Bioinformatics, 33(17), 2017, 2642–2650
doi: 10.1093/bioinformatics/btx295
Advance Access Publication Date: 5 May 2017
Original Paper

OXFORD

Genome analysis

An efficient graph kernel method for non-coding RNA functional prediction

Nicolò Navarin[1] and Fabrizio Costa[2,3,*]

[1]Department of Mathematics, University of Padova, Padova 35121, Italy, [2]Department of Computer Science, University of Freiburg, D-79110 Freiburg, Germany and [3]Department of Computer Science, University of Exeter, Exeter EX4 4QF, UK

*To whom correspondence should be addressed.

Associate Editor: Alfonso Valencia

Received on September 13, 2016; revised on March 28, 2017; editorial decision on April 29, 2017; accepted on May 4, 2017

Abstract

Motivation: The importance of RNA protein-coding gene regulation is by now well appreciated. Non-coding RNAs (ncRNAs) are known to regulate gene expression at practically every stage, ranging from chromatin packaging to mRNA translation. However the functional characterization of specific instances remains a challenging task in genome scale settings. For this reason, automatic annotation approaches are of interest. Existing computational methods are either efficient but non-accurate or they offer increased precision, but present scalability problems.

Results: In this article, we present a predictive system based on kernel methods, a type of machine learning algorithm grounded in statistical learning theory. We employ a flexible graph encoding to preserve multiple structural hypotheses and exploit recent advances in representation and model induction to scale to large data volumes. Experimental results on tens of thousands of ncRNA sequences available from the *Rfam* database indicate that we can not only improve upon state-of-the-art predictors, but also achieve speedups of several orders of magnitude.

Availability and implementation: The code is available from http://www.bioinf.uni-freiburg.de/~costa/EDeN.tgz.

Contact: f.costa@exeter.ac.uk

1 Introduction

In the early 2000s, a systematic analysis of transcription in human cells (Willingham and Gingeras, 2006) found a significant discrepancy between the observed transcriptional activity and the activity predicted for protein-coding genes. It was shown that up to 90% of the genome was being transcribed, but that only a minor portion of RNA transcripts (1.5%) was encoding for protein open reading frames. This finding was suggesting the presence of a 'hidden layer' of regulatory elements within the human and other eukaryal genomes: the set of such RNA sequences was termed non-coding RNAs (ncRNAs). Today it is known that there exist many types of ncRNAs that have catalytic functions (like enzymes) or that play key roles in both normal cellular processes and disease states. mapping ncRNAs with The functional annotation of ncRNAs, either by *in silico* or by experimental approaches, has since become a fundamental task in bioinformatics and biology. The identification of ncRNAs is however a harder task than gene identification since one cannot rely on the presence of strong statistical signals such as protein open reading frames. In addition, the conservation of sequence information in ncRNAs is subject to a lower evolutionary pressure than in proteins. A significant component of the functionality of a ncRNA is in fact due to its folding structure (Tinoco and Bustamante, 1999). As a consequence ncRNAs evolve with a characteristic substitution pattern that preserves base-pair interactions, resulting in compensatory double substitutions (e.g. AU into GC) and compatible single substitutions (e.g. AU into GU). To tackle the problem of ncRNA identification and structural characterization we can however exploit both curated resources and experimentally determined

secondary structure collections. The European Bioinformatics Institute currently maintains the RFam database (Burge *et al.*, 2013) that gathers information about several thousands ncRNA families defined on the basis of a shared common ancestor. Protocols like SHAPE (Wilkinson *et al.*, 2006) can then be exploited to improve structure prediction tools (Deigan *et al.*, 2009) as they offer, for a single RNA sequence, *in vitro* evidence for the binding state of each nucleotide. Although currently no technique exist that can extract all functionally active ncRNA elements in a cell, there are protocols, like hiCLIP (Sugimoto *et al.*, 2015), that can identify double stranded RNAs at transcriptome level. Crucially this information can be used to uncover trans-acting regulation of ncRNAs and cis-regulatory motives.

The amount of information on functional RNA structures will likely keep on growing at ever increasing speed. There is therefore a need for computational methods that can help characterize and organize ncRNAs at large and very large scale. In this article, we consider the problem of building an *in silico* classifier to automatically annotate a large set of putative ncRNA sequences. Tackling this problem requires to address several key issues, among which that of (i) efficiency, (ii) flexibility and (iii) robustness, that we detail in the following.

We say that an approach is *efficient* if it can exploit large numbers (from thousands to hundreds of thousands) of RNA sequences annotated with structural information in training and if it can be applied to genomes or transcriptomes in their entirety.

We say that an approach is *flexible* if it can make use of independent sources of information on ncRNAs structural properties. In particular, it should be possible to model secondary structure information derived from *in vivo* or *in vitro* experiments and/or derived from computational approaches. Flexible solutions should ideally be able to accommodate multiple structural hypotheses, uncertainty on the hypothesis and complex information such as the presence of pseudo-knots.

Finally, we say that an approach is *robust* if it can cope with uncertainty on the ncRNA specification, in particular it should be able to deal with imprecise boundaries, i.e. when the start and end of the ncRNA is not known precisely. This issue is due to noise in the next generation sequencing mapping phase which is a common and essential step in the quantification of RNA expression. Imprecise boundaries are known to cause folding algorithm to yield significantly different structures (see Will *et al.*, 2012).

In this work, we propose a machine learning based approach to address all the aforementioned issues. Our method is efficient as it exploits recent advances in representation (Costa and De Grave, 2010; Da San Martino *et al.*, 2012b, 2016; Shervashidze *et al.*, 2011) and model induction (Bottou, 2010) to scale to large data volumes. The system is flexible as it is operates via a kernel on a graph representation that can encode arbitrarily complex information. We guarantee robustness using a windowed approach that jointly considers subsequences of different sizes and different starting positions. Finally the method is precise and it can reliably distinguish among sequences that belongs to known ncRNA families and sequences that do not belong to any family, a key property when annotating-omics datasets.

This article organized as follows: in Section 2 we review a variety of approaches for the *in silico* prediction of ncRNAs functions; in Section 3 we present our approach, explaining the various strategies to extract structural information and how to encode it in a graphical format suitable for processing by an efficient graph kernel; finally in Section 4 we empirically investigate the sensitivity/specificity trade off and compare the proposed approach to strong popular baselines.

