Toward Applying Machine Learning to Design Rule Acquisition for Automated Graphics Generation

Michelle X. Zhou **Sheng Ma**

IBM T. J. Watson
30 Saw Mill River Rd. Rt. 9A
Hawthorne, NY 10532
+1 914 784 7000
{mzhou, shengma}@us.ibm.com

Abstract†

Effective automated graphics generation systems rely on well-established design rules. We are currently exploring how machine learning techniques may be used to acquire design rules automatically. In this paper, we present a model that addresses four fundamental aspects toward applying machine learning to graphics design rule formulation. In particular, learning spaces and learning goals describe where learning may take place and what needs to be learned. Learning features and learning strategies discuss how to describe various learning data and how to choose proper learning techniques. Using our model, we have designed and conducted two experiments to demonstrate two different applications of machine learning in graphics rule acquisition.

Introduction

Existing automated graphics generation systems use hand-crafted design rules to map data to be conveyed onto proper visual presentations (e.g., Mackinlay, 1986; Roth and Mattis, 1991; Zhou and Feiner, 1998a). But hand-crafting design rules can be laborious as experts may need to process large amounts of evidence before extracting rules. As the rule base grows, it is also difficult to integrate new rules with existing rules, and discover/rectify the inconsistency among the rules. Thus, we are exploring how to use machine learning to automatically acquire design rules.

Graphics design rules may be formed by classifying governing principles from a set of sample presentations, or by generalizing observed design patterns (inductive learning). New rules may be added to an existing rule base by following a specific user instruction (learning from instruction), or by transforming similar existing rules to cover a new situation (learning by analogy). Consequently, machine learning helps acquire various design rules automatically, and improves the overall quality of a rule base (e.g., simplifying expert-derived rules and removing redundancies). Hence better graphics generation systems can be constructed to make more effective design decisions.

Although machine learning has been widely applied to rule acquisition in other domains (e.g., speech synthesis in [Pan and McKeown, 1998]), its application to graphics rule formulation has hardly been addressed. In this paper, we introduce a general model that addresses how to systematically acquire graphics design rules using machine learning. In particular, our model focuses on four fundamental aspects toward applying machine learning to rule acquisition: learning spaces, learning goals, learning features, and learning tools.

We first establish three learning spaces to distinguish various learning problems related to rule formulation. To solve different learning problems, we describe how to formulate specific learning goals systematically. We present two feature spaces to characterize two basic types of information involved in our learning problems and discuss how to extract relevant features using specific learning goals. Based on the type of design rules to be acquired, we examine the applicability of various learning strategies. In addition to the general model, we also present two experiments to demonstrate the use of machine learning in graphics rule acquisition. Specifically, these two experiments help establish two types of rules using different learning techniques.

Following a brief introduction on related work, we describe where, what, and how machine learning may occur in rule acquisition. We then discuss our experimental settings and results. Finally, we present our conclusions and an agenda for future work.

Related Work

Two main types of information involved in rule formulation are a generation system's input (e.g., data to be conveyed) and output (graphics). To prepare these information for machine learning, we have adopted previous research results on input and output information characterization. On the input side, researchers establish data characterization taxonomies to abstract what and how presentation-related data properties influence visual encoding strategies (e.g., Roth and Mattis, 1990; Zhou and Feiner, 1998a). Moreover, researchers analyze presentation intents (e.g., Casner, 1991; Roth and Mattis, 1990) to address how different user information-seeking goals affect design decisions. To describe visual output systematically, on the other hand, researchers characterize different presentation formats [Lohse et al., 1994], formulate a set of graphics languages [Mackinlay, 1986], or define the formal syntax or semantics of a particular visual representation [Marks, 1991]. Nevertheless, we have extended these efforts to address both the input and output more comprehensively. For example, we can describe a wide range of visual presentations at multi-

† We would like to thank Shimei Pan (Columbia University) for all useful discussions and suggestions.

ple levels of abstraction.

Learning in Design Rule Acquisition

Automated graphics generation is a visual mapping process that automatically maps a set of inputs (e.g., data to be conveyed) to a visual output. To guide this mapping, design rules must specify various relations between the input and output. As the input and output may vary a great deal, we must model them using a set of common features so that general design rules can be extracted to address different situations. Thus, we establish three learning spaces to identify three basic learning problems related to graphics rule acquisition (Figure 1). To solve these learning problems, we illustrate how to identify and formulate specific learning goals in corresponding learning spaces. For various observational data, we also address how to form relevant features to describe them in specific learning goals. Finally we discuss how to acquire various graphics rules using different learning strategies.

