \frac{FP}{TN + FP} \times 100, \\
\text{AE\%} &= \text{(average error)} = \frac{(FN + FP)}{(TP + TN + FP + FN)} \times 100, \\
\text{Sn\%} &= \text{(sensitivity or recall)} = \frac{TP}{TP + FN}, \\
\text{Sp\%} &= \text{(specificity or precision)} = \frac{TN}{TP + FP}, \\
\text{CC}\% &= \frac{\text{TP} \times TN \times FP \times FN}{\sqrt{(TP + FP) \times (TN + FN) \times (TP + FN) \times (TN + FP)}}.
\end{align*}
\]

The Table 2 shows that Sequence Alignment Kernel based method outperformed other tested methods on average error and false negatives. This result is important, since it is clear that Sequence Alignment Kernel does not use any prior information as to what chunks of the promoter region are important and what are not. Obviously, some information ‘between the boxes’ is also important and needs more attention.\(^\text{12}\)

\(^{\text{9}}\)For full description see (Gordon, 2002).

\(^{\text{10}}\)Relative to the TSS they are: (i) [−71⋅⋯⋅−61] in mononucleotides, (ii) [−60⋅⋯⋅−41] in mononucleotides, (iii) [−40⋅⋯⋅−35] in pentanucleotides, (iv) [−17⋅⋯⋅−8] in hexanucleotides, (v) exact position of TSS in mononucleotides.

\(^{\text{11}}\)The parameters that were found to give the best prediction were: \(l = 2, d_1 = 4, d_2 = 3\).

\(^{\text{12}}\)It is interesting to note that, although boxes & regulatory sites method is quite close to Sequence Alignment Kernel when tested on the positive examples and coding negative example set, on the set composed of positive examples and non-coding negative example set it seriously falls behind. We think it happens because the regulatory sites are well spread in the non-coding regions, so they affect both positive and negative examples alike.
neural network for prediction of E.coli promoters, dividing them into three classes with 16, 17 and 18 bases separating −35 and −10 regions. Overall 77% were correctly identified, but the level of false positives was very high. Later, this work was continued (O’Neill, 1991, 1992) and the following prediction accuracies were achieved: 78–100% for 16 bp spacing, 97% for 17 bp spacing and 79% for 18 bp spacing.

Another paper (Demeler and Zhou, 1991) describes ‘neural network optimization for E.coli promoter prediction methods’. A neural network was trained on a set of 80 known E.coli promoter sequences and a different number of random sequences. The prediction accuracy of the resulting weight matrix was tested against a separate set of 30 known promoter sequences and 1500 random sequences with equal composition of A, C, G and T bases. Accuracies of 100% on promoters and 98.4% on the random sequences were achieved with optimal parameters. However, these figures could have been very much affected by the choice of data. First, both training and test sets were very small. Second, because of the way the negative examples were generated, the promoter search protocol is likely to be highly sensitive to the average A/T ratio of the input due to A/T relative richness of promoters (Mulligan and McClure, 1986; O’Neill and Chiafari, 1989).

Horton and Kanehisa (1992) reported ‘perceptron type neural network for prediction of E.coli σ70 promoters’. Moreover, they reconstructed five previously reported methods and compared the quality of prediction of these methods and their own approach on the same data sets. Although prediction accuracy in previous reports (Demeler and Zhou, 1991; O’Neill, 1991, 1992) was very high, training and testing of perceptron type neural network in the same data gave comparable results. The difference in prediction rates with these different data sets seems to be explainable to a large extent by the differences in information content of the combined training and test sets’ (Horton and Kanehisa, 1992); both of previously used data sets were essentially subsets of the one used by Horton and Kanehisa. In particular, O’Neill’s data only contained promoters with 17 bp spacing.

Later, Mahadevan and Ghosh (1994) reported on using neural networks trained on 106 promoters and random sequences which were 60% A/T rich and tested on 126 promoters for recognition of E.coli promoters of all spacing classes (15–21 bp). This network showed 98% accuracy on promoters and 90.2% accuracy on non-promoters (tested on 500 randomly generated sequences).