2 Related works

Amongst the most prominent approaches to model functionally related ncRNA sequences we can distinguish those that need to perform comparative genome analysis (Parker *et al.*, 2011), and those that require in input only sequences of nucleotides. In this work we consider the latter case. Amongst these we can identify approaches that are based exclusively on sequence information, only on structural conformations or that can take both sources of information into account.

Sequence-based approaches have the advantage of being computationally efficient, but they are not suited to detect evolutionarily distant homologies, as they cannot relay on the more conserved structural information, and suffer therefore from elevated false negative error (Will *et al.*, 2007; Wilm *et al.*, 2006).

Purely structural approaches are based on the folding structure topology and ignore the nucleotide composition information. In Childs *et al.* (2009), the authors extract a number of graph properties, defined over the graph representation of the minimum free energy (MFE) conformation. These approached have the advantage of being applicable to highly dissimilar sequences, but, in addition to being computationally expensive, they incur in a high false positive error as sequence-specific clues are ignored (as an example just consider miRNA families which cannot be distinguished since they all form a single hairpin). For these reasons we do not discuss further these approaches.

Approaches that combine both structural and sequential information try to reach a better compromise between sensitivity and specificity. These methods range from more sophisticated graph kernels (Sakakibara *et al.*, 2007) to the state-of-the-art INFERNAL (Nawrocki *et al.*, 2009). Often these approaches have a strong modeling bias. INFERNAL for example does not cope well with variable sub-structure sizes (Mosig *et al.*, 2009) and suffers from severe scalability problems as it requires (i) the pre-computation of the alignment of all the input sequences and (ii) a computationally expensive calibration phase; graph-kernel-based methods commonly require the pairwise evaluation of a similarity notion between structures, yielding a computational complexity of $O(n^2)$ which prevents applications to large scale settings.

2.1 Sequence-based methods

The most popular sequence identification method is BLAST (Altschul *et al.*, 1990) that compares a sequence of interest q to a large database of sequences, to yield the closest matches. To do so, BLAST identifies compatible k-mers (subsequences of length k) and then expands the matching regions to find increasingly larger $k + i$-mers until maximal matching regions are found. BLAST can be used to predict family membership by using the similarity notion it defines in a k-nearest neighbors predictive system. With respect to the characteristics of efficiency, flexibility and robustness: BLAST is relatively efficient (linear complexity although with high constant factors due to the k-mer approximate matching); it is not flexible as it does not consider structural information; it is robust as it will try to find locally matching cores within the larger query sequence.

2.2 Sequence-structure-based methods

Predictive power is increased when considering structure information, but the computation of the *true* secondary structure of an RNA sequence is not an easy task. Even the most accurate predictors, based on experimentally tabulated energy models, when predicting the optimal MFE structure do not always obtain accurate results (Ding and Lawrence, 2003). Rather than considering a single answer

one could allow for multiple hypothesis to co-exist. The true structure is in fact likely to be among the set of the best sub- optimal structures. However, considering the whole ensemble of alternatives incurs in exponential costs (Hofacker and Schuster, 1999), which would negatively impact efficiency.

2.2.1 INFERNAL

Among the methods that consider the secondary structure information, the most popular one is INFERNAL (Nawrocki *et al.*, 2009). The tool is based on a variant of *profile stochastic context-free grammars* called covariance models (CMs). INFERNAL starts from an alignment and a pre-computed consensus structure, i.e. a secondary structure shared by sequences in the same family. CMs are closely related to profile Hidden Markov Models (HMM) (Yoon, 2009), as they both capture position-specific conservation information. However, in a profile HMM each position of the profile is treated independently, while in a CM base-paired positions are inter-dependent. Indeed, for many of these base-pairs, it is not the specific nucleotides that make up the pair that is conserved by evolution, but rather the fact that the pair maintains Watson-Crick base-pairing. With respect to the requirements of efficiency, flexibility and robustness, INFERNAL does not scale to large settings, both in training, since alignments of more than a few hundred sequences do not generally yield meaningful results, and in testing (see the Experimental sections); it is not flexible as it implements its own folding algorithm and hence cannot take experimental folding clues or multiple folding hypothesis into consideration; it is not robust since it relies on a global alignment procedure that usually breaks if sequence boundaries are misspecified.

2.2.2 Kernel methods

In the last decade, progress has been made in the Machine Learning and Data Mining community to extend the input data type from fixed size vectors to more flexible formats, ranging from sequences, to trees and finally to graphs. A successful paradigm has emerged in the field of supervised learning that makes use of linear models with good generalization properties (i.e. support vector machines [SVM]; Boser *et al.*, 1992) which can be easily extended both to structured input and to a non-linear setting using the so-called *kernel-trick* (Aizerman *et al.*, 1964), i.e. an implicit mapping into a very high-dimensional space (referred as *feature* space) expressed via a suitable dot product between two examples.

To deal with entities represented as graphs, a variety of graph kernels have been proposed in literature. Different notions of similarity are obtained choosing diverse types of substructures to consider, ranging from paths to small subgraphs.

Stem kernel. Stem kernel (Sakakibara *et al.*, 2007) is a natural extension of the *all-subsequences string kernel* (Shawe-Taylor and Cristianini, 2004) for RNA sequences. The feature space of the *all-subsequences string kernel* is defined as all the possible subsequences of the input string, both the contiguous and non-contiguous ones. For example, two RNA sequences CUG and CAU have four common subsequences: ϵ (the empty string), C, U and C-U, where - represents the bond between two nucleotides. Note that the characters in a subsequence do not need to be contiguous. The all-subsequences kernel calculates the inner product of the feature vectors by counting all common subsequences (considering gaps). The Stem kernel is a simple kernel for RNA secondary structures that maps them in a feature space representing all the possible base pairs. The kernel calculates the inner product in the feature space implicitly, starting from RNA sequences, thus no additional information about the secondary structure is needed. The computational complexity of calculating the Stem kernel between two RNA sequences of length n is $O(n^4)$.

Marginalized kernel on RNA sequences. Karklin,Y. *et al.* (2005) proposed the application of marginalized kernel to RNA sequences represented as a *labeled dual graph*. In this representation, every node represents helical regions (sequences of paired nucleotides) and the edges represent the loops (sequences of non-paired nucleotides). This representation uses only the information about the pairing of the nucleotides in the sequence, discarding the information about the type of structure (stems, hairpins, bulges and internal loops) and resort to an implicit way to encode it in final graph. A marginalized kernel (a kernel that counts the common random walks between two graphs) is then applied on these graphs. The computational complexity on RNA sequences of length n is $O(n^3)$. With respect to the requirements of efficiency, flexibility and robustness, marginalized graph kernels do not scale, both because of the high per-sequence cost and because of the quadratic cost of a pairwise similarity evaluation; they are not very flexible as they commit to a specific way to compute all structures and cannot easily take into account experimental evidence to bias structural hypothesis; finally, they are not robust since the structure computation yields significantly different structures when boundaries are misspecified.