Learning Spaces

Learning problems related to rule formulation can be partitioned into three learning spaces (Figure 1). Specifically, the *information learning space* emphasizes learning the properties of various inputs to a synthesis system. The *visual learning space* focuses on modeling the properties of different visual outputs. To establish design rules, the *rule learning space* stresses extracting the mappings between various input information and visual output.

Information Learning Space. To initiate a visual mapping, inputs of a generation system are matched against the conditions of design rules. Here our inputs include three types of information: presentation data, presentation intents, and presentation context. *Presentation data* is the data to be visually conveyed (e.g., a patient medical record). *Presentation intents* specify user's information-seeking goals (e.g., summarizing the patient's medical record). *Presentation context* describes the intended audience (e.g., students or teachers) and environment (e.g., the use of a high-resolution color computer screen).

To express a wide variety of presentation data, we use a

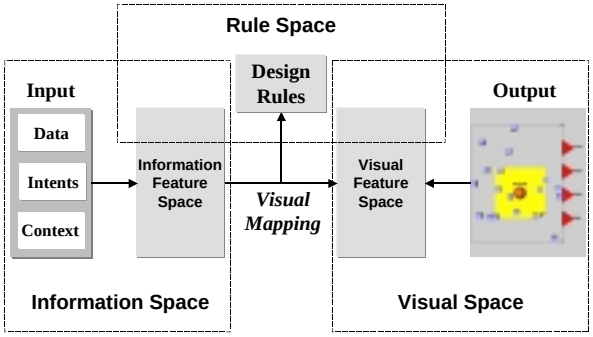

Figure 1. Three learning spaces in rule acquisition

set of presentation-related data properties captured in a data characterization taxonomy [Zhou and Feiner, 1998a]. Nevertheless, it is not always easy to specify these data properties precisely and efficiently by hand. This task becomes more difficult when data are voluminous and complex. Thus, we intend to use machine learning to automatically label data properties. For example, we know that the importance of data may affect visual encoding strategies. To determine data importance, we can use machine learning to establish importance clusters based on other data properties (e.g., data semantic domain).

With little knowledge about the data, in many applications, users may not know their *exact* information-seeking goals (presentation intents). In these cases, machine learning may be used to generalize the correlations between observed data properties and presentation intents/context. These correlations could help automatically formulate specific presentation intents or choose proper presentation context based on known data properties. For example, we may learn that there is a correlation between presentation intents, such as Summarize/Elaborate, and the quantity of the data to be conveyed. Using the data quantity information, a generation system can then suggest proper presentation intents for communicating the data.

Visual Learning Space. Since there are few models that assess various visual effects accurately, we may use learning techniques to model salient visual features and visual relations precisely. For example, visual similarity is often used to describe a prominent perceptual feature in a visual presentation. To model visual similarity between two visual objects v_i and v_j precisely, we may define a similarity measure using Euclidean distance:

$$(1).\ \text{Similarity}(v_i, v_j) = \sum_{k=1}^{K} w_k (f_i[k] - f_j[k])^2 .$$

Here each visual object is expressed using a set of lower-level features (e.g., color and shape). K is the total number of features; $f_i[k]$ is the kth feature of v_i and w_k is the weight of $f[k]$. The weight is used to measure how much a feature contributes to perceivable visual similarity. To model the relevance of various features, machine learning may be used. Suppose that we have a set of sample presentations that exhibit different degrees of visual similarity. We can then train a learning system on these samples to extract the relations between various lower-level features and the visual similarity effect.

Once we assess various visual effects accurately, we can develop specific design rules for achieving them. Using Formula (1), for example, the similarity score may be lower between two objects that share the same shape but in different colors (a red and a green sphere) than between two objects with different shape but in the same color (a red cube and a red sphere). As a result, a design rule may be formed to assert that similar shapes should be used to achieve a higher degree of visual similarity (lower score).

