doi:10.2533/chimia.2011.10 Chimia 65 (2011) 10–13

Glycoinformatics: Data Mining-based Approaches

Hiroshi Mamitsuka*

Abstract: Carbohydrates or glycans are major cellular macromolecules, working for a variety of vital biological functions. Due to long-term efforts by experimentalists, the current number of structurally different, determined carbohydrates has exceeded 10,000 or more. As a result data mining-based approaches for glycans (or trees in a computer science sense) have attracted attention and have been developed over the last five years, presenting new techniques even from computer science viewpoints. This review summarizes cutting-edge techniques for glycans in each of the three categories of data mining: classification, clustering and frequent pattern mining, and shows results obtained by applying these techniques to real sets of glycan structures.

Keywords: Data mining · Frequent subtrees · Glycan structures · Machine learning · Probabilistic models

1. Introduction

Oligosaccharides and glycans are major cellular macromolecules associated with a variety of important biological phenomena, including antigen-antibody interaction[1] and cell fate controlling.[2] Proteins and DNAs consist of twenty types of amino acids and four types of nucleotides, respectively. Likewise glycans have building blocks, called monosaccharides, which however are more diverse, the major ones being fructose, galactose, glucose and mannose.[3] The uniqueness of carbohydrates is in the connection of monosaccharides, where two or more monosaccharides can connect to one monosaccharide without any cycle, forming branch-shaped extensions. In a computer science sense, glycans can be *trees*, consisting of *nodes* and *edges* (which connect nodes), corresponding to monosaccharides and chemical linkages, respectively. Moreover glycans are *rooted directed ordered trees* because i) a glycan connects to a protein (an amino acid) by only one monosaccharide, called the *root*, ii) connections are directed from the root to *leaves*, meaning that two connected monosaccharides can be a *parent* (closer to the root) and a *child* (closer to a leaf), *ancestors* and *descendants* being able to be defined in a similar manner, and iii) monosaccharides (children) can be ordered by carbon numbers attached to linkages to another monosaccharide (parent), meaning that children are ordered from the oldest sibling to the youngest sibling. Hereafter a *subtree* means a node in a tree and its all descendants and edges from that node to leaves, while a *supertree* of tree T is a tree having T as a subtree. The unique structure of carbohydrates makes them different from other macromolecules such as proteins and DNAs which are simpler sequences of building blocks. Furthermore this uniqueness has made it hard to determine the tree structures of glycans experimentally by which the size of glycan structure databases has been kept small and its speed to develop has been forced to be slow. However, due to the development of carbohydrate research, the current number of structurally different glycans in a major database on glycans reaches more than ten thousands, which will be further increased in the near future by using high-throughput techniques.[4] Thus in glycoinformatics, developing efficient approaches for mining rules or patterns from trees and applying the approaches to a glycan database would be reasonable and promising to find biological significance embedded in glycan structures.[5,6]

*Correspondence: Prof. H. Mamitsuka
Kyoto University
Institute for Chemical Research
Bioinformatics Center
Gokasho, Uji 611-0011, Japan
Tel.: +81 774 38 3023
Fax: +81 774 38 3037
E-mail: mami@kuicr.kyoto-u.ac.jp

2. Mining from Glycan Structures: Data Mining Techniques for Trees

In general, current approaches for mining from data or machine learning can be classified into roughly three types: i) classification (or supervised learning), ii) clustering (or unsupervised learning) and iii) frequent pattern mining.[7,8] Here we briefly explain the difference between these concepts, under which the input is always called *examples* (which are in our case rooted directed ordered trees). In classification, examples are labeled or class labels are attached to examples, and the purpose is to make a classifier/predictor which can predict a class to be assigned to a newly given example. This is supervised learning. On the other hand, in clustering, examples are not labeled, and the purpose here is to assign examples to some groups by which we can see the similarity between examples, such as if two examples are in the same group, they are similar. This is unsupervised learning, which can give labels to examples, while labels are given in supervised learning. In frequent pattern mining, the input is unlabeled examples (so in some sense this is unsupervised learning, but in this review we do not categorize frequent pattern mining into unsupervised learning). The purpose is to find patterns which appear a larger number of times than a prefixed number, by which we can see what patterns appear frequently.

