ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Weighted Tree Kernels for Sequence Analysis

Christopher J. Bowles and James M. Hogan

School of Electrical Engineering and Computer Science,
Queensland University of Technology,
2 George St, Brisbane, QLD, 4000. AUSTRALIA.
{j.hogan@qut.edu.au}

Abstract. Genomic sequences are fundamentally text documents, admitting various representations according to need and tokenization. Gene expression depends crucially on binding of enzymes to the DNA sequence at small, poorly conserved binding sites, limiting the utility of standard pattern search. However, one may exploit the regular syntactic structure of the enzyme's component proteins and the corresponding binding sites, framing the problem as one of detecting grammatically correct genomic phrases. In this paper we propose new kernels based on weighted tree structures, traversing the paths within them to capture the features which underpin the task. Experimentally, we find that these kernels provide performance comparable with state of the art approaches for this problem, while offering significant computational advantages over earlier methods. The methods proposed may be applied to a broad range of sequence or tree-structured data in molecular biology and other domains.

1 Introduction

Trees are a fundamental abstraction for the representation of structured data, with particular utility in Natural Language Processing (NLP) (to capture grammatical structure; [2]), and in molecular biology to support functional and structural classification of proteins [10]. The challenge is to encapsulate the problem structure in a form which admits rapid evaluation of similarity. Trees are of value if the problem is inherently hierarchical, particularly if parent nodes support some abstraction from their children. In this work, we view bacterial promoter prediction (see section 4) as an NLP style parsing task, mapping a genomic sentence to its parse tree with respect to a grammar defined by the molecular structure of the binding protein and its associated binding sites. Genomic similarity is then defined over the resulting tree structures. In contrast to earlier node-labelled trees, these representations are also edge-weighted, with numerical values assigned to each arc connecting nodes within a tree, and we hae developed a number of new kernels which allow evaluation of this edge-weighted similarity. These are presented here in the context of Support Vector Machine (SVM) classification for the bacterial promoter problem, but they have more general application in phylogenetic analysis, molecular structure classification, and general network problems.

This paper is organised as follows: Section 2 presents a brief introduction to the SVM, tree kernel methods and our notation, supporting the introduction of our new kernels in Section 3. Section 4 presents the bacterial promoter prediction

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

problem and the results of our experiments with the new kernel methods. We conclude in Section 5 with discussion of these results and the prospects for refinements to our methods.

2 Tree Kernels and the Support Vector Machine

The binary SVM [1] is a linear classifier taking a set of labelled inputs $(x_i, y_i), i = 1 \dots m$, where x_i is a training point drawn from an appropriate input space (usually $x_i \in \mathbb{R}^n$) and $y_i \in \{-1, +1\}$ is the associated class label. SVM training determines a decision boundary $w^t x_i + b = 0$, optimal in maximizing the margin of separation between the classes. As is well known [1], data appear in the training problem only through their inner products, so structured data can be incorporated through task specific kernels such as those proposed here.

For consistency, we adopt the definitions and notation of [7], and the taxonomy of [6]. A tree T is a directed, connected, acyclic graph in which each vertex (node) other than the root has an in-degree of one. The in-degree $d^-(n)$ is the number of edges directed into node n, while the out-degree $d^+(n)$ is the number of edges directed outwards. Any node with out-degree zero is a leaf. Children of node v are the nodes connected to edges directed out from v, so node v has $d^+(v)$ children. Here, each tree is assumed *structured:* the children of node v have a fixed ordering, and can be referenced as $ch_1(v), \dots, ch_{d^+(v)}(v)$.

Tree kernels compute similarity between trees through comparison of their consituent structures, although there may be differences in the fragments chosen. Moschitti [6] labelled these choices as *Subtrees* (STs) and *Subset Trees* (SSTs). An ST is simply a node and all of its descendents. An SST need not include the descendents; if a descendent node v is excluded from the SST, then all of v's children are also excluded. The SST is a more general structure, and provides a domain of potential SSTs whose size is exponential in the number of nodes in a tree. The size of the ST domain for a given tree is linear with the number of nodes in the tree, as there is only one possible ST for each node.