Rule Space Learning. Since automated graphics genera-

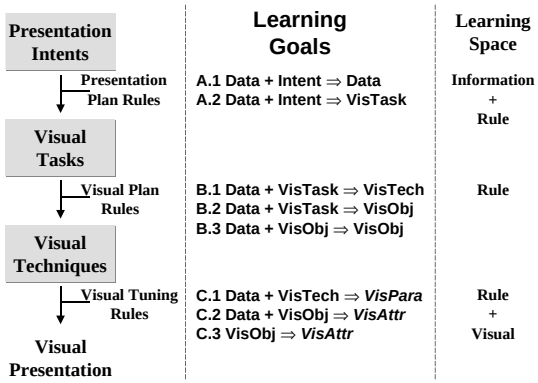

Figure 2. Design-process-driven learning goals

tion usually takes place at multiple levels [Seligmann and Feiner, 1991; Zhou and Feiner, 1998a], we need different design rules to handle different levels of design tasks. In general, learning in visual space helps acquiring design rules that address lower-level graphics synthesis tasks. For example, the similarity rule mentioned above describes how to achieve a specific visual effect (resemblance) using the combinations of visual primitives (e.g., shape). In contrast, learning in rule space focuses on extracting design principles that handle higher-level design tasks, such as how to encode a particular type of data. Thus, rule space learning is mainly concerned with extracting the mapping principles between a system's input and its output (Figure 1).

To induce mappings, we first need to obtain an accurate assessment of both the input and output. We then use different machine learning methods to extract the mappings. Moreover, we hope to assess the probability of the mapping occurrence. This probability is known as the confidence factor (CF). As a result, design rules can be specified using these mappings and be ranked by the confidence factors to help solve potential design conflicts. That is, a rule with a higher CF is preferred than a rule with a lower CF when the conditions of both rules are satisfied.

Learning Goals

To solve various learning problems, we formulate *learning goals* to specify the particular learning objectives to be achieved. As every design rule can be considered a mapping between a set of conditions and actions, a general learning objective in rule acquisition may be formulated as to discover the mappings between rule conditions and actions:

(2). $C_1 + ... + C_i ... + C_n \Rightarrow A_1, ..., A_j, ..., A_m$,

where C_i denotes the ith condition and A_j is the jth action; and we have total n conditions and m actions.

Process-Driven Learning Goals. Using Formula (2), we can formulate general learning goals. However, if the number of conditions or actions is large, as in graphics design, we need to identify and formulate very specific learning goals so that they can be achieved efficiently. To formulate learning goals systematically and comprehensively, we use a design-process-driven approach. This is because different synthesis processes may require different types of design rules, which in turn influence the formulation of relevant learning goals. Here we discuss how to formulate a set of learning goals based on a general top-down task-based design process (Figure 2).

This process starts with a set of presentation intents that specify the general presentation tasks to be achieved (e.g., Inform task). Using a set of *presentation plan rules*, the system can map the high-level intents onto a set of visual tasks (e.g., a visual Emphasize task). These visual tasks form an *presentation plan* that describes *what* desired visual effects to be achieved [Zhou and Feiner, 1998b]. *Visual plan rules* then map visual tasks (presentation plans) onto a set of concrete visual techniques, such as Highlight and Enlarge. These visual techniques define a *visual plan* that specifies *how* to achieve the desired effects. To form a complete visual presentation, all parameters of a visual technique (e.g., the scale amount of a Scale technique) must be specified. *Visual tuning rules* are used to determine these parameters.

We formulate three categories of learning goals for acquiring the three types of rules needed in this process. Figure 2 lists these learning goals along with their relevant learning spaces. To establish presentation plan rules, our main goal is to learn the mappings between presentation data/intents and visual tasks (Goal A.2). For example, a specific learning goal may be to learn how a Summarize intent is mapped to different types of visual tasks (e.g., Cluster or Emphasize). To refine vague presentation intents, another type of presentation plan rule is used to relate presentation intents to certain data properties. For example, one design rule may connect a pattern-seeking intent with certain data semantic structures, such as data clusters or data flows. This implies that the vague pattern-seeking intent can be achieved by a more specific data-cluster/data-flow displaying intent. Thus we formulate Goal A.1 to learn how particular presentation intents are related to data properties.