3 Materials and methods

In this section, we detail how to derive a sparse vector representation for ncRNAs using an explicit graph kernel (Costa and De Grave, 2010; Da San Martino *et al.*, 2016) that can then be used directly for classification tasks e.g. by efficient Stochastic Gradient Descent SVMs.

3.1 Graph kernel

We adopt the recently introduced (Costa and De Grave, 2010) fast kernel called Neighborhood Subgraph Pairwise Distance Kernel (NSPDK), since this kernel is suitable for large datasets of sparse graphs with discrete vertex and edge labels. Here, to increase efficiency, we choose an explicit version that materializes all the features in a sparse vector representation. The NSPDK kernel considers as features, all pairs of small subgraphs (neighborhood subgraphs up to radius r^*) that can be connected by a shortest-path of length at most d^*. The hyper parameters r^* and d^* are user-defined and in practice are small integers (<10). The type of features that the NSPDK is considering, when considering graphs encoding RNA structures, is depicted in Figure 2. The NSPDK kernel between two graphs is defined as the sum of the products between the counts of matching features (for all radii and distances). Note that this computation requires to solve the (rooted) graph isomorphism problem. Since running an exact isomorphism test is computationally expensive, the authors propose to substitute the test with a more efficient graph invariant computation (see Costa and De Grave, 2010 for further details).

More formally, let $G = (V, E)$ be a graph, with V_G being the set of vertices and $E = \{(u, v) | u, v \in V\}$ the set of edges. Two vertices are adjacent if there is an edge connecting them. A path is an alternating sequence of vertices and edges, starting and ending at a vertex, in which each edge is adjacent in the sequence to its two endpoints. The shortest-path distance between two vertices u and v is the number of edges in the shortest path connecting them. The *neighborhood* of radius r of a vertex $v \in V(G)$ is the set of vertices at a distance less than or equal to r from v. In a graph, an induced

subgraph of a set of vertices W is the graph that have W as vertices, and contains every edge of the original graph whose endpoints are in W. The *neighborhood subgraph* of radius r of a vertex v is the subgraph induced by the neighborhood of radius r of v. It is denoted by $N_r^v(X)$. Two graphs G and G' are said to be isomorphic if it exists a bijection $f : V(G) \to V(G')$ such that $(u, v) \in E_G$ if and only if $(f(u), f(v)) \in V_{G'}$.

We define a relation $R_{r,d}(A^u, B^v, G)$ between two rooted graphs A^u, B^v and a graph G to be true if both A^u and B^u are in $\{N_r^v | v \in V(G)\}$ (where the set inclusion is up to isomorphism) and the shortest-path distance between u and v in G is exactly d. In other words, the relation is true for all pairs of neighborhood graphs of radius r whose roots are at distance d in a given graph G. We define the *NSPDK* kernel as:

$$K_{r*,d*}(G, G') = \sum_{r=0}^{r*} \sum_{d=0}^{d*} \sum_{\substack{A^v, B^u \in R_{r,d}^{-1}(G) \\ A^{v'}, B^{u'} \in R_{r,d}^{-1}(G')}} \delta(A^v, A^{v'})\, \delta(B^u, B^{u'})$$

where δ is the Kronecker delta function. The computational complexity of the kernel is $O(|V||V_{r^*}||E_{r^*}| \log |E_{r^*}|)$, where $|V_{r^*}|$ and $|E_{r^*}|$ are the maximum number of nodes and edges, respectively, among the subgraphs of radius at most r^*.

3.2 Explicit feature representation

The idea here is to materialize the implicit feature encoding which is key to obtain linear efficiency in the classification phase. Differently from (Costa and De Grave, 2010), we here make use of the integer code for the invariant graph encoding as a feature indicator (Da San Martino *et al.*, 2012a; Frasconi *et al.*, 2014). In this way we can interpret the integer associated to each feature (i.e. each pair or neighborhood subgraphs of radius r whose roots are at distance d) as the feature key and the (normalized) count of occurrences as its value. This allows us to obtain an explicit feature encoding for a given graph G as a sparse vector in R^m (with a very high dimensionality m). The feasibility of the approach lies in the fact that the encoding does not produce an exponential number of features, as it would happen with most graph kernels that enumerate all possible general subgraphs. Instead NSPDK limits the number of non-zero features to $O(r^* d^* |V(G)|^2)$, i.e. one feature for each pair of vertices times each possible combination of values for the radius and the distance. Note that typically $r^* \in [0, 5]$ and $d^* \in [0, 10]$ and hence the multiplicative factor is $\approx 5 - 50$. Moreover for sparse graphs the number of vertices that are reachable within fixed small distance is typically small (depending on the average degree) so that the dependency on the vertex set size can be more tightly approximated by $O(|V(G)|)$. As a result each graph is mapped into a sparse vector that lives in a very high-dimensional feature space but that has a number of non-zero features which is a small multiple of the number of the graph's vertices.

3.3 Multiple structures representation

There are several ways to represent RNA secondary structures, including the bracketed representations (where nucleotides are converted to nodes and bonds to edges), and tree representations (where base pairs are converted to 'stem' nodes and loop nucleotides are converted to 'loop' nodes). Each representation has different advantages and disadvantages including information loss and complexity of calculation (Fera *et al.*, 2004). In this article, we chose to adopt a loss-less representation where nodes represent nucleotides and edges are the bonds between them, either of the backbone type or of the binding type.

It is known that the predicted MFE structure is often not the one that is truly active in the cell (Ding and Lawrence, 2003). To address this issue we allow multiple suboptimal solutions. Given that the number of suboptimal solutions grows at an exponential rate w.r.t. the free energy threshold, one needs to resort to some strategy to select a small subset representative of the overall structure landscape. To this end we follow *RNAshapes* (Giegerich *et al.*, 2004). The idea is to select structures that have *fundamental differences*. RNAshapes formalizes the notion of abstract shape and allows the efficient computation of the representative structure of minimal free energy within each abstract shape class, which is advantageous since the number of different shapes is considerably smaller than the number of different structures. The abstraction procedure preserves hairpins and multi-loops, but abstracts notions such as the primary sequence, stack lengths, bulges, internal loops and single-stranded regions. RNAshapes supports five different abstraction levels, allowing the preservation of more details up to all loops and unpaired regions.