2.1 *Classification over Trees*

Currently the most popular approach in supervised learning is so-called support vector machines (SVMs), in which we use a kernel function that represents a similarity between two examples.[9] The objective here is to find a classifier which can separate examples in one class from

those in the other class (if class labels are binary). The simplest way is to use a linear function, corresponding to using an inner product as a kernel function. However, it would be hard to separate examples depending upon class labels by using a simple linear function. This means that it is important to design a good kernel function by which we can separate examples easily. For the case that examples are trees, a kernel function for computing a similarity between two input trees has been already proposed.[10] The idea behind the 'tree' kernel is to check subtrees, which appear in the two input trees commonly, and if they are bigger and/or the number of common subtrees is larger, the two input trees should be more similar. This can be computed in a very efficient manner by using dynamic programming, and a tree kernel specialized for glycans was also proposed.[11]

2.2 *Clustering: Probabilistic Models for Trees*

There are a variety of approaches in unsupervised learning. In this review, we focus on probabilistic models, which are models with probability parameters, to be estimated from given data. Rather than clustering, probabilistic models allow binary classification if binary classes are like the examples in question and others. This case, after training a probabilistic model from examples in question, we can compute the likelihood for any given example, showing how likely the given example can be in the class in question. A standard probabilistic model for time-series data or sequences is the hidden Markov model (HMM), which has been used in a lot of applications, including speech recognition,[12] natural language processing[13] and analyzing biological sequences, *e.g.* amino acid sequences.[14] A HMM can be defined by a state transition diagram, in which states are connected by edges. A HMM (or its state transition diagram) has two types of probability parameters, one being letter generation probabilities attached to states to generate letters, *i.e.* amino acid types, and the other being state transition probabilities attached to edges. A standard assumption on the Markov process is the first-order Markov property, which is very simple for sequences and means that the current state depends upon only one state away. Thus given a sequence, we can just repeat the following two steps, from left to right on the sequence: we first generate the corresponding letter, *i.e.* amino acid, at a state with a letter generation probability and then transit to another state with a state transition probability.

Using these probabilities, we can compute the likelihood that a given sequence is generated under a HMM (or its state transition diagram). Training (or learning) probability parameters of HMM is usually based on maximum likelihood, which means that probability parameters are estimated so that the likelihoods of given sequences are maximized. There are a variety of extensions of HMMs, major ones being context free grammars[15] and tree grammars[16] to deal with complex dependencies in sequences. Another direction is for trees, the first trial being the hidden tree Markov model (HTMM)[17] in which left-to-right on a sequence is simply applied to parent-to-child over a tree. This model can be applied to glycans. However, glycans are rooted directed ordered trees, while ordered children are not considered in HTMM at all. Thus the ordered tree Markov model (OTMM)[18,19] and the probabilistic sibling-dependent tree Markov model (PSTMM)[20,21] have been developed for glycans.