For visual plan rules, on the other hand, the learning focuses on extracting correlations among the data, visual techniques, and visual objects. Here all visual tasks are considered abstract visual techniques. Specifically, the first type of goal (Goal B.1) is to learn the relations between abstract and concrete visual techniques so that abstract techniques can be refined. To refine the abstract technique Emphasize, for example, we want to learn how it is related to concrete techniques, such as Highlight and Focus. Instead of relating to other visual techniques, certain visual techniques (e.g., Plot) are concerned with the creation of new visual objects (e.g., BAR-CHART or SCATTER-PLOT). To define a complex visual object, we need relate the object to its components. Thus we formulate goals (Goal B.2–3) to acquire various visual object construction rules.

We acquire visual tuning rules by achieving three types of learning goals. To determine a visual technique's parameters (e.g., the duration of Scale), Goal C.1 aims to learn how to specify various parameters based on data properties or other parameters (e.g., the duration in Scale may depend on the scale amount). Goals C.2–3 intend to learn how to determine specific visual attribute values (e.g., a specific position). Compared with Goal B.3, which stresses uncovering semantic relations among visual objects (e.g., a table heading is the identifier of a table), Goals C.2–3 emphasize

extracting the syntactic relations (e.g., a diagram layout).

Learning Features

Machine learning in rule acquisition is a process that infers condition-action-like patterns by comparing and classifying commonalities across observational data. To enable comparison and classification, we must express all data uniformly as a collection of learning features. *Learning features* are a set of data descriptors; for example, all visual objects may be described using their shape or color. As different types of data require different descriptors, we establish two feature spaces to describe two basic types of data involved in rule formulation. *Information feature space* contains a set of attributes that describe the presentation data, intents, and context. On the other hand, *visual feature space* consists of a set of properties that characterize various visual techniques and visual objects.

Information Feature Space. Information feature space is made up of three sub-feature spaces: data feature space, intent feature space, and context feature space (Table 1).

Data feature space contains a set of presentation-related data properties, all of which, except closeness, are extracted from a data characterization taxonomy [Zhou and Feiner, 1998a]. Here, *data closeness* is a quantitative measure of the semantic distance between presentation data. We add this property to our feature space, as it may affect visual mappings (e.g., using closeness for visual grouping).

We have also dropped some of the data properties (e.g., Form and Sense) that are in our characterization taxonomy. This is either because the property has appeared in other feature spaces (e.g., the data sense now is described by the role in the intent feature space), or it is too difficult to be described accurately. For example, we use Form in our taxonomy to characterize the high-level natural appearances of presentation data. While these high-level descriptions are too general for machine learning, lower-level accurate descriptions are too difficult to formulate because of the diversity of natural appearances.

Whereas the intent feature space describes a data's role at different levels of abstraction [Zhou and Feiner, 1998a], the context feature space depicts the properties of a presentation context (e.g., the display property).

Visual Feature Space. Based on previous visual analysis (e.g., Winn and Holliday, 1982; Kosslyn, 1989; Keller and Keller, 1993; Tufte, 1997), we have identified and summarized a set of features for describing various visual objects and visual techniques (Table 2). Compared with previous studies, which usually use a qualitative analysis, our feature set describes visual objects and techniques at a much finer granularity. This fine-grained description provides the foundation for automated graphics generation as well as for machine learning.

We use two sets of features to describe visual objects and visual techniques separately. For visual objects, we describe their structural and perceptual features. Specifically, *structural features* (ObjType and Part-of-Object) describe the compositional relationships between visual objects in a visual hierarchy [Zhou and Feiner, 1998a]. Our visual hierarchy contains five types of visual objects: visual discourse, visual frame, visual structure, visual unity, and visual primitive. Each higher-level visual object is recursively built using the objects below it (e.g., a visual unity contains visual primitives). *Perceptual features*, on the other hand, describe the perceptual relationships between visual objects (Proximity, Similarity, Continuation, Closure, Emphasis, Sequence, and Symmetry). To facilitate machine learning, these perceptual features must be described accurately (e.g., using quantitative measure). For visual techniques, we stress their effects (Effect/SideEffect), their role in a visual discourse (Role), and their execution style (Style).

Features of specific visual objects or techniques can also be added. For example, we can describe a visual primitive such as color by its hue and intensity, or a technique by its temporal duration. But here we only focus on the shared properties of all visual objects or techniques.