As an example, we report the different shape abstractions for the same sequence used in Steffen *et al.* (2006), where the underscore symbol '_' stands for unpaired region and the brackets '[]' stand for a stem region:

```
AUCGGCGCACAGGACAUCCUAGGUACAAGGCCGCCCGUU
.((((.((.(((...))).(((....)))))))).
```

Shape_Type	Result
1	_[_[_[]_[]]]_
2	[_[[][]]]
3	[[[][]]]
4	[[][]]
5	[[][]]

Given a fixed shape abstraction level we follow (Giegerich *et al.*, 2004) and consider all possible shapes in which a sequence can fold. Since many structures can have the same shape, we select as the unique representative of the shape the MFE configuration, called the *shrep* (for shape representative). We rank all the shreps and consider only the k most energetically favorable ones. Both the abstraction level and the value for k are considered as hyper parameters to be optimized during the model selection phase.

3.4 Misspecified sequence boundaries

Although the abstract shape approach protects against committing to a single wrong structural hypothesis, we still have to address the problem of uncertainty on the exact sequence boundaries. This problem does not only arise in the case of genome annotation but can also manifest itself in the case of transcriptomic data do to noise in the reads decoding and in the mapping process. Uncertainty in the sequence boundaries has important consequences since folding algorithms are not robust to this type of noise and can yield drastically different structures when a few nucleotides are added or removed from the beginning or end of the sequence. To achieve robustness we adopt a multi-windowed approach: instead of considering only the full sequence we consider the set of all subsequences obtained from a sliding window approach. Instead of committing to a single window and shift size, we iterate the sliding window approach for multiple values of the window (a fix the shift to a fraction of the size). Each resulting subsequence is then folded using the RNAshapes approach detailed in the previous section, yielding k suboptimal graphs per

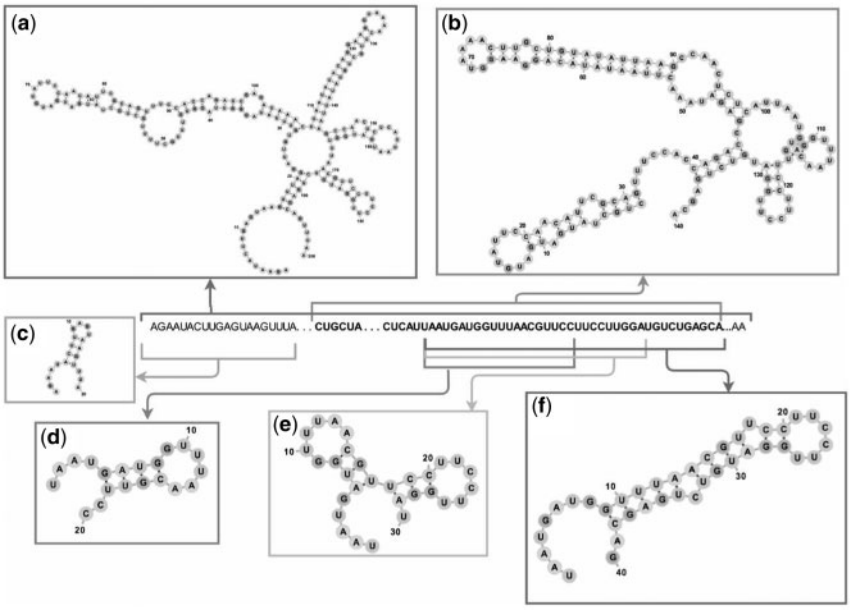

Fig. 1. Summary of the proposed approach. **(a)** Folding obtained for the complete (signal + padding) sequence; **(b)** folding for the signal sequence alone. The proposed approach splits each sequence in overlapping windows. The procedure is iterated for different window sizes. **(c)** Folding for the first 20 nt of the sequence. Note that in this window contains only random context. **(d–f)** Foldings for window size 20, 30 and 40, respectively. The resulting graph is the disjoint union of all the graphs produced by the windowing approach combined with the RNAshapes multiple folding strategies

subsequence. The disjoint union of all the graphs constitutes the final graph encoding for the original sequence. Figure 1 depicts the windowing approach, where in bold we represent the original sequence retrieved from Rfam.

Note that the window size (ranging from 25 to 100 nucleotides in our experiments) influences the *locality* of the structural features that the method is aware of: larger window sizes allow to capture multi-loop structures, while smaller ones can only capture single hairpin loops. In Figure 1 we give an example of how the proposed approach can correctly identify some of the true hairpins that are not detected when folding the entire sequence.

3.5 Stacking base pairs

It is known that quadruplets formed by two consecutive stacking base pairs carry considerable information about the stability and hence likelihood of the corresponding stem structure. To better encode this type of information we introduce additional vertices with a non-informative label and link them to each of the four nucleotides of the stack; in this way the neighborhood subgraphs features correspond exactly to individual stacking base pairs (see Fig. 2).

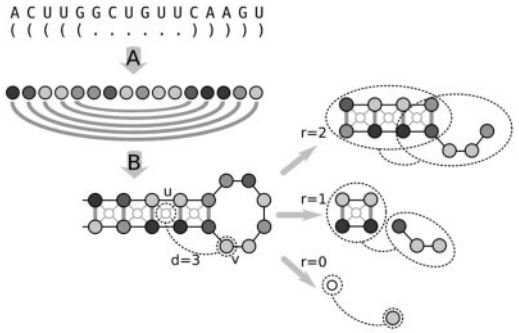

Fig. 2. RNA secondary structure encoding and graph kernel features. **(A)** The graph encoding includes nucleotide information (vertex labels) and binding information (edge labels), here depicted with different shades of gray to distinguish backbone links from base pairing. **(B)** Additional vertices are inserted in order to induce features related to stacking base-pairs quadruplets (thin light gray vertices at the center of each stacking pair). Right: example of features induced by the graph kernel NSPDK for a pair of vertices *u, v* at distance 3 with radius 0,1,2. Neighborhood graphs are enclosed in dashed ovals

4 Experimental analysis

We cast the ncRNAs annotation task as a multiclass problem where each class is a functionally distinct set of ncRNAs. We adopt the *one-versus-all* multiclass formulation and evaluate the predictive performance using the area under the precision/recall curve (APR) which is more informative in highly unbalanced tasks than area under the curve for the receiver operating characteristic (AUC ROC).