The SST kernel originated in NLP studies [2]. The kernel is defined as $K(T_1, T_2) = h(T_1) \cdot h(T_2)$, where h is an n-dimensional vector of the $h_i(T)$ – the number of occurrences of the i-th SST present in the training data. The vector $h(T)$ has the potential to be extremely large, but its creation is only implicit (see below). To compute K, we define N_1 and N_2 as the set of nodes in T_1 and T_2 respectively. An indicator function $I_i(n)$, is created to return 1 if the i-th SST is found rooted at node n, otherwise 0. By traversing the nodes in the tree, a count of the occurrences of the i-th SST in tree T_1 can be calculated as $\sum_{n \in N_1} (I_i(n))$ i.e. the count of the nodes at which SST i is rooted.

It follows that the kernel can then be calculated by checking each combination of pairs of nodes between the two trees for matching SSTs according to

$$K(T_1,\ T_2) = h(T_1) \cdot h(T_2) = \sum_i h_i(T_1) h_i(T_2) = \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \sum_i I_i(n_1) I_i(n_2)$$

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

To remove the need for the creation of the vector containing all of the SSTs, the explicit use of the indicator function can be removed by defining a function C so that

$$\sum_i I_i(n_1)I_i(n_2) = C(n_1, n_2).$$

Here we will generalise the $C(n_1, n_2)$ of Moschitti [6], using a parameter σ to include both SST ($\sigma = 1$) and ST ($\sigma = 0$) kernels:

$$C(n_1, n_2) = \begin{cases} 0 & n_1 \neq n_2 \\ 1 & (n_1 = n_2) \wedge (d^+(n_1) = d^+(n_2) = 0) \\ \prod_{k=1}^{d^+(n_1)} (\sigma + C(ch_k(n_1),\ ch_k(n_2))) & \text{otherwise} \end{cases}.$$

The recursive nature of this function ensures a walk down the nodes in the tree, increasing the count until mismatching nodes are found. By pre-filling a matrix of the $C(n_1,\ n_2)$ values, the SST kernel can be calculated in $O(|N_1||N_2|)$ time. When $\sigma = 0$, the subtree kernel, the function will evaluate to zero if any pair of nodes in the subtree do not match.

3 New Kernels

3.1 Edge Weighted Tree Kernel

This method generalises fragment comparison to encompass soft matching of edge weights between respective parent and child nodes. Two parent-child pairs (p_1, c_1) and (p_2, c_2) are considered equal if parent nodes are equal ($p_1 = p_2$) , child nodes are equal ($c_1 = c_2$) and the edge weights are equal up to some threshold parameter t: $|d(p_1, c_1) - d(p_2, c_2)| \leq t$. Evaluation can be introduced by modifying the third, recursive case in the definition above, so that if n_1 and n_2 are the same and are not leaves,

$$C(n_1, n_2) = \prod_{k=1}^{d^+(n_1)} (\sigma + D(n_1, ch_k(n_1), n_2, ch_k(n_2)))$$

where

$$D(p_1,\ c_1, p_2,\ c_2) = \begin{cases} C(c_1,\ c_2) & \text{if } |d(p_1,\ c_1) - d(p_2,\ c_2)| \leq t \\ 0 & \text{if } |d(p_1, c_1) - d(p_2, c_2)| > t \end{cases}$$

Essentially, this modification introduces some tolerance within which we accept the similarity of the fragments as sufficient, supporting some modest generalisation of the earlier sructure.

3.2 Path-Spectrum Tree Kernel

Edge weights may be used to assign a cost to each path between a pair of nodes within the tree. Here we choose to compare the structure of a tree through the spectrum of weighted paths from root to leaf that exist given the domains of

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

node values, edge values and the rules and constraints placed upon relationships between nodes in a tree. As with string spectrum kernels [5], the *Path Spectrum Kernel* (PSK) allows comparison *and* identification of important spectral elements. This work differs from earlier tree measures [3] in considering edge as well as node values, and in operating over paths rather than tree fragments. The PSK is usually more efficient than the SST kernels, as the path set is commonly of far lower cardinality than the fragment set.