Feature Extraction. Unlike in other domains, where features are usually described by numerical or nominal values, many of our features are specified using structures. For example, the value domain for data feature Domain is a tree, which can be used to specify the semantic category of a data instance at multiple levels of abstraction (a person is an Entity and Human). Moreover, a feature may have multiple value domains. For example, the value domain of visual feature Part-Of-Object may be a set of parts of a higher-level object, such as a table chart's header or cell, or a set of ele-

Data Features	Definition	Value Domain
Domain	Semantic category of data (data may belong to multiple categories)	KB Ontology
Type	Decomposability of a data	Atomic/Composite
Transience	Whether data change with time	Static/Dynamic
Scalability	Whether data can be presented using scalable visual variables (e.g., size)	Coordinates/ Amounts
Importance	Whether a piece of data is more important than others. Importance may be measured by normality/abnormality, routine/urgency	0: unimportant 1: important
Ordering	The order among a set of data	Quantitative/Ordinal/Nominal
Closeness	How closely two data are related	Numerical
Relation	How data are related	Relation hierarchy
Cardinality	The number of values in a relation	Numerical

Intent Features	Definition	Value Domain
Part-of-Discourse	The role of the data in a visual discourse	Focus/Peripheral/ Transition/
Role	The illustrative role of the data (e.g., an Identifier or Locator)	Visual task taxonomy

Context Features	Definition	Value Domain
Color	The available color of the a computer display or a number of colors available	Black-White/ Grayscale/Color
Resolution	The resolution of a display	Integer × Integer
Size	The size of a display	Integer × Integer
Medium	The available media	Visual/Multimedia

Table 1. Input information features (presentation data/intent/context)

Object Features	Definition	Value Domain	Technique Features	Definition	Value Domain
ObjType	The object type in the visual hierarchy	Visual Hierarchy	Effect	The effect of a visual technique (e.g., Tabulate/Reposition)	Effect hierarchy
Part-of-Object	The specific constituent of a visual object (e.g., header of a table-chart)	Structural components	VisRole	The role of a technique in a visual discourse	Core/Context/
Proximity	How closely two visual objects are close to each on the screen	Numerical	SideEffect	The side effect of a technique (e.g., Zoom may lose context)	Effect hierarchy
Similarity	How similar two visual objects appear alike	Numerical	Style	The type of transition used	Instant/Animated
Continuity	Whether one visual object can be perceived as a continuation of another object	True/False			
Closure	Whether an object spatially encloses other objects	True/False			
Emphasis	Whether a visual object looks more prominent than others	Ture/False			
Sequence	The order of visual objects to be perceived.	Numerical			
Symmetry	Whether visual objects appear symmetry	True/False			

Table 2. Visual features (visual object/technique)

ments in a lower-level object, such as a visual unity's geometry, color, and position. The complexity of our feature domains usually produces an overwhelmingly large number of possible feature assignments. To systematically determine relevant features and their assignments, we use a learning-goal-based approach.

Initially, we may pick out all possible relevant features based on a goal description. We then assign each selected feature using its most general value. Because of the general description, observational data may not be well distinguished and the learned concepts may appear trivial. In this case, we update features one at a time using more specific values. This incremental process helps avoid over specifying the data as nothing can be generalized. Moreover, this systematic trial-and-test approach prevents us from missing out useful features or feature assignments. In addition, we can usually make a good initial estimate based on the specific goal description (see Experiments section).

Learning Strategies

In this section, we discuss how to use different learning strategies to acquire rules in different situations. In particular, we consider four common learning strategies: learning from examples, learning from observation, learning from instruction, and learning by analogy.

Learning From Examples. Intuitively, one way to acquire graphics design rules is to learn from a set of presentations created by design experts. This type of learning usually generalizes a new concept based on a set of given examples (training data) of a particular concept (target). Thus it is known as *learning from examples* [Mitchell, 1997].

A key aspect to the success of learning from examples lies in the specification of examples. We express each training example using two vectors:

$$\text{Example}_i = \{\boldsymbol{f}_i, \boldsymbol{t}_i\} \text{ where } \boldsymbol{f}_i = (f_i[1], \ldots, f_i[\mathsf{K}]) \;\&\; \boldsymbol{t}_i = (t_i[1], \ldots, t_i[\mathsf{M}]).$$

For the ith example, $\boldsymbol{f}_i$ is a feature vector that specifies total K features and $\boldsymbol{t}_i$ is a target vector that describes M targets. In our domain, each example is a presentation; $\boldsymbol{f}_i$ is a set of features of design facts and $\boldsymbol{t}_i$ is a set of design actions.