4.1 Robustness analysis

4.1.1 Dataset construction

We extract data from Rfam (Gardner *et al.*, 2011), a database that catalogs ncRNAs using curated sequence alignments and CMs. ncRNA sequences are grouped in 'families' if they share the same function and have a clear common ancestor. Out of the total set of 2588 Rfam families we selected a subset of families that could satisfy the following requirements. (i) A family should have a number of members that allows statistical learning, we therefore selected the families with at least 100 sequences. ncRNAs typically have a length in the range of 40–400 nt. As current structural prediction tools becomes unreliable for sequences that exhibit interactions spanning more than 150 nt (Lange *et al.*, 2012), we selected families with an average sequence length <150 nt. (ii) To study the robustness of various approaches we added noise to each sequence boundary by adding a random number of nucleotides both on the left and on the right of the sequence. The *padding* is constructed so as to respect the nucleotide frequency of each specific sequence, while the length of the added noisy context varies from 0 to the necessary number of nucleotides so as to obtain sequences of the same length of 250,

Table 1. Predictive performance estimate (APR) for the baseline methods BLAST and INFERNAL and our approach on the dataset of sequences of original length and fixed (padded) length of 200 nt

Class	Rfam	No.	Average	Original len.			Padding to 200 nt		
ID	class	seq.	len.	BL	IN	Our	BL	IN	Our
1	RF00001	1180	104 ± 27	0.59	**0.85**	0.81	0.61	**0.92**	0.78
2	RF00005	703	114 ± 26	0.52	0.02	**0.57**	0.49	0.41	**0.50**
3	RF00015	1056	121 ± 27	0.88	0.57	**0.99**	0.89	0.98	**0.99**
4	RF00016	222	105 ± 27	0.5	0	**0.59**	0.5	0	**0.78**
5	RF00019	1225	110 ± 6	0.6	**0.96**	0.81	0.61	**0.98**	0.72
6	RF00020	264	117 ± 5	0.54	**0.95**	0.8	0.54	**0.85**	0.74
7	RF00026	318	105 ± 4	0.55	**0.98**	0.91	0.49	**0.96**	0.85
8	RF00029	572	92 ± 24	0.55	0.89	**0.90**	0.57	0.73	**0.85**
9	RF00031	233	65 ± 4	0.51	**0.62**	0.15	**0.51**	0.07	0.11
10	RF00050	146	132 ± 20	0.56	0.78	**0.90**	0.56	0.83	**0.98**
11	RF00059	594	106 ± 19	0.6	**0.99**	0.89	0.6	0.88	**0.93**
12	RF00066	238	60 ± 7	0.54	0.88	**0.97**	0.47	**0.90**	0.81
13	RF00097	340	102 ± 13	0.51	**0.91**	0.22	0.51	**0.91**	0.2
14	RF00140	124	99 ± 14	0.57	**0.87**	0.85	0.61	0.81	**0.83**
15	RF00156	290	120 ± 19	0.52	0.86	**0.91**	0.52	**0.95**	0.91
16	RF00162	111	143 ± 15	0.5	**0.81**	0.65	0.5	**0.73**	0.6
17	RF00163	287	45 ± 2	0.51	0.78	**0.91**	0.54	**0.66**	0.43
18	RF00169	195	99 ± 2	0.52	**0.95**	0.84	0.52	**0.87**	0.73
19	RF00263	109	115 ± 20	0.5	**0.88**	0.14	0.5	**0.77**	0.56
20	RF00322	195	107 ± 21	0.54	**0.91**	0.81	0.55	**0.96**	0.9
21	RF00406	165	132 ± 4	0.52	**0.88**	0.75	0.27	0.69	**0.81**
22	RF00409	1168	138 ± 3	0.88	**0.98**	0.97	0.91	0.92	**0.98**
23	RF00420	428	121 ± 4	0.55	**0.97**	0.9	0.55	**0.92**	0.89
24	RF00504	583	99 ± 22	0.53	0.95	**0.97**	0.85	0.85	**0.98**
25	RF00557	142	142 ± 19	0.53	**0.88**	0.8	0.53	0.87	**0.89**
26	RF00560	292	129 ± 6	0.65	**0.93**	0.91	0.65	0.88	**0.95**
27	RF00619	185	117 ± 15	0.53	**0.94**	0.67	0.36	**0.92**	0.49
28	RF00645	110	123 ± 20	0.61	**0.85**	0.83	0.61	0.73	**0.89**
29	RF00655	234	105 ± 9	0.58	0.89	**1.00**	0.53	0.71	**1.00**
30	RF00779	248	82 ± 20	0.52	0.88	**0.94**	0.57	0.9	**0.99**
31	RF00875	233	82 ± 3	0.55	0.91	**0.94**	0.64	0.85	**0.95**
32	RF00876	124	86 ± 4	0.52	0.89	**0.90**	0.44	**0.94**	0.73
33	RF00906	1270	137 ± 20	0.54	**0.99**	0.97	0.54	**0.99**	0.96
34	RF00989	283	114 ± 8	0.55	**0.89**	**0.89**	0.58	0.8	**0.96**
35	RF01016	524	119 ± 5	0.65	0.96	**1.00**	0.64	0.89	**1.00**
36	RF01028	140	70 ± 17	0.53	0.89	**0.99**	0.56	**0.99**	**0.99**
37	RF01055	105	148 ± 16	0.5	**0.77**	0.63	0.5	**0.82**	0.72
38	RF01059	952	102 ± 13	0.54	0.98	**0.99**	0.53	0.97	**1.00**
39	RF01063	198	88 ± 15	0.51	0.88	**0.98**	0.53	0.7	**0.87**
40	RF01699	223	111 ± 33	0.54	**0.87**	0.84	0.54	0.67	**0.93**
41	RF01705	152	99 ± 28	0.5	0.73	**0.87**	0.51	0.52	**0.85**
42	RF01725	148	107 ± 8	0.5	**0.69**	0.56	0.51	0.35	**0.56**
43	RF01731	119	140 ± 38	0.52	**0.89**	0.66	0.27	0.5	**0.76**
44	RF01734	110	73 ± 12	0.5	**0.80**	0.56	**0.50**	**0.50**	0.2
45	RF01739	125	94 ± 30	0.5	0.82	**0.84**	0.5	0.5	**0.76**
46	RF01942	491	103 ± 14	0.58	**0.97**	0.87	0.63	0.52	**0.93**
47	RF02012	116	135 ± 12	0.5	**0.84**	0.69	0.5	0.5	**0.85**
AVG		367	112 ± 23	0.55	∗0.84	0.80	0.55	0.76	∗0.79

Note: BL, blast; IN, infernal; Our, our proposed approach. The methods with * perform significantly better than the other ones.

