Views: Fundamental Building Blocks in the Process of Knowledge Discovery

Hideo Bannai
Human Genome Center, Institute of Medical Science
University of Tokyo, 4-6-1 Shirokanedai
Minato-ku, Tokyo 108-8639, Japan
bannai@ims.u-tokyo.ac.jp

Yoshinori Tamada
Department of Mathematical Sciences
Tokai University, 1117 Kitakaname
Hiratuka-shi, Kanagawa 259-1292, Japan
tamada@hunter.ss.u-tokai.ac.jp

Osamu Maruyama
Faculty of Mathematics, Kyushu University
Kyushu University 36,
Fukuoka 812-8581, Japan
om@math.kyushu-u.ac.jp

Kenta Nakai
Human Genome Center, Institute of Medical Science
University of Tokyo, 4-6-1 Shirokanedai
Minato-ku, Tokyo 108-8639 Japan
knakai@ims.u-tokyo.ac.jp

Satoru Miyano
Human Genome Center, Institute of Medical Science
University of Tokyo, 4-6-1 Shirokanedai
Minato-ku, Tokyo 108-8639 Japan
miyano@ims.u-tokyo.ac.jp

Abstract

We present a novel approach to describe the knowledge discovery process, focusing on a generalized form of attribute called *view*. It is observed that the process of knowledge discovery can, in fact, be modeled as the design, generation, use, and evaluation of views, asserting that views are the fundamental building blocks in the discovery process. We realize these concepts as an object oriented class library and conduct computational knowledge discovery experiments on biological data, namely the characterization of N-terminal protein sorting signals, yielding significant results.

Introduction

Fayyad *et al.* (1996) describes the KDD process as "the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data", and gives an outline of the basic KDD process consisting of: 1) data selection 2) data preprocessing 3) data transformation 4) data mining (hypothesis generation) 5) hypothesis interpretation/evaluation 6) knowledge consolidation.

Considering that a large portion of the knowledge discovery process almost always consists of a *trial-and-error interaction* between the domain expert and the problem (Cheeseman & Stutz 1996), there is a strong need for facilitating the *human intervention* (Langley 1998) to discovery systems, such as: incorporating ingenious "tailor-made" attributes designed by experts, integrating various expert knowledge as well as experts' intuitions concerning the domain, and assisting in the evaluation of the knowledge which is obtained. Also, another aspect which is recognized as a key in the KDD process, is the generation or discovery of "good" attributes/features (Motoda 1999) which help explain the data.

Much work has been done on these topics: For example, CLEMENTINE (Khabaza & Shearer 1995) is a successful commercial tool, focusing on human intervention and extensibility, integrating multiple modeling and discovery algorithms with tools for data access, data manipulation and preprocessing, visualization and reporting. Methods to extend the space of available attributes and/or features, such as *constructive induction*, *feature construction*, have appeared frequently in the machine learning literature (Michalski 1983; Matheus & Rendell 1989; Bloedorn & Michalski 1998).

Each has been successful in its own aim, but more general concepts would help to better understand the process of knowledge discovery as a whole. In this paper, we give a more mathematical abstraction of the knowledge discovery process by focusing on a generalization of attribute, named *view* (Maruyama *et al.* 1998; 1999), which defines a specific way of looking at, or understanding the given entities. Views are essentially functional attributes which, given an entity, returns a value representing a certain aspect of the entity. Formal definitions for these concepts are given in the next section.

We observe that steps of the KDD process can be described as the design, generation, use, and evaluation of views. We also define *operations* on views, as a way to generate new views from existing views, inspiring the diversity of view *design* by experts. While the careful design of views by experts offers an elegant interface for human intervention to the discovery process, we also show that views can describe a general framework, analogous to constructive induction techniques, which can (at least partially) automate the generation of new views.

From these properties, we conclude that views are fundamental building blocks of the knowledge discovery process. We have realized the notions we have defined, as a software library of views and view operators named HYPOTHESISCREATOR($\mathcal{HC}$), hoping to accelerate this process. We give an account of a series of computational experiments conducted on biological data using $\mathcal{HC}$, demonstrating how our ideas are applied to real world applications.