Among various learning-from-examples algorithms [Ma and Ji, 1999], a decision-tree-based approach is considered a better one because of its predictable training time and comprehensive output (e.g., Breiman et al., 1984; Quinlan, 1993). We use this approach to induce a set of classification rules from the training data. These classification rules categorize the mappings between a set of design facts and design actions (targets). For example, one classification rule may look like this:

$$\mathsf{IF}\ f[1] = \mathsf{VAL}\ \mathsf{THEN}\ t[2] = \mathsf{T}\ [80.0\%].$$

Here 80% is the confidence factor, the probability of this mapping ($f[1] = \mathsf{VAL} \Rightarrow t[2] = \mathsf{T}$) occurrence.

To learn design rules from examples, we not only need to collect relevant sample presentations, but also need to define very specific learning goals. A specific learning goal helps identify relevant features, since presentation samples can be described at different levels of abstraction or detail. For example, to learn how a data object can be encoded using different *types* of visual primitives such as color or shape, we may not care about the particular colors used in a presentation (see a detailed example in next section).

Learning From Observation. Since learning-from-examples requires a large set of training data that must be expressed using specific features, this approach may not be appropriate in certain situations. Recall that we want to learn how visual features affect a similarity measure in Formula (1). If the similarity is exhibited through visual patterns, it would be very difficult to describe these patterns precisely in a set of examples. In this case, an alternative approach may be used. Instead being given a set of examples of a particular concept (e.g., similar visual patterns), a learning system must infer the concept by observing the characteristics of the relevant environment (e.g., visual features) [Michalski and Stepp, 1983]. This is known as *learning from observation* (unsupervised learning).

Since learning from observation requires a tremendous amount of inference, researchers have developed various practical approaches to simplify the learning process. Most relevant to our domain are clustering algorithms and optimization approaches. Using clustering algorithms, we can characterize various data relations; for example, we may use pattern clusters to measure visual similarity [Duda and Hart, 1973]. We can also use optimization approaches to approximate the desired relations when an exact relation is difficult to establish (see more in Experiments).

Learning From Instruction. It is desirable that a generation system can augment its rule base by directly taking instructions from a user. For example, users may want data to be presented in a particular format that cannot be suggested by existing rules. This type of learning is known as *learning from instruction*. To instruct the system, we must represent instructions in a machine-understandable format. As knowledge representation is an active research topic itself, alternative approaches have been used to learn from instruction. One of these approaches is known as *reinforcement learning* where machine-learned concepts are incrementally adjusted based on specific user feedbacks [Mitchell, 1997]. For example, users may score the learning results to help the system adjust the weights in Formula (1). Since the user can only give instructions on predefined concepts, reinforcement learning is somewhat relieved from the burden of processing complex knowledge.

Learning By Analogy. New rules can also be acquired by transforming existing rules that address situations that bear strong similarity to the desired new situation. For example, the rule that describes how to show a car's parts (e.g., engine and trunk) may be transformed to display a tree's parts (e.g., root and leaves). Since this process requires that facts analogous in relevant parameters be retrieved from memory, transformed, and applied to the new situation, this type of learning is known as *learning by analogy* [Mitchell, 1997]. In the above example, the analogy is drawn upon two parameters: the presentation goal (show an object's parts) and the inherent property of the objects involved (both cars and trees have physical forms). To determine relevant parameters that help draw a meaningful analogy, we need to characterize the conditions of existing rules (e.g., using object property to describe the car rule).

In addition to acquiring new rules by analogy, we can also generalize rules based on the commonalities between previous and new situations and the successful applications of the modified rules in the new situation. For example, if the transformation of the car rule is successfully applied to a tree, we may derive a generalized rule to handle how to convey the parts for any object that has a physical form.

Experiments

We have designed and conducted two experiments to demonstrate the application of machine learning in graphics rule formulation. The first experiment demonstrates how to establish a higher-level design rule by learning the correlations between visual techniques from a set of sample presentations. The second illustrates how to establish a low-level visual mapping using an optimization approach.

Learning Visual Technique Correlations

Using a learning-from-examples approach, we describe how to achieve the type of learning goal described by B.1 in Figure 2.