300, 350 and 400 nt. In this way we can control for the confounding effect of the sequence length which could otherwise be used by a learning algorithm as a discriminative feature to identify specific families. (iii) To control for near rote learning and test for the generalization capacity we split each family in three subsets: one for training, one for validation and one for testing, ensuring that no sequence in the validation and test splits had sequence similarity >50% with any other sequence in the training split. The families that yielded an empty validation or test set under these requirements were discarded. Note that the validation set was used only to tune the hyper-parameters of the learning algorithm, while the test set was used to derive an unbiased estimate of the predictive performance. In Table 1 we report the 47 Rfam families that satisfied all the requirements. The average number of examples per class (including training, validation and test sets) is 367, with a maximum of 1270 examples and a minimum of 105, for a total of 17 270 sequences.

4.1.2 Experimental results and discussion

We evaluated the predictive performance of our method in comparison to the two baselines presented in Section 2.1. Other methods presented in Section 2.2.2 have not been evaluated since their computational complexity is too high for the large scale setting we are interested in.

We first tested if the sequence length is alone a significant predictor for family membership. Using a *k-nearest neighbor* classifier exclusively on the sequence length yielded an average APR of only 0.09.

We then considered as a baseline BLAST. To classify a sequence s, we query the training database to obtain the k highest matches (where k is an hyper-parameter of the method). Each retrieved sequence counts as a vote for the class it belongs to and the prediction is then obtained via the majority vote. The second baseline is based on an INFERNAL model (see Section 2.2.1) for each family. The sequences in each family are aligned with the sequence alignment program MUSCLE (Edgar, 2004) and a consensus structure is identified with RNAalifold (Hofacker *et al.*, 1989), finally we built a calibrated CM with INFERNAL. In this way, given some representative sequences in the training set, we generate a model for each family. INFERNAL can then estimate the probability of a query sequence w.r.t. each model. As a prediction we consider the model with the highest confidence. Finally, for our proposed method, described in Section 3, we trained a multi-class SVM using the one-versus-all approach. As learning procedure, we adopted the Stochastic Gradient Descent because it is very fast compared with more classic algorithms (e.g. SMO) while achieving very similar predictive performances (Zhang, 2004). For the assessment of the classification performance, we considered the area under the Precision/Recall curve (APR) measure. After preliminary experiments, we fixed the shape type parameter t of RNAshapes to $t = 4$. We optimized the other hyper parameters of RNAshapes using a grid-search approach. We validated the parameter w controlling the window size in the set {75, 100}. The parameter M controlling the number of foldings generated from each fragment has been validated in the set {2, 3}. Note that the performance of the kernel methods can be further improved considering, for example, multiple window sizes at the same time. However, this approach would increase the parameter space of the method, and thus we decided to leave this approach as a future work. Finally, for each dataset the kernel parameters have been optimized using a grid-search approach on the following sets: $r = \{1, 2, 3\}$, $d = \{3, 6\}$. Thus, a total of 6 kernel parameter configurations have been tested, that combined with the RNAshapes parameters gives a total of 24 possible parameter configurations for the proposed method. Finally, the α parameter of the SGD has been validated in the set $\{10^{-1}, .., 10^{-7}\}$.

In Table 1 we report the experimental results on the test set for the original dataset, and for the dataset where padding has been added to each sequence (up to 200 nt). We start our discussion considering the dataset where no padding has been added. In this case, INFERNAL is the best performing method. This is not a surprise since the dataset we consider comes from the Rfam database, where the INFERNAL tool is used to discover the sequences belonging to a family, starting from a small set of manually aligned seeds (see Section 4.1.1). However, in this setting our proposed method shows predictive performances that are close to the INFERNAL ones. Moreover, the computational time required from our proposed method is considerably lower that the one of INFERNAL, as will be detailed in Section 4.3. Let us consider the dataset with padding to 200 nucleotides. Surprisingly BLAST performs well for a few RNA families. Indeed, for some ncRNA families the sequence carries enough information about the function. In these cases, it's useless to apply techniques that consider the secondary structure because of the additional computational complexity. Moreover in these cases, considering the structure may also introduce additional noise to the data, resulting in a decay in classification performances. This happens in the classes 9 and 44 in our dataset. However, for the majority of the classes, the approaches that considers secondary structure information (INFERNAL and NSPDK) achieve significantly higher classification performance w.r.t. BLAST. The comparison between INFERNAL and the NSPDK is tighter: INFERNAL wins in 18 cases, while NSPDK wins in 27 (on class 36 the performances for the two methods are the same). On average, the APR value on the test set for BLAST is 0.55, for INFERNAL it is 0.76 while for the NSPDK kernel is 0.79. In order to assess if the difference in performance among the proposed method and the considered baselines is statistically significant, we performed a Wilcoxon signed-rank test (Wilcoxon, 1945). We consider the performance difference significant if the one-tail test results in a P-value <0.05. Both INFERNAL and the proposed method perform significantly better than BLAST in all the considered settings. INFERNAL performs significantly better than our proposed method when no padding is present (as expected: we recall that the dataset is biased in favour of INFERNAl), while our proposed method performs significantly better than infernal when considering padding to 200 nucleotides. We than analyzed what happens when we increase the size of the padding. Table 2 summarizes this set of experiments. We can see that, adding more and more padding, the predictive performances of the two methods (INFERNAL and our proposed method) decrease. However, our proposed method shows constantly higher AUC with respect to INFERNAL, showing that it is more robust to the noise in the boundaries of the sequences. Also for these experiments, we computed the same statistical test. The proposed method performs significantly better than infernal with all the considering padding lengths (with the only exception of length 300 nt, where the proposed method performs better but the difference is not statistically significant).

4.2 Genome simulation analysis

In this section, we present our second set of experiments, aimed at assessing the *robustness* of the proposed method when in the dataset there are non-functional RNA sequences, i.e. sequences that do not belong to any ncRNA family.

4.2.1 Dataset construction

We started from the same dataset as in the previous section, fixing the length of the sequences to 200 nucleotides. We want to generate sequences that do not belong to any class, in order to mimick the fact that in whole-genome analysis, the majority of the RNA is non-functional or it does not belong to any known ncRNA family. To do so, for each sequence in the dataset we generated a number t of shufflings (possibly preserving the di-nucleotides, i.e. shuffling two nucleotides at a time). We tested different values of t, i.e. 1,2,6,10, obtaining increasingly difficult classification problems. Note that, with $t = 1$, we generated one shuffling for each sequence in the original dataset, and we considered the standard shuffling (i.e. we shuffled the single nucleotides). For the other values of t, we generated the same number of single and di-nucleotide shufflings. Thus, the number of samples in each version of the dataset is increasing.

Table 2. APR results of INFERNAL and our proposed method, at the increasing of noise around the real sequences

Sequences length	APR on test set		Test times	
	IN	Our	IN	Our
original	*0.84 ± 0.10	0.80 ± 0.10	7 h 30 min	15 min
200	0.76 ± 0.11	*0.79 ± 0.11	30 h	37 min
250	0.73 ± 0.12	*0.78 ± 0.11	48 h	43 min
300	0.73 ± 0.13	**0.75** ± 0.11	75 h	46 min
350	0.69 ± 0.11	*0.72 ± 0.12	100 h	1 h 12 min
400	0.56 ± 0.12	*0.68 ± 0.13	127 h	1 h 30 min

Note: The method with * performs significantly better than the other.

Table 3. APR results of INFERNAL and our proposed method, with the presence of a 'zero' class containing random sequences

Sequences length	APR test set		Test times	
	IN	Our	IN	Our
1× negatives	0.08 ± 0.10	*0.76 ± 0.11	102 h	2 h 32 min
2× negatives	—	0.72 ± 0.11	—	3 h 30 min
6× negatives	—	0.70 ± 0.12	—	3 h 40 min
10× negatives	—	0.62 ± 0.13	—	5 h

Note: The method with * performs significantly better than the other.

With $t = 1$, the new dataset have the double the sequences as in the original dataset, with $t = 2$ three times and so on.

4.2.2 Experimental results and discussion

Table 3 reports the APR of the proposed method, together with the INFERNAL baseline, on this set of experiments. In this setting, the *Infernal* baseline performs poorly already with $t = 1$. On the contrary, our proposed method is able to deal with such scenario. As expected, the predictive performances decrease increasing t. Overall, this experiment shows that our proposed approach is robust to this kind of noise, i.e. to the presence of sequences that do not belong to any ncRNA family.

4.3 Computational runtime analysis

In this section, we discuss the computational requirements of the considered methods. We divide the computational times in the *training* phase and the *test* phase. The *training* phase consists in all the operations that have to be performed in order to build the predictive model, and that are executed only once. The *test* phase, on the contrary, comprises the step necessary to classify the test instances. For the BLAST baseline the training phase consists in the computation of some indices on the training sequences. The test phase consists in the scanning of the target sequences against the training ones. This baseline is very fast, and requires only few seconds to be computed (both training and test phases) in all our experimental settings. INFERNAL training phase requires to build a CM for each class, and to *calibrate* it. Its test phase consists in the scanning (with the tool *cmscan*) of the target sequences against the generated CMs. As for the proposed method, its training phase consists in the feature generation for the training and validation instances, the training of the classifier (one for each parameter combination), and the parameter selection (via classification of validation instances). The test phase consists in the feature generation for the test instances, and their classification (with SVM).

Let us start our discussion with the first experimental setting, presented in Section 4.1. The training step for the INFERNAL baseline on the dataset with no padding required ∼10 h; with a fixed length of 200 nt, it required ∼20 h; with sequences of 350 nt, the calibration phase required ∼90 h. The computational times required for the test phase, for all the considered sequence lengths, are reported in Table 2, fourth column. We can see that they are in the order of several hours. Indeed, this is the slowest method we considered. As for the proposed method, with no padding the training phase require at most 3 h for each parameter configuration. With sequences of 200 nt, the training phase required at most 4.5 h; increasing the length to 350 nt, it required at most 6 h. The computational times required by our proposed method for the test phase, are reported in Table 2, fifth column. Our proposed method is very fast on the test phase. Note that there are several techniques for speeding up the training phase of kernel methods. For example, parameter estimation can be performed on just a subset of the training data. Moreover, instead of a grid-search, a local search on the parameters grid can be performed. Moreover, a single training procedure can be performed considering all the parameter configurations at once (Massimo *et al.*, 2016). The exploration of these approaches is however out of the scope of this article, where we put emphasis on the test times, and we leave the analysis of these techniques for a future work. Concluding, the INFERNAL baseline and the proposed method have comparable training times (in orders of magnitude), due to the fact that many parameter configurations have to be validated for the proposed method. However, in the test phase, our proposed method is thousands of times faster than INFERNAL.

We now discuss the second experimental setting, presented in Section 4.2. For the case with $t = 1$, Infernal training phase required 113 h. The scanning phase, as reported in Table 3, fourth column, required 102 h. The performance of the method is poor, and it is computationally very demanding, so we decided not to go further with higher values of t. As for the proposed method, with $t = 1$ the training phase required 5.5 h; with $t = 2$ the training phase required overall at most 6 h. With $t = 6$ it required 12 h, while with $t = 10$ the computational time increases to 15 h. The computational times required by the proposed method for the test phase are reported in Table 3, fifth column. Also in this scenario, our proposed method is very fast compared with INFERNAL.

5 Conclusions and future work

In this article we proposed a novel approach for non-coding RNA functional annotation based on graph kernel techniques capable of addressing the key requirements of (i) efficiency, (ii) flexibility and (iii) robustness. An extensive empirical evaluation shows that (i) it is possible to learn from tens of thousands of examples and test in linear time paving the way to efficient genome scale annotations; (ii) it is possible to easily encode multiple windows and suboptimal folding structures to improve predictive accuracy; and (iii) it is possible to be resilient to sequence boundary misspecification and obtain a prediction accuracy that degrades gracefully with the increase in noise level. Given the efficiency of the proposed approach we plan to develop an efficient and easy to use web server to perform ncRNA annotations for novel genomes. In the future we will investigate the encoding of more complex structural information such as the presence of pseudo-knots and how to include structural information derived from experimental protocols such as SHAPE and hiCLIP.