Basic Concepts

The initial idea of view was presented by Maruyama *et al.* (1998; 1999), whose purpose was to facilitate human intervention and attribute generation in the discovery process. In this section, we further refine the definitions and give several examples, trying to show how this simple, mathematically trivial idea fits into the discovery process.

Definition 1 (entity) An *entity* is anything which can be identified as an individual (i.e. an object). We shall call the set of entities which is under consideration, the *entities of interest* E. □

For example, we consider proteins e_p as entities, and the set of plant proteins E_P as the entities of interest. Entities can be distinguished from each other by definition, but *how* they differ is ascertained through various attributes they possess. An attribute, or feature, for an entity is generalized as follows:

Definition 2 (view, viewscope) A *view* is a function $v : E \rightarrow R$ over entities. Let $v(e)$ denote the value that a view returns for a certain entity e. The range R of v is called the *range set* of v. For convenience, we call a set of possibly infinite views $V = \{v_1, v_2, \ldots\}$ a *viewscope*. Similarly, let $V(e) = \{v_1(e), v_2(e), \ldots\}$. □

A view returns a value which is expected to represent some aspect of the given entity. When parametric functions are regarded as views, we call them *parametric views*. A parametric view together with the space of parameters can define a viewscope.

Example 1 An *amino acid sequence view* v would return the amino acid sequence for a particular protein. e. g. if $e_p \in E_P$ is the ATP17 protein of Saccharomyces cerevisiae (Baker's yeast), $v(e_p)$ = "MIFKRAVSTL. . .". □

The first step in the KDD process is data selection. Data can be regarded as the set of entities, accompanied by initially given views. For example, if we are given a table of items (rows) and their attributes (columns), the entity set would consist of the items, and the initially given views would be the mapping from each item to their attributes in the table (one view for each column). If we can design a Boolean valued view which returns *true* for entities we want/need, and *false* for entities we do not want/need, we can create a subset of the entities by filtering the original set using this view.

Definition 3 (view operator) We call any function over views (or viewscopes) and/or entities, which returns a view (or viewscope), a *view operator*. □

Definition 4 (view design) Views and view operators are combined to create new views. We call the structure of such combinations, the *design* of the view. □

When human experts design views, they can embed their knowledge, intuitions, and ideas. Views provides an interface for human intervention to the discovery process.

There are view operators which are not dependent on the views which they operate upon. These view operators may be defined by a function over range sets.

Definition 5 (range set based view operator) A *range set based view operator* Ψ is induced by a function $\psi : R \rightarrow R'$ over a range set R. With ψ and view v with range set R:

$$\Psi[v] : E \rightarrow R' \equiv \psi \circ v \equiv E \xrightarrow{v} R \xrightarrow{\psi} R' \qquad (1)$$

where $\Psi[v]$ denotes the new view induced by the application of operator Ψ to the range set of v. Similarly, for viewscope $V = \{v_1, v_2, \ldots\}$, we define $\Psi[V] \equiv \{\Psi[v_1], \Psi[v_2], \ldots\}$. Obviously, a parametric view operator will result in a parametric view. □

The next steps in the KDD process is the preprocessing, and transformation of the data. It is not difficult to see that preprocessing and transformation can be regarded as applying appropriate view operators to the initial views.

Example 2 View operators can be n-ary functions: e.g. for two views v_1, v_2 returning a Boolean values, we can create a conjunction view (representing the logical "and" of two attributes) using a 2-ary operator ϕ:

$$\phi[v_1, v_2](e) \equiv \phi(v_1(e), v_2(e)) = v_1(e) \wedge v_2(e). \qquad (2)$$

□

Example 3 (neighborhood operator) A neighborhood operator can be defined to generate views which are in the *neighborhood* (according to some specified definition) of the original viewscope: e.g. locally modifying the parameters in a parametric view. This type of operator may be used, for example, to conduct local optimization of views. □

The above examples closely resemble techniques appearing in the context of constructive induction, or feature construction (Bloedorn & Michalski 1998).

In the data mining stage of the KDD process, rules or hypotheses are generated by various learning algorithms. These generated rules may also be regarded as views. For example, a decision tree can be regarded as a function which returns for each entity e, a corresponding value at the leaf of where e ends up getting classified.

Definition 6 (learning based view operator) A *learning based view operator* $\mathcal{H}_L$ can be induced from a learning algorithm L, which uses the entity set and available views to create a new view. □

Example 4 (hypothesis generation (supervised)) A supervised learning process can be written as an operation on both the entity set and viewscope: $\mathcal{H}[E, V, v_c] = V'$, where $\mathcal{H}$ is a learning based view operator induced from some kind of induction algorithm (for example, C4.5 for the

Table 1: Summary of representing the KDD process with view(scopes).

Elements of the KDD Process	Description in terms of view(scope)
1) data selection	classification and filtering of entities according to a certain view, which decides whether the entity is used or not.
2) data preprocessing	Preprocessing of data can be expressed as a function over data, so naturally may be defined by a view(scope).
3) data transformation	Transformation can also be expressed as a function over data, so naturally may be defined by a view(scope).
4) data mining	Data mining can be expressed as a generation of new view(scope). Hypothesis generation algorithms can be considered as view operators.
5) interpretation/evaluation	interpretation/evaluation of a view(scope)
6) knowledge consolidation	recursively using newly generated view(scopes)

case of decision trees), E is the entity set, V is the set of views (or attributes/features) available to the algorithm, and v_c is the "answer" view. V' is the set of generated views (consisting of a single view, or perhaps multiple views: e.g. the top scoring views). The resulting view(s) $v' \in V'$ is expected to satisfy the property $v'(e) \simeq v_c(e)$ for most $e \in E$. □

Example 5 (hypothesis generation (unsupervised)) An unsupervised learning process can be written as an operation on both the entity set and viewscope: $\mathcal{H}[E, V] = V'$, this time not requiring v_c as in the supervised case.

For example, for entities E, viewscope V of numerical values, and clustering algorithm CL, $\mathcal{H}_{CL}$ will create a viewscope representing the clustering of the entities: $\mathcal{H}_{\mathcal{CL}}[E, V] = C$. Where C is a set of newly generated views (again consisting of a single view, or perhaps multiple views). A view $c \in C$ would return a cluster identifier $c(e)$ (telling which cluster the entity is clustered to) for each $e \in E$, and we would expect the distances (defined somewhere in relation to values from V) between the entities in the same cluster are small, and those in different clusters are large. □

Since hypotheses are equivalent to views, the evaluation/interpretation of a mined hypothesis is, in another words, the evaluation/interpretation of the mined view, meaning the manual evaluation of the view by a domain expert, or an automated evaluation according to some *score* (e.g. accuracy).

Since we observed that hypothesis generation algorithms generate views, the newly generated views may be used afterwards, perhaps in the next discovery task, or in refining the current task. This represents the *consolidation of the knowledge* gained from the data mining step.

The correspondence between views and the KDD process is summarized in Table 1.

Because these elements are captured abstractly as views and view operations, exploiting them can be done in a seamless, uniform manner. For example, since some view operators may operate on any view with a certain range set, we see that we can *reuse* these operators. Some hypothesis generation operators also have this property, and can be used for a variety of views. The same goes for preprocessing and transformations algorithms on data, which can be used for different entities (datasets).

The trial-and-error interaction between the domain expert and the problem, as is done in (Cheeseman & Stutz 1996) can be regarded as the trial-and-error of view design: the expert searches for good views and good view design, testing the views by applying them to the data, interpreting the outcome, making local modifications, trying completely different views and view designs, After such extensive investigations, the expert may, perhaps, finally understand the data he/she is facing with, and consider a view generated in the process as worthy knowledge.

These properties have lead us to develop an object oriented software library named HypothesisCreator($\mathcal{HC}$), realizing a competent set of views and view operators, trying to boost this interaction.

Characterization of N-Terminal Protein Sorting Signals

To illustrate the concepts described in the previous sections, we give a brief account of a successful knowledge discovery endeavor which we actually experienced working with biological data. The following case is presented as an example for our ideas, and a more detailed discussion of the experiments and the results will be given in a biological journal.