Learning Goal. Specifically, we want to learn the correlations between a high-level visual technique Reveal and three lower-level visual techniques, Expose, Separate, and Overlay. We express this learning goal as:

(3). Reveal $<o_1, o_2> \Rightarrow$ {Expose, Separate, Overlay}.

Here, Reveal $<o_1, o_2>$ is a visual goal that aims to convey a data object o_2 that is usually "hidden by" data object o_1. The three lower-level techniques describe the specific ways of bringing out the hidden object o_2. In particular, Expose uses standard graphics techniques, such as CutAway, Open, or SetTransparency, to expose the hidden object (Figure 3a). Separate reveals the hidden object by splitting and slicing the representation of o_1 (Figure 3b). Unlike Expose and Separate, which modify the visual objects that hide o_2, Overlay places the representation of o_2 directly on top of the existing display (Figure 3c).

Training Data and Tool. Our training data is collected from three books [Keller and Keller, 1993; Wurman, 1996; Wildbur and Burke, 1998]. Because of the diversity of these presentations, we may hope to avoid biased conclusions that might be only meant for a specific application domain. From about 500 illustrations, we have identified 60 presentations that are related to visual revealing.

Based on our learning goal in Formula (3), we initially extract five features from Table 1 and Table 2 as our learning input. We use one target output to specify one of the three Reveal techniques: Expose, Separate, and Overlay. Among the five features, Relation describes the semantic relation between the two data objects, while Domain_1 and Domain_2 specify their semantic categories. ObjType_1 and ObjType_2 describe the type of visual encoding for the two data objects. We exclude indistinguishable or irrelevant features; for example, feature Part-of-Discourse for o_2 is always

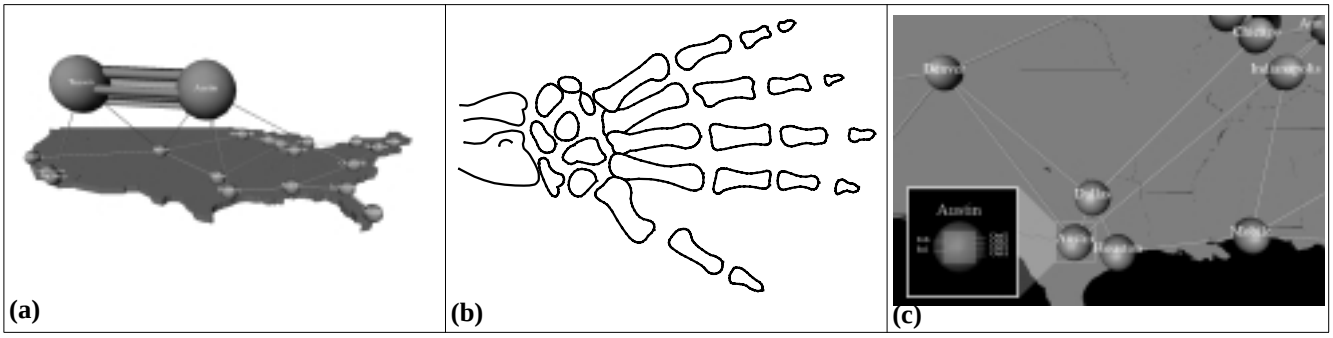

Figure 3. Three visual revealing techniques

Relation **Values**	**Value Definitions**
Constituent	o_2 is a part of o_1
Attributive	o_2 is an attribute of o_1
Enumeration	o_2 is an enumeration of o_1
Specify	o_2 is a specification (e.g., detail) of o_1
Instance	o_2 is an instance of o_1
Subconcept	o_2 is a sub-concept (subclass) of o_1

Figure 4. Possible feature values for Relation

Focus, and feature Ordering is irrelevant to our goal.

All five features can be described at multiple levels of abstraction. For example, Figure 4 lists three top-level Relation values (Constituent, Attributive, and Enumeration) and three refined relation values for Enumeration (Specify, Instance, and Subconcept). Assigning top-level values to all five features, we obtain the most general feature assignment. By this assignment, the three samples in Figure 3(a–c) are:

a. Reveal (PhysicalLink, VirtualLink):
Constituent, Entity, Entity, VisUnity, VisUnity, *Expose*
b. