The melting pot of automated discovery: principles for a new science

Jan M. Żytkow

Computer Science Department, UNC Charlotte, Charlotte, N.C. 28223
and Institute of Computer Science, Polish Academy of Sciences
zytkow@uncc.edu

Abstract. After two decades of research on automated discovery, many principles are shaping up as a foundation of discovery science. In this paper we view discovery science as automation of discovery by systems who autonomously discover knowledge and a theory for such systems. We start by clarifying the notion of discovery by automated agent. Then we present a number of principles and discuss the ways in which different principles can be used together. Further augmented, a set of principles shall become a theory of discovery which can explain discovery systems and guide their construction. We make links between the principles of automated discovery and disciplines which have close relations with discovery science, such as natural sciences, logic, philosophy of science and theory of knowledge, artificial intelligence, statistics, and machine learning.

1 What is a discovery

A person who is first to propose and justify a new piece of knowledge K is considered the discoverer of K. Being the first means acting autonomously, without reliance on external authority, because there was none at the time when the discovery has been made, or the discovery contradicted the accepted beliefs.

Machine discoverers are a new class of agents who should be eventually held to the same standards. Novelty is important, but a weaker criterion of novelty is useful in system construction:

Agent A discovered knowledge K iff A acquired K without the use of any knowledge source that knows K.

This definition calls for cognitive autonomy of agent A. It requires only that K is novel to the agent, but does not have to be made for the first time in the human history. The emphasis on autonomy is proper in machine discovery. Even though agent A discovered a piece of knowledge K which has been known to others, we can still consider that A discovered K, if A did not know K before making the discovery and was not guided towards K by any external authority. It is relatively easy to trace the external guidance received by a machine discoverer. All details of software are available for inspection, so that both the initial knowledge and the discovery method can be analyzed.

The existing systems would not reach success in making discoveries if we humans did not provide help. But even a limited autonomy is sufficient, or else we would hold machine discoverers to higher standards than humans. Consider historical details of any human discovery in order to realize the amount of method, knowledge and data which has been provided by others. Even the most revolutionary discovery made by an individual is a small incremental step prepared by prior generations.

2 Who are automated discoverers

One of the main research directions in machine discovery has been the automation of discovery in science. Many of the recent results can be found in collections edited by Shrager & Langley (1990), Edwards (1993), Zytkow (1992, 1993), Simon, Valdes-Perez & Sleeman (1997), and in Proceedings of 1995 AAAI Spring Symposium on Systematic Methods of Scientific Discovery, 1998 ECAI Workshop on Scientific Discovery, and AISB'99 Symposium on Scientific Creativity. Risking a slight oversimplification, research on scientific discovery can be split into discovery of empirical laws and discovery of hidden structure. We will use many systems as examples, but the size limits preclude a systematic outline of discovery systems and their capabilities.

2.1 Knowledge discovery in databases

Automation of scientific discovery can be contrasted with knowledge discovery in databases (KDD) which is a fast-growing field, driven by practical business applications. KDD is focused on data which are fixed and collected for purposes alien to discovery. Both the search techniques and the results of knowledge discovery in databases are far more simplistic than those of automated scientific discovery. Numerous publications describe KDD research, for instance collections of papers, typically conference proceedings, edited by Piatetsky-Shapiro & Frawley (1991), Piatetsky-Shapiro (1993), Ziarko (1994), Komorowski & Zytkow (1997), Zytkow & Quafafou (1998), Chaudhuri & Madigan (1999) and many others.

2.2 A rich brew is melting in the pot

Knowledge discovery tools have been inspired by many existing research areas: artificial intelligence, philosophy of science, history of science, statistics, logic and good knowledge of natural sciences as they provide the clearest standards of discovery. Each of these areas has been occupied with another aspect of discovery. Machine discovery uses the creative combination of knowledge and techniques from the contributing areas, but also adds its own extra value, which we try to summarize in several principles.

3 Principles of autonomy

Consider two agents A_1 and A_2, who are exactly the same, except that A_2 possesses a few extra means useful in discovery, that apply, for instance, in data acquisition or in inductive generalization. A_2 is more autonomous than A_1, as those extra means increase A_2's discovery capabilities. They are unavailable to A_1, unless provided by other agents.

This simple analysis would be a poor philosophy when applied to humans, because our mental techniques are so difficult to separate, but it makes good sense when agents are computer systems that can be easily cloned and compared. We can draw our first principle:

$\mathcal{A}1$: Autonomy of an agent is increased by each new method that overcomes some of the agent's limitations.

Admittedly, each machine discoverer is only autonomous to a degree, but its autonomy can increase by future research. This principle leads to a program: identify missing functions of discovery systems and develop methods that supply those functions.

The mere accumulation of new components, however, is not very effective. Each new component may have been designed to provide a particular missing function, but after it is completed, it may be used in new creative ways, in combination with the existing methods. Such combinations multiply the overall discovery capability. As a result of integration, more discovery steps in succession can be performed without external help, leading to greater autonomy:

$\mathcal{A}2$: Autonomy of an agent is increased by method integration, when new combinations of methods are introduced.

The BACON program (Langley, Simon, Bradshaw & Zytkow, 1987) was the first to follow these two principles. It started from BACON.1 which is a heuristic equation finder, followed by BACON.3 which generates data and applies recursively BACON.1, then by BACON.4 which converts nominal variables to numerical, and uses both BACON.3 and BACON.1 to generate numerical laws. Finally, BACON.5 augments earlier BACONs with the reasoning based on symmetry and conservation. Similar programs led to the increased discovery capabilities of FAHRENHEIT (Zytkow, 1996), IDS (Nordhausen & Langley, 1993), BR (Kocabas, 1991; Kocabas & Langley, 1995), and MECHEM (Valdes-Perez, 1993, 1994).

Many methods use data to generate knowledge. When applied in sequence, elements of knowledge generated at the previous step become data for the next step. This perspective on knowledge as data for the next step towards an improved knowledge is important for integration of many methods:

$\mathcal{A}3$: Each piece of discovered knowledge can be used as data for another step towards discovery:

$$\textbf{Data-1} \xrightarrow{\textbf{Step-1}} \textbf{Knowledge-1} = \textbf{Data-2} \xrightarrow{\textbf{Step-2}} \textbf{Knowledge-2} = \textbf{Data-3}$$

A single step rarely permits a decisive evaluation of results. A combination of steps provides a more informed evaluation, that is extra reasons for acceptance of an alternative generated at step 1. For instance, several equations of comparable simplicity can often fit the same data within acceptable accuracy. Further steps can help in making choices among the competing equations. Some equations may provide a better generalization to a new variable or lead to a broader scope in result of boundary search (FAHRENHEIT; Zytkow, 1996):

$\mathcal{A}$4: Autonomous choice is improved by evaluation that is back-propagated to results reached at earlier steps.

Principles A1-A4 guide some of scientific discovery systems, but are still a remote ideal in KDD, dominated by simple algorithms and human guidance.

4 Theory of knowledge

Knowledge of external world goes beyond data, even if data are the primary source of knowledge. It is important to see beyond formal patterns and understand elements of the formalism in relation to elements of the external world.

Consider a fairly broad representation of a regularity (law, generalization):

Pattern (relationship) *P* holds in the range *R* of situations.

In practical applications this schema can be narrowed down in many ways, for instance:

(1) **if** $P_1(A_1)\&...\&P_k(A_k)$ **then** $Rel(A, B)$

where $A, B, A_1, ..., A_k$ are attributes that describe each in a class of objects, while $P_1, ..., P_k$ are predicates, such as $A_1 > 0$ or $A_2 = a$. An even simpler schema:

(2) **if** $P_1(A_1)\&...\&P_k(A_k)$ **then** $C = c$

covers all rules sought as concept definitions in machine learning.