Background Proteins are composed of amino acids, and may be regarded as string sequences consisting of an alphabet of 20 characters. Most proteins are synthesized in the cytosol, and are carried to specific locations inside the cell. In most cases, the information determining the sub-cellular localization site is represented as a short amino acid sequence segment called a protein sorting signal (Nakai 2000). If we can somehow detect the amino acid sequence encoding this information, we would be able to predict the localization sites. Since cellular functions are often localized in specific compartments, this prediction of localization sites of various proteins is an important and challenging problem in the field of molecular biology, and would allow us to obtain indications of the functions for unknown or unannotated proteins. Further, if the rules for prediction were biologically interpretable, this knowledge could help in designing artificial proteins with desired properties. We consider here the typical N-terminal signals (signals that are known to appear somewhere near the "beginning" of the protein), which are mitochondrial targeting peptides (mTP), chloroplast transit peptides (cTP), and signal peptides (SP).

Mitochondrial targeting peptides are known to be rich in arginine (R), alanine (A), and serine (S), while negatively charged amino acid residues (aspartic acid (D) and glutamic acid (E)) are rare (von Heijne, Steppuhn, & Herrmann 1989). Only weak consensus sequences have been found.

Table 2: View operators used in our experiments.

Operator	Type of Operator	Description
$\mathcal{S}_{i,j}$	string → string	Substring: return a specific substring $[i,j]$ of a given string.
$\mathcal{I}_I$	string → string	Alphabet Indexing: return an indexed string, using an alphabet indexing I.
$\mathcal{P}_{p,A}$	string → bool	Pattern Match: return true if pattern p matches the string using pattern matching algorithm A, and false otherwise.
$\mathcal{D}_h$	string → vector⟨float⟩	AAindex: a homomorphism of a mapping h: char→ float (h corresponds to an entry in the AAindex Database).
$\mathcal{A}$	vector⟨float⟩ → float	Sum: returns the sum of the values of each element in the vector. (this value is referred to as the *indexing sum* in the text)
$\mathcal{T}_{s,t,b}$	double → bool	Threshold: return $b \in \{\mathrm{true}, \mathrm{false}\}$ if the input value v is within a certain threshold. ($s \leq v \leq t$)
$\mathcal{B}_o$	bool×bool → bool	Boolean Operation: $o \in \{\mathrm{and}, \mathrm{or}\}$.
$\mathcal{N}$	bool → bool	Boolean Not Operation: Negation of the input.

Further, they are believed to form an amphiphilic α-helix important for import into the mitochondrion. Chloroplast transit peptides are known to be rare in acidic residues, and also believed to form an amphiphilic α-helix (Bruce 2000). It has been established that a concrete consensus sequence does not occur in signal peptides. Rather, a three-region structure is conserved: a positively charged n-region, a hydrophobic h-region, and a polar c-region (von Heijne 1990).

TargetP (Emanuelsson *et al.* 2000) is the best predictor so far in the literature in terms of prediction accuracy, but since it is a neural network based system, it is difficult to understand the underlying rules for its prediction. PSORT (Nakai & Kanehisa 1992) and MitoProt (Claros & Vincens 1996) are systems which do utilize various expert knowledge about sorting signals, but they are somewhat obsolete and their prediction accuracy is unsatisfactory. The aim of our work was to discover simple and understandable rules which still had a practical prediction accuracy.

Approach The data we used was obtained from the TargetP web-site (http://www.cbs.dtu.dk/services/TargetP/). These data are divided into two data sets: plant and non-plant sequences. We describe our analysis on the plant data set of 940 sequences containing 368 mTP, 141 cTP, 269 SP, and 162 “Other” (consisting of 54 nuclear and 108 cytosolic proteins) sequences.

We designed views from two standpoints. One aimed to capture “global” characteristics of the N-terminal sequences. Since existing knowledge about the signals seemed to depend on biochemical properties of the amino acids contained, we decided to use the AAindex database (Kawashima & Kanehisa 2000), which is a compilation of 434 *amino acid indices*, where each amino acid index is a mapping from one amino acid to a numerical value, representing various physiochemical and biochemical properties of the amino acids.

The other view aimed to capture “local” characteristics: although strong consensus patterns do not seem to be present in the signals, there does seem to be a common structure to each of the signals. Therefore an *alphabet indexing* + approximate pattern view was designed. An alphabet indexing (Shimozono 1999) can be considered as a discrete, unordered version of amino acid indices, mapping amino acids to a smaller alphabet (in our case, $\{1,2,3\}$). After transforming the original amino acid sequence into a sequence of the alphabet indices, a pattern is sought.

Starting with the proteins as entities E and an initial view v which returns the amino acid sequence of each protein in E, the two types of views (which return Boolean values) can be defined as follows:

$$V_1 \equiv \mathcal{P}_{p,A}[\mathcal{I}_I[\mathcal{S}_{i,j}[v]]] \tag{3}$$

$$V_2 \equiv \mathcal{T}_{s,t,b}[\mathcal{A}[\mathcal{D}_h[\mathcal{S}_{i,j}[v]]]] \tag{4}$$

See Table 2 for the definitions of the view operators. Note that each operator (except $\mathcal{A}$) is parametric, and the range of the parameters defines the space of views to be searched.

The parameter space was designed as follows (taking into account the existing knowledge about the sorting signals): $[i,j]$: the 72 intervals $[5n+1, 5k]$ (where $n = 0 \ldots 8$ and $k = 1 \ldots 8$), p: all patterns of length $3 \sim 10$, A: approximate matching (Wu & Manber 1992) with up to $1 \sim 3$ mismatches, h: all entries of the AAindex database, b, s, t: all possible combinations (appearing in the data). For the alphabet indexing I, we first start with a random alphabet indexing, and a local optimization strategy (Shimozono *et al.* 1994) using a neighborhood operator was adopted.

After extensively searching for *good* views which discriminate each of the signal types from sets of other signals, we combined the discovered views into a *decision list* (See Figure 1) whose nodes consist of conjunctions of views from V_1 and V_2 (except for distinguishing SP, where only 1 view from V_2 was used).

Results The obtained parameters are summarized in Table 3. Concerning views of the form V_1, important features concerning sorting signals were discovered. For SP, the hydropathy index (Kyte & Doolittle 1982) was found to be the most effective. This knowledge is not entirely new, but it was surprising that such a simple rule could explain the signals so well - almost as good as TargetP. For mTP and cTP, the isoelectric point index (Zimmerman, Eliezer, & Simha 1968) was found to be effective. Since this index can be regarded as a more accurate version of the net amino acid charge, we can see that although mTP and cTP lack in negatively charged amino acids, mTP tend to be more positively

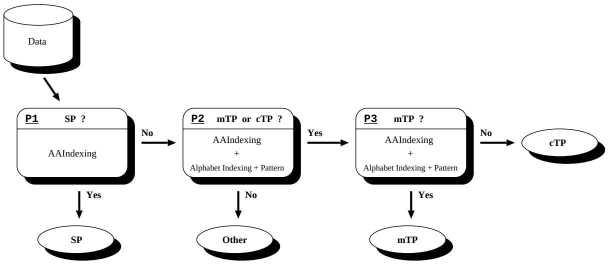

Figure 1: The decision list for predicting localization sites. The summary of parameters is given in Table 3.

Table 3: Summary of the Parameters used for the Final Hypotheses

Node	Substring	Amino acid index	Alphabet Indexing	Pattern	Mismatch allowance
P1	$[6, 25]$	Hydropathy Index	not used	not used	not used
P2	$[1, 30]$	Negative Charge	AI1	112111221	2 ins/del
P3	$[1, 15]$	Isoelectric Point	AI2	211211221	3 ins/del

Name	Alphabet Indexing		
	1	2	3
AI1	ACFLMPQSTVWY	IR	DEHKN
AI2	ACDEFGHLMNQSTVWY	KR	IP

charged. The values of the views after applying the sum operator $\mathcal{A}$ is plotted in Figure 2.

For views of V_2, the alphabet indexing and pattern were again found to capture existing knowledge about the patterns. For example, the patterns capture a periodicity of arginine (R) and/or lysine (K) which are characteristics of the amphiphilic α-helix structure of mTP and cTP.

Cross validation scores of the Matthews correlation coefficient (MCC) (Matthews 1975),

$$\frac{tp \times tn - fp \times fn}{\sqrt{(tp + fn)(tp + fp)(tn + fp)(tn + fn)}}$$

are summarized in Table 4. We can see that our predictor competes very well with TargetP.

Conclusion

We have presented an approach to describe the process of knowledge discovery in terms of *views*, and have seen that views are fundamental concepts in this process. The concept of views allows us to model various (if not all) steps of the knowledge discovery process, as well as provide an interface for human intervention in the knowledge discovery process. An example of a successful real world discovery task, in which the concepts discussed were observed to play an important role, was also presented.

The source code for the $\mathcal{HC}$ library implementing our ideas is available from its web-site (http://www.HypothesisCreator.net/).

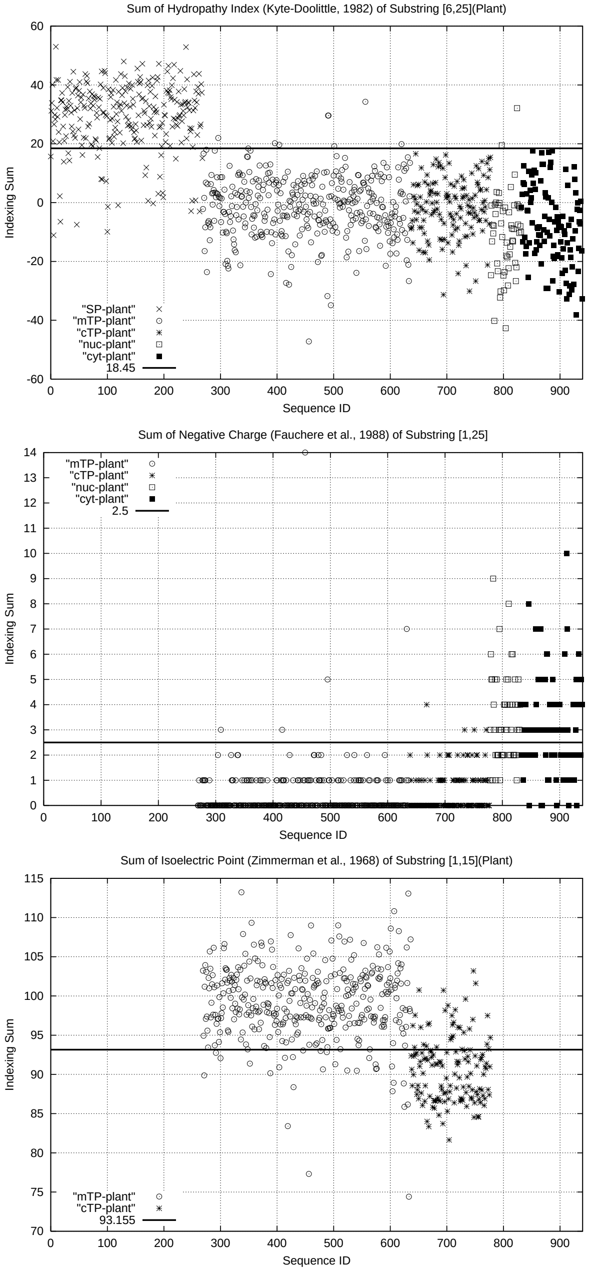

Figure 2: Indexing sum values for the best substring, amino acid index, and thresholds for distinguishing SP (top), (mTP+cTP) vs Other (middle), and mTP vs cTP (bottom). These represent a simple rule, for example for SP: calculate the indexing sum, with respect to the hydropathy index, for the 20 amino acids at position 6 through 25, and then check the sum against the threshold. (Note that the x-axis only represents the ID of the sequences, (in arbitrary order: SP, mTP, cTP, nuc, cyt from left to right) and does not give any information about the characteristics of the sequences)

Table 4: The Prediction Accuracy (5-fold cross validation) of the Final Hypothesis (scores of TargetP (Emanuelsson *et al.* 2000) in parentheses)

		Predicted category					
True category	# of seqs	cTP	mTP	SP	Other	Sensitivity	MCC
cTP	141	112 (120)	15 (14)	0 (2)	14 (5)	0.79 (0.85)	0.64 (0.72)
mTP	368	41 (41)	304 (300)	9 (9)	14 (18)	0.83 (0.82)	0.79 (0.77)
SP	269	16 (2)	8 (7)	237 (245)	8 (15)	0.88 (0.91)	0.89 (0.90)
Other	162	13 (10)	6 (13)	2 (2)	141 (137)	0.87 (0.85)	0.80 (0.77)
Specificity		0.62 (0.69)	0.91 (0.90)	0.96 (0.96)	0.80 (0.78)		

References

Bloedorn, E., and Michalski, R. S. 1998. Data-driven constructive indunction. *IEEE Intelligent Systems* 30–37.

Bruce, B. D. 2000. Chloroplast transit peptides: structure, function and evolution. *Trends in Cell Biology* 10:440–447.

Cheeseman, P., and Stutz, J. 1996. Bayesian classification (AutoClass): Theory and results. In *Advances in Knowledge Discovery and Data Mining*. AAAI Press/MIT Press.

Claros, M. G., and Vincens, P. 1996. Computational method to predict mitochondrially imported proteins and their targeting sequences. *European Journal of Biochemistry* 241(3):779–786.

Emanuelsson, O.; Nielsen, H.; Brunak, S.; and von Heijne, G. 2000. Predicting subcellular localization of proteins based on their N-terminal amino acid sequence. *Journal of Molecular Biology* 300(4):1005–1016.

Fayyad, U.; Piatetsky-Shapiro, G.; and Smyth, P. 1996. From data mining to knowledge discovery in databases. *AI Magazine* 17(3):37–54.

Kawashima, S., and Kanehisa, M. 2000. AAindex: Amino Acid index database. *Nucleic Acids Research* 28(1):374.

Khabaza, T., and Shearer, C. 1995. Data mining with Clementine. IEE Colloquium on 'Knowledge Discovery in Databaes'. *IEE Digest* No. 1995/021(B), London.

Kyte, J., and Doolittle, R. 1982. A simple method for displaying the hydropathic character of a protein. *Journal of Molecular Biology* 157:105–132.

Langley, P. 1998. The computer-aided discovery of scientific knowledge. In *Lecture Notes in Artificial Intelligence*, volume 1532, 25–39.

Maruyama, O.; Uchida, T.; Shoudai, T.; and Miyano, S. 1998. Toward genomic hypothesis creator: View designer for discovery. In *Discovery Science*, volume 1532 of *Lecture Notes in Artificial Intelligence*, 105–116.

Maruyama, O.; Uchida, T.; Sim, K. L.; and Miyano, S. 1999. Designing views in HypothesisCreator: System for assisting in discovery. In *Discovery Science*, volume 1721 of *Lecture Notes in Artificial Intelligence*, 115–127.

Matheus, C. J., and Rendell, L. A. 1989. Constructive induction on decision trees. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 645–650.

Matthews, B. W. 1975. Comparison of predicted and observed secondary structure of t4 phage lysozyme. *Biochim. Biophys. Acta* 405:442–451.

Michalski, R. S. 1983. A theory and methodology of inductive learning. *Artificial Intelligence* 20(2):111–161.

Motoda, H. 1999. Fascinated by explicit understanding. *J. Japanese Society for Artificial Intelligence* 14:615–625.

Nakai, K., and Kanehisa, M. 1992. A knowledge base for predicting protein localization sites in eukaryotic cells. *Genomics* 14:897–911.

Nakai, K. 2000. Protein sorting signals and prediction of subcellular localization. In *Advances in Protein Chemistry*, volume 54. Academic Press. 277–344.

Shimozono, S.; Shinohara, A.; Shinohara, T.; Miyano, S.; Kuhara, S.; and Arikawa, S. 1994. Knowledge acquisition from amino acid sequences by machine learning system BONSAI. *Transactions of Information Processing Society of Japan* 35(10):2009–2017.

Shimozono, S. 1999. Alphabet indexing for approximating features of symbols. *Theoretical Computer Science* 210:245–260.

von Heijne, G.; Steppuhn, J.; and Herrmann, R. G. 1989. Domain structure of mitochondrial and chloroplast targeting peptides. *European Journal of Biochemistry* 180:535–545.

von Heijne, G. 1990. The signal peptide. *J. Membr. Biol.* 115:195–201.

Wu, S., and Manber, U. 1992. Fast text searching allowing errors. *Commun. ACM* 35:83–91.

Zimmerman, J.; Eliezer, N.; and Simha, R. 1968. The characterization of amino acid sequences in proteins by statistical methods. *J. Theor. Biol.* 21:170–201.