At last, Leung et al. (2001) presented ‘basic gene grammars and DNA-ChartParser for language processing of E.coli promoter DNA sequences approach’. The method was tested on 300 E.coli promoters and 300 non-promoter random sequences. Four experiments, performed using different ‘grammar rules’, have yielded the best prediction of 76% accuracy with 82% specificity and 69% sensitivity.

It should be noted that the set of known E.coli σ70 promoters that we used both for training and test is the largest and includes the previously used ones. Horton and Kanehisa (1992) have noted the drop in prediction accuracy with larger training and/or test data sets. Taking into consideration the tendency observed, our sequence alignment kernel gives quite comparable results. Moreover, in some of the experiments mentioned above random sequences were used as negative examples. We believe it is not quite fair, as they may differ too much from the actual genomic sequences. The negative examples we are using are not only from the same genome—they are from non-coding regions, to avoid the distributional ‘hints’ of the coding areas.

The future research would include reconstruction of other published methods, then training and testing them on our expanded data set. There is also a direction we would like to undertake to assess the ‘trustworthiness’ of our predictions, by using confidence and credibility measures for each individual prediction (Gammerman and Vovk, 2002).

**ACKNOWLEDGEMENTS**

The authors are grateful to Heladia Salgado, member of RegulonDB (Salgado et al., 2000 http://www.cifn.unam.mx/Computational_Genomics/regulondb/) database support group and Jose Carlos Gonzalez of Universidad Politecnica de Madrid (UPM) for useful discussions and

**Table 2. Different methods compared, results averaged over 100 executions**

<table>
<thead>
<tr>
<th>Method</th>
<th>AE%</th>
<th>FN%</th>
<th>FP%</th>
<th>Sn</th>
<th>Sp</th>
<th>CC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence Alignment Kernel + SVM</td>
<td>16.5</td>
<td>18.5</td>
<td>14.6</td>
<td>0.82</td>
<td>0.84</td>
<td>0.67</td>
</tr>
<tr>
<td>Boxes + SVM</td>
<td>19.1</td>
<td>23.6</td>
<td>14.8</td>
<td>0.76</td>
<td>0.83</td>
<td>0.62</td>
</tr>
<tr>
<td>Boxes + threshold</td>
<td>19.5</td>
<td>24.4</td>
<td>14.8</td>
<td>0.76</td>
<td>0.83</td>
<td>0.61</td>
</tr>
<tr>
<td>Zone likelihood + SVM</td>
<td>21.0</td>
<td>32.2</td>
<td>10.4</td>
<td>0.68</td>
<td>0.86</td>
<td>0.59</td>
</tr>
<tr>
<td>Locality-improved kernel + SVM</td>
<td>22.5</td>
<td>33.1</td>
<td>12.5</td>
<td>0.67</td>
<td>0.84</td>
<td>0.56</td>
</tr>
<tr>
<td>Boxes &amp; regulatory sites + SVM</td>
<td>19.3</td>
<td>24.9</td>
<td>14.1</td>
<td>0.75</td>
<td>0.83</td>
<td>0.62</td>
</tr>
<tr>
<td>Blast-based Nearest Neighbours</td>
<td>23.5</td>
<td>38.8</td>
<td>9.1</td>
<td>0.61</td>
<td>0.86</td>
<td>0.55</td>
</tr>
<tr>
<td>Neighbours</td>
<td>16.8</td>
<td>22.7</td>
<td>11.3</td>
<td>0.77</td>
<td>0.87</td>
<td>0.67</td>
</tr>
<tr>
<td>Neighbours</td>
<td>30.3</td>
<td>25.7</td>
<td>34.6</td>
<td>0.74</td>
<td>0.67</td>
<td>0.40</td>
</tr>
<tr>
<td>Blast-based Nearest</td>
<td>34.6</td>
<td>40.9</td>
<td>28.7</td>
<td>0.59</td>
<td>0.66</td>
<td>0.31</td>
</tr>
<tr>
<td>Neighbours</td>
<td>35.4</td>
<td>40.9</td>
<td>30.2</td>
<td>0.59</td>
<td>0.64</td>
<td>0.29</td>
</tr>
</tbody>
</table>

Upper rows: negative data is taken from coding regions (bold shows best values). Lower rows: negative data is taken from non-coding regions (italics show best values).
REFERENCES