A good fit to data is important, but discoverer should know objects described in data and understand predicates and constants that occur in a generalization. Only then can knowledge be applied to other similar situations.

$\mathcal{K}$1: Seek objective knowledge about the real world, not knowledge about data.

This principle contrasts with a common KDD practice, when researchers focus entirely on data. Sometimes specific knowledge about data is important: which data are wrong, what data encoding schemas were used.

An example of disregard for objective knowledge is a new popular mechanisms for concept learning from examples, called bagging. Consider a system that produces decision trees. When provided with different samples of data it will typically generate different trees. Rather than analyzing commonalities in those trees in order to capture objective relations in data, all the trees are used jointly to predict values of the class attribute. If you wait a few minutes you will get another dozen of trees. Many parameter values in such trees are incidental, but the method has no concern for objective knowledge.

Schemas such as (1) or (2) define vast, sometimes infinite, hypothesis spaces. Hypotheses are generated, often piece by piece, evaluated and retained or eliminated.

$\mathcal{K}2$: [Principle of knowledge construction] All elements of each piece of knowledge are constructed and evaluated by a discovery system.

Schemas such as (2) define hypothesis spaces used by many systems. According to $\mathcal{K}2$ the same construction steps must be made by each system, when they operate in the same hypothesis space. If the construction mechanisms were made clear, it would be easy to compare them.

Likewise, different systems use the same or very similar evaluation measures:

– accuracy: how close are predictions to actual data;
– support: what percentage of cases is covered by RANGE and by PATTERN.

Other measures are concerned with probabilistic component of knowledge:

– predictive strength: degree of determinism of predictions;
– significance: how probable is that data could have been generated from a given statistical distribution.

Predictions are essential for hypothesis evaluation. It is doubtful that we would consider a particular statement a piece of knowledge about external world if it would not enable empirically verifiable predictions.

$\mathcal{K}3$: A common characteristic of knowledge is its empirical contents, that is such conclusions which are empirically verifiable predictions.

Knowledge improvement can be measured by the increased empirical contents. Logical inference is used in order to draw empirically verifiable conclusions. The premises are typically general statements of laws and some known facts, while conclusions are statements which predict new facts.

When we examine carefully the results of clustering, the way they are typically expressed, we do not see empirical contents. Exhaustive partitioning of data space leads no room for empirical contents.

Empirical contents can occurs in regularities (laws, statements, sentences), not in predicates which may or may not be satisfied. Concepts, understood as predicates, have no empirical contents.

$\mathcal{K}4$: Each concept is an investment; it can be justified by regularities it allows to express.

We can define huge numbers of concepts, but such activity does not provide knowledge. The whole universe of knowledge goes beyond concept definitions.

Many knowledge pieces can be expressed in a simple form of classification rules, association rules, contingency tables, equations, neural networks, logical statements and decision trees. But when the process of discovery continues, producing large numbers of such pieces, their management becomes a major problem. Some discovery systems organize large numbers of simple pieces into a global graph representation. Links in a graph are used to represent relationships between pieces of knowledge, while frame-like structures represent knowledge contained in individual nodes in the graphs.

$\mathcal{K}$5: Knowledge integration into graphs provides fast, directed access to individual pieces of knowledge.

System such as DIDO (Scott and Markovitch, 1993) FAHRENHEIT, IDS, LIVE (Shen, 1993) use tools for constructing, maintaining, and analyzing the network of knowledge emerging in the discovery process. FAHRENHEIT's knowledge graph (Żytkow, 1996) allows the system to examine any given state of knowledge and seek new goals that represent limitations of knowledge in the graph.

For instance, when an equation E has been found for a sequence of data, new alternative goals are to find the limits of E's application or to generalize E to another control variable. Generalization, in turn, can be done by recursively invoking the goals of data collection and equation fitting (BACON.3: Langley et.al. 1987; and FAHRENHEIT).

5 Principles of search

Discoverers explore the unknown. They examine many possibilities which can be seen as dead ends from the perspective of the eventually accepted solutions, because they do not become components of the accepted solutions. This process is called search. We can conclude that:

$\mathcal{S}$1: If you do not search, you do not discover.

5.1 Search spaces

The search space, also called problem space or state space, has been introduced in artificial intelligence as a conceptual tool to enable a theoretical treatment of the search process (Simon, 1979).

A search space is defined by a set of states S and a two argument relation $M \subseteq S \times S$, called a move relation. M contains all direct state to state transitions. The aims of search are represented by the evaluation function $E : S \rightarrow \mathcal{R} \times \ldots \times \mathcal{R}$.

In practice, the states are not given in advance, because search spaces are very large, often infinite. States are constructed by search operators from the existing states. Operators are algorithms which implement the relation M, by acts of construction of new states from the existing states, that is from states which have been earlier constructed.

In applications to discovery, states have the meaning of tentative knowledge pieces or hypotheses. They are implemented as instances of datastructures that represent different possible pieces of knowledge. The move relation represents incremental construction of knowledge elements. The evaluation function provides several metrics that apply to hypotheses. In summary:

$\mathcal{S}2$: **Make the search design simple and explicit. Define datastructures that represent tentative pieces of knowledge and operations on those pieces. Define metrics on pieces of knowledge**

Search design must be flexible enough so that it can be adjusted to a variety of problems, data and computational resources. Yet it must be simple enough so that search properties are understood, problems fixed and the scope of search modified when needed.

5.2 Discovery as problem solving search

A simple search problem can be defined by a set of initial states and a set of goal states. The task is to find a trajectory from an initial state to a goal state. In the domain of discovery the goal states are not known in advance. They are typically defined by threshold values of tests such as accuracy and statistical significance. Goal states exceed those thresholds. Without reaching the threshold, even the best state reached in the discovery process can be insufficient.

$\mathcal{S}3$: **[Herbert Simon 1] Discovery is problem solving. Each problem is defined by the initial state of knowledge, including data and by the goals. Solutions are generated by search mechanisms aimed at the goals**.

The initial state can be a set of data, while a goal state may be an equation that fits those data (BACON.1 and other equation finders). The search proceeds by construction of terms, by their combinations into equations, by generation of numerical parameters in equations and by evaluation of completed equations.

While new goals can be defined by pieces of knowledge missing in a knowledge graph, plans to accomplish those goals are different search mechanisms. The same goal can be carried by various plans. For instance, a variety of equation finders represent different plans at reaching the same or a very similar goal: BACON.1, COPER (Kokar, 1986), FAHRENHEIT, IDS (Nordhausen and Langley, 1993), KEPLER (Wu and Wang, 1989), Dzeroski and Todorovski (1993).

Goals and plans can be called recursively, until plans are reached which can be carried out directly, without reference to other goals and plans. Some equation finding systems use complex goals decomposed into similar subgoals, and repeatedly apply the same search mechanisms to different problems. They design many experiments, collect sequences of data and search for equations that fit those data (BACON.3, FAHRENHEIT, SDS Washio & Motoda, 1997).

Search spaces should be sufficiently large, to provide solutions for many problems. But simply enlarging the search space does not make an agent more creative. It is easy to implement a program that enumerates all strings of characters. If enough time was available, it would produce all books, all data structures, all computer programs. But it produces a negligible proportion of valuable results and it cannot tell which are those valuable results.

$\mathcal{S}4$: **[Herbert Simon 2] A heuristic and data-driven search is an efficient and effective discovery tool. Data are transformed into plausible**

pieces of solutions. Partial solutions are evaluated and used to guide the search.

Still, the search may fail or take too much time and a discoverer should be able to change the goal and continue.

$\mathcal{S}5$: [Recovery from failure] Each discovery step may fail and cognitive autonomy requires methods that recognize failure and decide on the next goal

For instance, if an equation which would fit the data cannot be found, those data can be decomposed into smaller fragments and the equation finding goal can be set for each fragment separately. If no regularity can be found, eventually data can be treated as a lookup table.

5.3 Search control

Search states can be generated in different orders. Search control, which handles the search at run-time, is an important discovery tool. Some of the principles that guide the design of search control are very broad and well known:

- decompose the goal into independent subgoals;
- evaluate partial results as early as possible;
- search is directed by the evaluation function; do not expect that the results satisfy a non-applied metric;

We can propose several principles specific to discovery.

$\mathcal{S}6$: [Simple-first] Order hypotheses by simplicity layers; try simpler hypotheses before more complex.

The implementation is easy, since simpler hypotheses are constructed before more complex. Also, simpler hypotheses are usually more general, so they are tried before more complex, that is more specific hypotheses. Thus if a simple hypothesis is sufficient, there is no need to make it more complex.

$\mathcal{S}7$: [Dendral] Make search non-redundant and exhaustive within each simplicity layer.

Do not create the same hypothesis twice, but do not miss any. The implementation may not be easy, but the principle is important. If there are many alternative solutions of comparable simplicity, what are the reasons to claim that one of them is true. Each solution is questionable, because they make mutually inconsistent claims. Typically, alternative solutions indicate the need for more data or further evaluation.

In set-theoretic terms, S and M together form a directed graph. But practically, states are constructed from other states rather than retrieved from memory. Therefore a search tree rather than a graph represents the history of search. An isomorphic hypothesis can be generated at different branches of the search tree. If that happens, typically the same hypothesis is generated a very large number of times. Each time the hypothesis can be expanded to a potentially large search subtree, leading to massive redundant search. Testing for isomorphism is complex, so that it may not be useful. Isomorph-free graph generation is preferred, when it can be arranged (Dendral, GELLMANN, Zytkow 1992).

6 Beyond simple-minded tools

While equations are often an adequate generalization tool for scientific data, in KDD applications the situation is more complex. Knowledge in many forms can be derived from data and it is not known which is the best form of knowledge for a given data set. The vast majority of data mining is performed with the use of single-minded tools. Those tools miss discovery opportunities if results do not belong to a particular hypothesis space. Further, they rarely consider the question whether the best fit hypothesis is good enough to be accepted and whether other forms of knowledge are more suitable for a given case. To improve the dominant practice, we should use the following principle:

$\mathcal{O}$1: [Open-mindness] Knowledge should be discovered in the form that reflects the real-world relationships, not one or another tool at hand.

It may be unnecessarily complex, however, to search for many forms of knowledge and then retain those best justified by data. The methodology applied in 49er (Zembowicz & Zytkow, 1996) uses simple forms of knowledge to guide search for more complex and specialized forms:

1. Use a single method to start data mining. Search for contingency tables.
2. Detect different forms of regularities in contingency tables by specialized tests.
3. Search for specialized knowledge in separate hypothesis spaces.
4. Combine many regularities of one type into specialized theories. For instance, create taxonomies, inclusion graphs, systems of equations.

Different forms of knowledge require search in different hypothesis spaces. But if data do not fit any hypothesis in a given space, much time can be saved if that space is not searched at all. This problem can be solved in step 2 above, by demonstrating non-existence of solutions in particular spaces.

7 Statistics

While many methods of statistics are still unknown to the discovery community, many textbook solutions have been implemented. Statistics adds a probabilistic component to deterministic regularities handled by the languages of logic, algebra, and the like.

So called random variables represent probabilistic distributions of measurements, which are due to error and to variability within a population.

Equations and other forms of deterministic knowledge can be augmented with statistical distributions, for instance, $y = f(x) + N(0, \sigma(x))$. $N(0, \sigma(x))$ represents Gaussian distribution of error, with mean value equal zero and standard deviation $\sigma(x)$.

Statistics offers methods which estimate parameter values of a distribution (population) from a sample, analyze bias of estimators, derive sampling distributions, then significance tests and confidence intervals for estimated parameters.

Most often a particular distribution is assumed rather than derived from data, because traditional statistical data mining operated on small samples and used visualization tools to stimulate human judgement. Currently, when large datasets are abundant and more data can be easily generated in automated experiments, we can argue for verification of assumptions:

$\mathcal{STAT}$1: Do not make assumptions and do not leave unverified assumptions.

For instance, when using the model $y = f(x) + N(0, \sigma(x))$ verify Gaussian distribution of residua, with the use of runs test and other tests of normality. Publications in statistics notoriously start from "Let us assume that ..." Either use data to verify the assumptions, and when this is not possible, ask what is the risk or cost when the assumptions are not met.

Another area which requires revision of traditional statistical thinking is testing hypothesis significance. Statistics asks how many real regularities are we willing to disregard (error of omission) and how many spurious regularities are we willing to accept (error of admission). In a given dataset, weak regularities cannot be distinguished from patterns that come from random distribution. Consider search in random data: when 100,000 hypotheses are examined, 1000 of them should be accepted when the significance threshold is the standard 1%. To minimize the number of such pseudo-regularities, we should set a demanding threshold of acceptance. 0.01% = 0.0001 would pass 10 spurious regularities. However, by chosing a demanding acceptance threshold we risk ignoring regularities which are real but weak. The problem arises in automated discovery because they search massive hypothesis spaces with the use of statistical tests which occasionally mistake a random fluctuation for a genuine regularity.

$\mathcal{STAT}$2: [Significance 1] Chose a significance threshold that enables middle ground between spurious regularities and weak but real regularities specific to a given hypothesis space.

While a significance threshold should admit a small percent of spurious regularities, it is sometimes difficult to compute the right threshold for a given search. Statistical significance thresholds depend on the number of independent hypotheses and independent tests. When those numbers are difficult to estimate, experiments on random data can be helpful. We know that those data contain no regularities, so all detected regularities are spurious and should be rejected by the test of significance.

$\mathcal{STAT}$3: [Significance 2] Use random data to determine the right values of significance thresholds for a given search mechanism.

The significance dilemma for a given regularity can be solved by acquisition of additional data.

8 Conclusions

In this paper we focused on selected principles that guide research in machine discovery. They are related to areas such as artificial intelligence, logic and philosophy of science, and statistics, but they are specific to machine discovery. We discussed a few ways in which these principles interact and some of the ways in which they affect research on discovery, in particular discovery systems construction.

References

1995 Working Notes AAAI Spring Symposium on Systematic Methods of Scientific Discovery. Stanford, March 27-29.

1998 ECAI Workshop on Scientific Discovery. Brighton, August 24.

1999 AISB Symposium on Scientific Creativity. Edinburgh. April 6-9.

Chaudhuri, S. & Madigan, D. eds. 1999. *Proceedings of the Fifth ACM SIGKDD Intern. Conf. on Knowledge Discovery and Data Mining*, ACM, New York.

Dzeroski, S. & Todorovski, L. 1993. Discovering Dynamics, *Proc. of 10th International Conference on Machine Learning*, 97-103

Edwards, P. ed. 1993. *Working Notes MLNet Workshop on Machine Discovery.* Blanes, Spain.

Kocabas, S. 1991. Conflict Resolution as Discovery in Particle Physics. *Machine Learning, 6*, 277-309.

Kocabas, S. & Langley, P. 1995. Integration of research tasks for modeling discoveries in particle physics. *Working notes of the AAI Spring Symposium on Systematic Methods of Scientific Discovery*, Stanford, CA, AAAI Press. 87-92.

Kokar, M.M. 1986. Determining Arguments of Invariant Functional Descriptions, *Machine Learning, 1*, 403-422.

Komorowski, J. & Zytkow, J.M. 1997. *Principles of Data Mining and Knowledge Discovery*, Springer.

Kulkarni, D., & Simon, H.A. 1987. The Process of Scientific Discovery: The Strategy of Experimentation, *Cognitive Science, 12*, 139-175

Langley, P., Simon, H.A., Bradshaw, G., & Zytkow J.M. 1987. *Scientific Discovery; Computational Explorations of the Creative Processes.* Boston, MIT Press.

Nordhausen, B., & Langley, P. 1993. An Integrated Framework for Empirical Discovery. *Machine Learning* 12, 17-47.

Piatetsky-Shapiro, G. ed. 1993. *Proc. of AAAI-93 Workshop on Knowledge Discovery in Databases.*

Piatetsky-Shapiro, G. & Frawley, W. eds. 1991. *Knowledge Discovery in Databases*, Menlo Park, Calif.: AAAI Press.

Scott, P.D., Markovitch, S. 1993. Experience Selection and Problem Choice In An Exploratory Learning System. *Machine Learning, 12*, p.49-67.

Shen, W.M. 1993. Discovery as Autonomous Learning from Environment. *Machine Learning, 12*, p.143-165.

Shrager J., & Langley, P. eds. 1990. *Computational Models of Scientific Discovery and Theory Formation*, Morgan Kaufmann, San Mateo:CA

Simon, H.A. 1979. *Models of Thought.* New Haven, Connecticut: Yale Univ. Press.

Simon, H.A., Valdes-Perez, R. & Sleeman, D. eds. 1997. *Artificial Intelligence* 91, Special Issue: Scientific Discovery.

Valdés-Pérez, R.E. 1993. Conjecturing hidden entities via simplicity and conservation laws: machine discovery in chemistry, *Artificial Intelligence*, 65, 247-280.

Valdés-Pérez, R.E. 1994. Human/computer interactive elucidation of reaction mechanisms: application to catalyzed hydrogenolysis of ethane, *Catalysis Letters*, 28, p.79-87.

Washio, T., & Motoda, H. 1997, Discovering Admissible Models of Complex Systems Based on Scale-Types and Identity Constraints, *Proc. IJCAI'97*, 810-817.

Wu, Y. and Wang, S. 1989. Discovering Knowledge from Observational Data, In: Piatetsky-Shapiro, G. (ed.) *Knowledge Discovery in Databases, IJCAI-89 Workshop Proceedings*, Detroit, MI, 369-377.

Zembowicz, R. & Żytkow, J.M. 1991. Automated Discovery of Empirical Equations from Data. In Ras. Z. & Zemankova M. eds. *Methodologies for Intelligent Systems*, Springer-Verlag, 1991, 429-440.

Zembowicz, R. & Żytkow, J.M. 1996. From Contingency Tables to Various Forms of Knowledge in Databases, in Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. & Uthurusamy eds. *Advances in Knowledge Discovery & Data Mining*, AAAI Press. 329-349.

Ziarko, W. ed. 1994. *Rough Sets, Fuzzy Sets and Knowledge Discovery*, Workshops in Computing, Springer-Verlag.

Żytkow, J.M. & Quafafou, M. 1998. *Principles of Data Mining and Knowledge Discovery,* Springer.

Żytkow, J.M. ed. 1992. *Proceedings of the ML-92 Workshop on Machine Discovery (MD-92).* National Institute for Aviation Research, Wichita, KS.

Żytkow, J.M. ed. 1993 *Machine Learning, 12.*

Żytkow, J.M. 1996. Automated Discovery of Empirical Laws, *Fundamenta Informaticae, 27*, p.299-318.

Żytkow, J.M. ed. 1997 *Machine Discovery*, Kluwer.