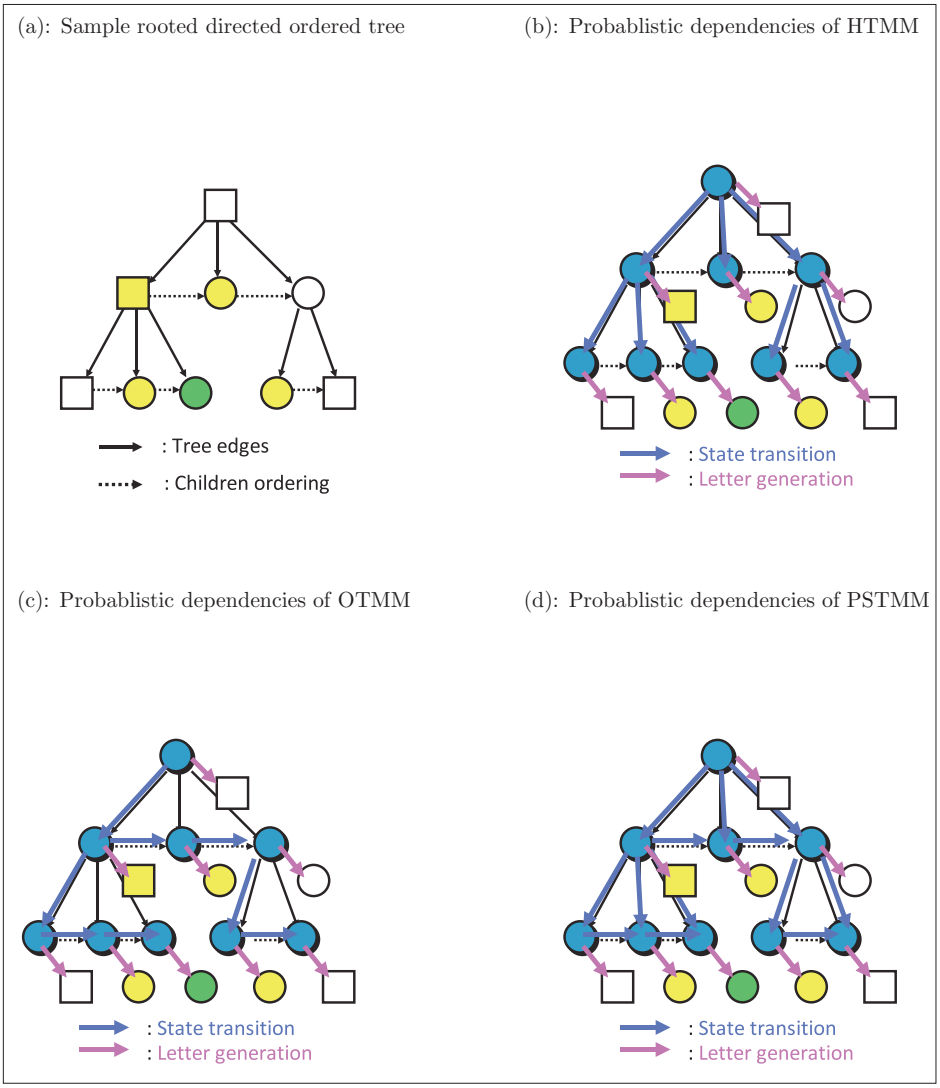

Fig. 1. (a) A sample rooted directed ordered tree and probabilistic dependencies of (b) HTMM, (c) OTMM and (d) PSTMM

Fig. 1 shows the difference between HTMM, OTMM and PSTMM. Fig. 1 (a) shows a sample rooted directed ordered tree, where the top square means the root, solid lines show directed edges (chemical linkages) from parents to children and dotted lines show ordered children. Fig. 1 (b) shows a HTMM over the tree in (a), where you can see that thick lines of state transitions are always directed from parents to children. Fig. 1 (c) shows an OTMM over the tree in (a), where you can see that thick lines of state transition are from parents to children if the children are the oldest; otherwise thick lines are from older siblings to younger siblings. That is, OTMM considers ordered children as well as parent-to-child dependencies *via* parents and the oldest siblings. Fig. 1 (d) shows a PSTMM over the tree in (a), where you can see that thick lines of state transitions are always from parents to children as well as from older siblings to younger siblings. This means that PSTMM always considers both parent-to-child dependencies and dependencies between ordered children. Thus we can see that PSTMM is the most complex and well-considered

model for glycans. However we need to think about other points, such as computational complexity of probabilistic models. HTMM is a simple modification of HMM from left-to-right to parent-to-child, meaning that these two models share the same time and space complexities, which is roughly $O(n^3)$ where n is the size of given sequences or trees and the state transition diagram is assumed to be the complete graph. This good property of HTMM is also shared by OTMM, which also keeps the same complexities. However, the time and space complexities of PSTMM are higher and roughly $O(n^4)$. This difference is computationally significant, because the state transition diagram can be a left-to-light one by which the complexity of OTMM can be $O(n^2)$ but PSTMM must keep $O(n^3)$. Another issue is that the number of glycans we can have is always relatively small, meaning that a complex model is likely to overfit to training data. In fact, ref. [19] shows that the predictive performance of PSTMM is high for training data but very low for test data, while that of OTMM is high for both training and test data, because the high complexity of PSTMM makes this model overfit to the training data. These results imply that OTMM is the most appropriate probabilistic model for glycans. On the other hand, PSTMM was applied to aligning multiple glycans by modifying it into a model called ProfilePSTMM,[22] which is a similar modification of HMM to ProfileHMM. This modification reduces the complexity of the original PSTMM by which ProfilePSTMM can avoid the overfitting issue.