Once the spectrum S has been defined, a tree T can be mapped to a feature vector with the feature map $\Phi(T) = (\phi_p(T))_{p \in S}$, where $\phi_p(T)$ is the number of times that the path p occurs in T, so that the i-th element of $\Phi(T)$ is the count of occurrences of the i-th path in the tree: $\Phi(T) = \langle \phi_{S_1}(T)\ ,\ \phi_{S_2}(T)\ ,\ \ldots,\ \phi_{S_n}(T) \rangle$. This vector can be created by traversing the path set and determining the index of the path within the spectrum, a depth-first-search with linear complexity. Path comparison has complexity $O(D)$, where D is the maximum tree depth. After sorting the spectrum elements, searching a spectrum of size S for a path is an $O(D \log S)$ operation, while performing this operation for each path in the tree is $O(PD \log S)$, where P is the number of leaf nodes (number of paths) in the tree. As before, the threshold t limits exploration of the spectrum.

4 Experiments

The tree kernels of section 3 were applied to the biological task of gene promoter prediction in bacterial genomes. This section outlines the problem, data, methods and results obtained using the new kernels with the SVM classifier to address this task. More details of this task may be found in the supplementary material at: http://eprints.qut.edu.au/67877.

The promoter prediction problem focuses on binding sites for the σ^{70} RNAP complex in *E. coli*, characterised by specific motifs at (approximately) known locations (the so-called -10 and -35 hexamers) relative to the *Transcription Start Site (TSS)*. Additional proteins may be coupled to the complex, each contributing to the binding process, forming the 'phrase' which motivates the tree kernel approach. Our approach involves a combination of tokens based on the binding sites for each of the components of the RNAP complex, each labelled according to their expected location. Valid interactions between element motifs are shown in figure 1, where edge weights indicate allowable gaps between motifs (motif constraints). Experiments were conducted with sequences of length 151 base pairs, indexed around a reference nucleotide at position 101. Positive sequence examples contained an experimentally identified TSS at the reference position, while negative sequence examples did not (figure 1). In essence, the aim of the classifier is to determine whether or not the nucleotide at the reference position is a TSS. Some 609 of the entries available were used to form positive sequences. Coding region DNA, i.e. that within an identified gene, was used as the negative data on the basis that a gene coding region should not contain a TSS. For each TSS site, a set of sequences referred to as the sliding window sequences was obtained. Each set consists of 201 sequences, each 151bp in length

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

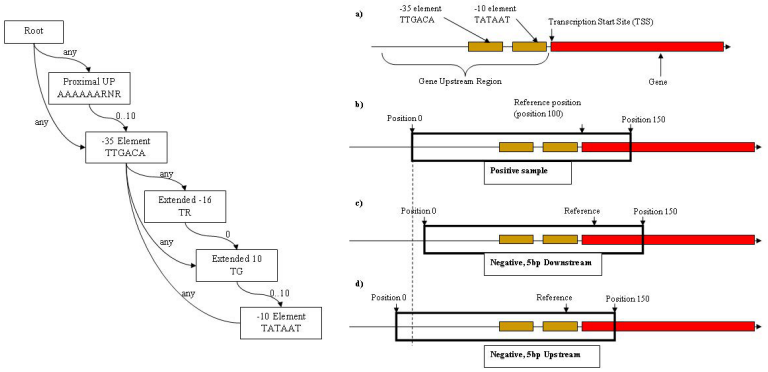

Fig. 1: Left: Relationships between promoter motifs; nodes are motifs, edges are gaps. Right: Sequence structure: a) Typical gene structure b) Positive sequence, reference position at true TSS c),d) Example negative sequences (shifted 5bp downstream, 5bp upstream)

with the reference position being 100bp upstream of the actual TSS for sequence 1, with negative sequences obtained by moving the windows downstream, shifting the reference position each time. Parse trees were based on the biological rules and constraints documented in figure 1.