Funding

This work was supported by the Federal Ministry of Education and Research [BMBF grant 031 6165A e:Bio RNAsys]; the German Research Foundation [DFG grant BA 2168/3-3]; and the University of Padova under the strategic project BIOINFOGEN.

Conflict of Interest: none declared.

References

Aizerman,M.A. *et al*. (1964) Theoretical foundations of the potential function method in pattern recognition learning. *Automat. Remote Contr*, **25**, 917– 936.

Altschul,S. *et al*. (1990) Basic local alignment search tool. *J. Mol. Biol*., **215**, 403–410.

Boser,B.E. *et al*. (1992) A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pp. 144–152. ACM Press, Pittsburgh, PA.

Bottou,L. (2010) Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, Paris, France.

Burge,S.W. *et al*. (2013) Rfam 11.0: 10 years of RNA families. *Nucleic Acids Res*., **41**, D226–D232.

Childs,L. *et al*. (2009) Identification and classification of ncRNA molecules using graph properties. *Nucleic Acids Res*., **37**, e66.

Costa,F. and De Grave,K. (2010). Fast neighborhood subgraph pairwise distance kernel. In: Joachims, J. F. and Thorsten (eds.), *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 255–262. Omnipress, Haifa, Israel.

Da San Martino,G. *et al*. (2012a). A memory efficient graph kernel. In: *The 2012 International Joint Conference on Neural Networks (IJCNN)*. IEEE, Brisbane, QLD, Australia.

Da San Martino,G. *et al*. (2012b). A Tree-Based Kernel for Graphs. In: *Proceedings of the Twelfth SIAM International Conference on Data Mining*, pp. 975–986.

Da San Martino,G. *et al*. (2016) Ordered decompositional DAG kernels enhancements. *Neurocomputing*, **192**, 92–103.

Deigan,K.E. *et al*. (2009) Accurate shape-directed rna structure determination. *Proc. Natl. Acad. Sci. USA*, **106**, 97–102.

Ding,Y. and Lawrence,C.E. (2003) A statistical sampling algorithm for RNA secondary structure prediction. *Nucleic Acids Res*., **31**, 7280–7301.

Edgar,R.C. (2004) MUSCLE: Multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res*., **32**, 1792–1797.

Fera,D. *et al*. (2004) RAG: RNA-As-Graphs web resource. *BMC Bioinformatics*, **5**, 88.

Frasconi,P. *et al*. (2014) klog: A language for logical and relational learning with kernels. *Artif. Intell*., **217**, 117–143.

Gardner,P.P. *et al*. (2011) Rfam: Wikipedia, clans and the "decimal" release. *Nucleic Acids Res*., **39**, D141–D145.

Giegerich,R. *et al*. (2004) Abstract shapes of RNA. *Nucleic Acids Res*., **32**, 4843–4851.

Hofacker,I.L., and Schuster,P. (1999) Complete suboptimal folding. *Biopolymers*, **49**, 145–165.

Hofacker,I.L. *et al*. (1989) Fast folding and comparison of RNA secondary structures. *Monatshefte fuer Chemie Chemical Monthly*, **125**, 167–188.

Lange,S.J. *et al*. (2012) Global or local? predicting secondary structure and accessibility in mrnas. *Nucleic Acids Res*., **40**, 5215–5226.

Massimo,C.M. *et al*. (2016) Hyper-parameter tuning for graph kernels via multiple kernel learning. In: *Neural Information Processing of ICONIP, Kyoto, Japan, October 16–21, 2016, Part I*, pp. 214–223. Springer.

Karklin,Y. *et al*, (2005) Classification of non-coding RNA using graph representations of secondary structure. In: *Pac Symp Biocomput*., 2005:4-15. January 4-8, Fairmont Orchid, Big Island of Hawaii, USA.

Mosig,A. *et al*. (2009) Customized strategies for discovering distant ncRNA homologs. *Brief. Funct. Genomic. Proteomic*, **8**, 451–460.

Nawrocki,E.P. *et al*. (2009) Infernal 1.0: inference of RNA alignments. *Bioinformatics (Oxford, England)*, **25**, 1335–1337.

Parker,B.J. *et al*. (2011) New families of human regulatory RNA structures identified by comparative analysis of vertebrate genomes. *Genome Res*., **21**, 1929–1943.

Sakakibara,Y. *et al*. (2007) Stem kernels for RNA sequence analyses. *J. Bioinformatics Comput. Biol*., **5**, 1103–1122.

Shawe-Taylor,J., and Cristianini,N. (2004) *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA.

Shervashidze,N. *et al*. (2011) Weisfeiler-Lehman graph kernels. *J. Mach. Learn. Res*., **12**, 2539–2561.

Steffen,P. *et al*. (2006) RNAshapes: an integrated RNA analysis package based on abstract shapes. *Bioinformatics (Oxford, England)*, **22**, 500–503.

Sugimoto,Y. *et al*. (2015) hiCLIP reveals the in vivo atlas of mRNA secondary structures recognized by Staufen 1. *Nature*, **519**, 491–494.

Tinoco,I. and Bustamante,C. (1999) How RNA folds. *J. Mol. Biol*., **293**, 271–281.

Wilcoxon,F. (1945) Individual comparisons by ranking methods. *Biometrics*, **1**, 80–83.

Wilkinson,K.A. *et al*. (2006) Selective 2′-hydroxyl acylation analyzed by primer extension (SHAPE): quantitative RNA structure analysis at single nucleotide resolution. *Nat. Protoc*, **1**, 1610–1616.

Will,S. *et al*. (2007) Inferring noncoding RNA families and classes by means of Genome-Scale Structure-Based clustering. *PLoS Comput. Biol*., **3**, e65.

Will,S. *et al*. (2012) LocARNA-P: Accurate boundary prediction and improved detection of structural RNAs. *RNA*, **18**, 900–914.

Willingham,A.T. and Gingeras,T.R. (2006) {TUF} love for junk {DNA}. *Cell*, **125**, 1215– 1220.

Wilm,A. *et al*. (2006) An enhanced RNA alignment benchmark for sequence alignment programs. *Algorithms Mol. Biol*., **1**, 1–11.

Yoon,B.-J. (2009) Hidden Markov models and their applications in biological sequence analysis. *Curr. Genomics*, **10**, 402–415.

Zhang,T. (2004) Solving large scale linear prediction problems using stochastic gradient descent algorithms, ICML 2004, Banff, Alberta, Canada.