2.3 *Mining Frequent Subtrees*

Classification and clustering are very useful in that they can give some information to examples. On the other hand, if classifiers or probabilistic models are complex, it is hard to obtain any rules from given examples. Mining frequent patterns is useful to find rules or patterns embedded in given examples, and these obtained patterns can further work for classification and clustering. Thus mining frequent patterns have been developed for different types of examples, such as items,[8] trees[23] and graphs.[8] Here we focus on trees or glycans, and in fact, conserved patterns have already been reported in glycans,[24] implying that similar subtree patterns could still be hidden and not yet found. Given a set of trees, the number of trees having a certain subtree is called the *support* of the subtree. A subtree is *frequent* if its support is equal to or higher than a prefixed number, which is called *minimum support* or *minsup*. In general, we can have a huge number of frequent subtrees, which are at the same time redundant, since if a subtree is frequent, its all subtrees are frequent. This means that frequent subtrees are sometimes so small and meaningless. To overcome these issues in mining frequent subtrees, we present a new method, which has two important features: 1) controlling the number of outputted frequent subtrees and 2) using hypothesis testing to output only significant subtrees.[23] That is, we first reduce the number of frequent subtrees by step 1 and then remove meaningless frequent subtrees by step 2. Step 1 was realized by the concept, called α-closed frequent subtrees, which means that a frequent subtree T is not outputted if the support of its supertree is larger than or equal to α × the support of T, where α takes a value between zero and one. That is, the intuitive idea of α-closed frequent subtrees is that frequent subtree T is not outputted if the support of its supertree has a value similar to the support of T. Step 2 needs a control dataset, which is for example randomly generated, and the support of frequent subtrees in the control dataset is computed. Step 2 then runs Fisher's exact test to retrieve only the frequent subtrees, which appear well in the given dataset but not so frequently in the control dataset. Final results are ranked by p-values of Fisher's exact test.