4.1 Results and Discussion

Results are shown below in table 4.1 for the region prediction problem, i.e. deciding whether or not a sequence contains a true TSS. Baseline performance is provided by a spectrum kernel [5] seeded with confirmed promoter motifs, which produces performance fractionally superior to that of [4], at 11.6% error the best reported for this problem, although variations in data sets make direct comparison difficult. The mismatch parameter m permits approximate string matching at the motif level up to m positions. All results are based on ten fold cross validation repeated ten times. For clarity, variance is not shown but all standard deviations are well under 5%. The Weighted ST kernel proved

Kernel	Method	Error	Recall	Precision
Dictionary	$m = 1$	14.5	83.7	86.9
Dictionary	$m = 3$	8.7	90.5	92.0
Weighted ST	$-35, -10$, $t = 0$	19.9	71.6	86.2
Weighted PSK	Best 600 (Gain Ratio) attributes $t = 4$	14.4	83.7	87.1
Weighted PSK	(with Dictionary; $m = 2$)	4.75	96.56	94.09
Weighted PSK	(with Dictionary; $m = 4$)	4.02	98.36	93.9

ineffective for this problem, with the simplest structure performing best, but at levels well under that provided by the dictionary kernel. Application of the Weighted PSK is far better, but in its most general form it remains inferior to the simple dictionary approach, the loss of sensitivity due to acceptance of inferior tokens seemingly outweighing the advantages provided by the phrasal structures.

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Efficient use of the path spectrum involves a tradeoff between the cardinality of the spectrum and its representational effectiveness. Here we used the best 600 paths based on gain ratio. Seeding the lexicon with confirmed motifs as in the baseline overcame earlier limitations, with performance clearly superior to the baseline for modest additional computational effort.

5 Conclusions

In this paper we have introduced two new tree kernel methods for phrase like structures, and demonstrated the effectiveness of the more sophisticated of the two in an important problem in bioinformatics, producing state of the art results for region based TSS prediction. Earlier kernel methods [4] provided substantial advantages over the commonly used position weight matrix predictors [8], [9] – reducing the FP rate by a factor of more than 20 – but the specialised skills needed to develop the models and the computational overhead required limited their direct adoption. Dictionary and tree kernel hybrids such as those introduced here do not depend upon an ensemble and allow the direct insertion of biological domain knowledge during model creation, thereby allowing frequent adaptation in response to experimental advances. In forthcoming work, we will publish the results of these approaches for the location specific inference of the TSS, and explore their peformance in limiting the false positive results for high levels of recall which plague the traditional approaches.

References

[1] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121-167, 1998. ISSN 1384-5810.

[2] M. Collins and N. Duffy. Convolution kernels for natural language. *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

[3] T. Kuboyama, K. Hirata, K. Hisashi, K. Aoki-Kinoshita, and H. Yasuda. A Spectrum Tree Kernel. *Transactions of the Japanese Society for Artificial Intelligence*, 22(2), 140-147, 2007.

[4] J.J. Gordon, M.W. Towsey, J.M. Hogan, S.A. Mathews and P. Timms. Improved Prediction of Bacterial Start Sites. *Bioinformatics*, 22(2), 142-148, 2006.

[5] C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: a string kernel for svm protein classification. *Pacific Symposium on Biocomputing*, 7:566-575, 2002.

[6] A. Moschitti. Making tree kernels practical for natural language learning. *Proceedings of the 11th International Conference of European Association for Computational Linguistics (EACL)*, 2006.

[7] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge University Press, 2004.

[8] G. Stormo. DNA binding sites: representation and discovery. *Bioinformatics*, 16(1), 16-23, 2000.

[9] TJ. Todt, M. Wels RS. Bongers, RS. Siezen, SAFT. van Hijum, M. Kleerebezem Genome-Wide Prediction and Validation of Sigma70 Promoters in Lactobacillus plantarum WCFS1 *PLOS ONE* DOI: 10.1371/journal.pone.0045097, 2012.

[10] Y. Yamanishi, F. Bach, and J.-P. Vert. Glycan classification with tree kernels. *Bioinformatics*, 23 (10):1211-1216, 2007.