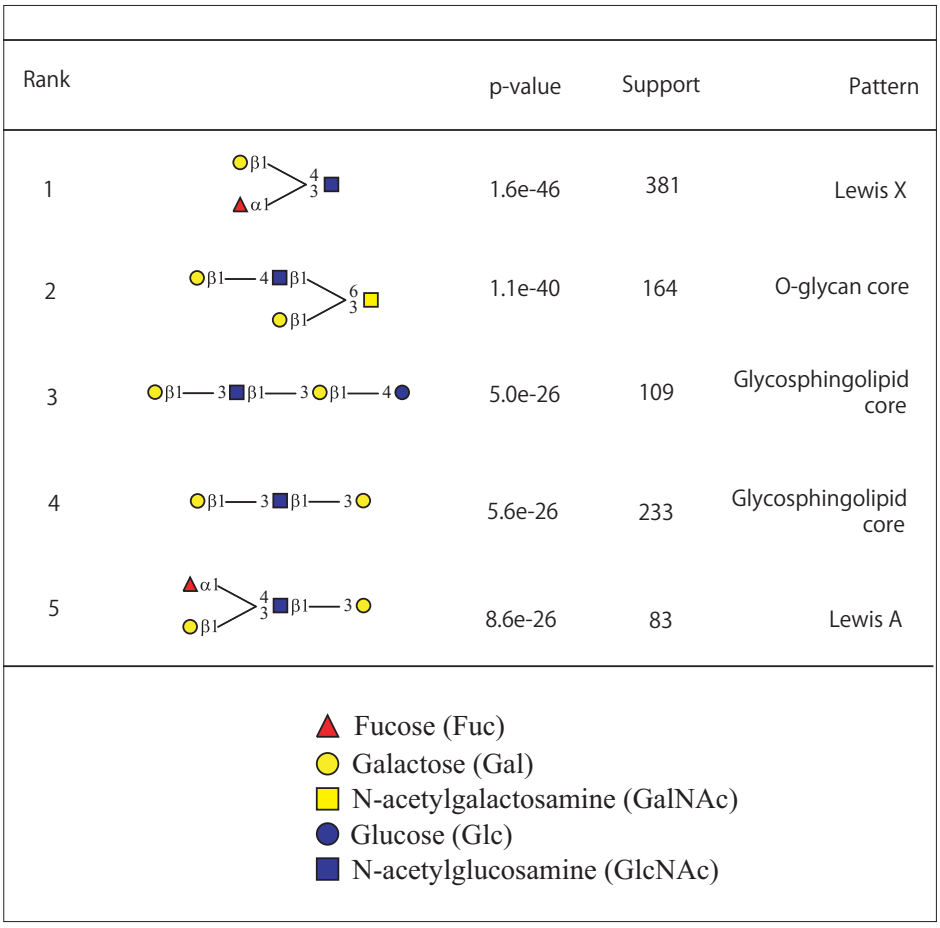

Fig. 2.

Fig. 2 shows the top five patterns obtained by running this approach over all glycans in a glycan database, revealing that each of the five patterns corresponds to a known, conserved pattern of glycans. Furthermore you can see that these patterns are not ranked by their supports but by their p-values.

Obtained patterns can be used to generate a binary feature vector of each glycan, by assigning 1 to a feature if this glycan has the pattern corresponding to this feature; otherwise zero. We can then apply these feature vectors to a classification problem of glycans by using a linear kernel and SVM. Fig. 3 shows the performance (shown by AUC, Area Under the ROC curve, a standard measure in machine learning) of this approach (displayed by 'Proposed Method'), outperforming those using three well-known tree kernels. This result shows that patterns are very useful even for classification, mainly because patterns are obtained from given data, meaning that patterns can be changed depending upon given data while tree kernels cannot.

Finally Table 1 shows a summary of data mining techniques for glycans or rooted directed ordered trees which we have introduced in this review.

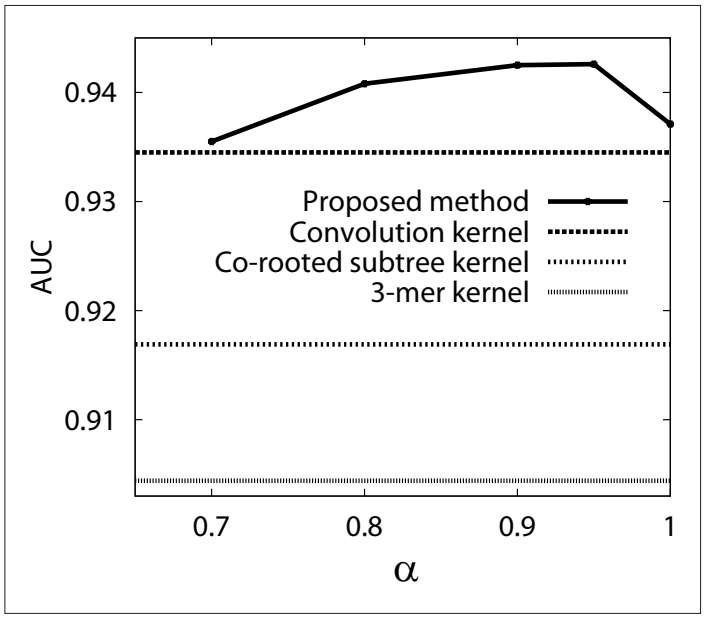

Fig. 3.

Table 1.

Technique	References
Supervised learning: kernel-based method	
Glycan kernel	[11]
Unsupervised learning: probabilistic-models	
OTMM	[20,21]
PSTMM	[18,19]
Profile PSTMM	[22]
Frequent pattern mining	
Frequent subtree mining	[23]

3. Conclusion

We have shown three types of problem settings in data mining and latest techniques for glycans or rooted directed ordered trees in each type. We have further shown that the techniques are proved to be useful for real glycans. In particular, mining frequent patterns obtained well-known patterns embedded in a glycan database automatically and the obtained patterns outperformed SVMs with tree kernels in a supervised learning setting. This implies that using frequent patterns is very useful and promising to further explore data mining approaches for glycans.

Received: August 31, 2010

[1] A. Varki, R. Cummings, J. Esko, H. Freeze, G. Hart, J. Marth, 'Essentials of glycobiology', CSHL Press, New York, USA, **1999**.
[2] P. Stanley, *Curr. Opin. Struct. Biol.* **2007**, *17*, 530.
[3] J. D. Marth, *Nat. Cell Biol.* **2008**, *10*, 1015.
[4] S. J. Haslam, S. J. North, A. Dell, *Curr. Opin. Struct. Biol.* **2006**, *16*, 584.
[5] R. Raman, S. Raguram, G. Venkataraman, J. C. Paulson, R. Sasisekharan, *Nat. Methods* **2005**, *2*, 817.
[6] H. Mamitsuka, *Drug Discov. Today* **2008**, *13*, 118.
[7] C. M. Bishop, 'Pattern recognition and machine learning', Springer, New York, USA, **2006**.
[8] J. Han, H. Cheng, D. Xin, X. Yan, *Data Min. Knowl. Disc.* **2007**, *15*, 55.
[9] J. Shawe-Taylor, N. Cristianini, 'Kernel methods for pattern analysis', Cambridge University Press, Cambridge, UK, **2004**.
[10] M. Collins, N. Duffy, 'Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics', *ACL,* **2002**, p. 263.
[11] Y. Yamanishi, F. Bach, J-P. Vert, *Bioinformatics* **2007**, *23*, 1211.
[12] L. R. Rabiner, B. H. Juang, *IEEE ASSP Mag.* **1986**, *3*, 4.
[13] 'Foundations of Statistical Natural Language Processing', Eds. C. Mannig, H. Schutz, MIT Press, **1999**.
[14] 'Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids', Eds. R. Durbin, S. R. Eddy, A. Krogh, G. Mitchison, Cambridge University Press, **1998**.
[15] K. Lari, S. J. Young, *Comput. Speech Lang.* **1990**, *4*, 35.
[16] N. Abe, H. Mamitsuka, *Mach. Learn.* **1997**, *29*, 275.
[17] M. Diligenti, P. Frasconi, M. Gori, *IEEE T. Pattern Anal.* **2003**, *25*, 519.
[18] K. F. Aoki, N. Ueda, A. Yamaguchi, M. Kanehisa, T. Akutsu, H. Mamitsuka, *Bioinformatics* **2004**, *20*, i6.
[19] N. Ueda, K. F. Aoki-Kinoshita, A. Yamaguchi, T. Akutsu, H. Mamitsuka, *IEEE T. Knowl. En.* **2005**, *17*, 1051.
[20] K. Hashimoto, K. F. Aoki-Kinoshita, N. Ueda, M. Kanehisa, H. Mamitsuka, in 'Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006)', Eds. T. Eliassi-Rad, L. H. Ungar, M. Craven, D. Gunopulos, ACM Press, **2006**, pp. 177.
[21] K. Hashimoto, K. F. Aoki-Kinoshita, N. Ueda, M. Kanehisa, H. Mamitsuka, *ACM Trans. Knowl. Discov. Data* **2008**, *2*, Article 6.
[22] K. F. Aoki-Kinoshita, N. Ueda, H. Mamitsuka, M. Kanehisa, *Bioinformatics* **2006**, *22*, e25.
[23] K. Hashimoto, I. Takigawa, M. Shiga, M. Kanehisa, H. Mamitsuka, *Bioinformatics* **2008**, *24*, i167.
[24] P. M. Lanctot, F. H. Gage, A. J. Varki, *Curr. Opin. Struct. Biol.* **2007**, *11*, 373.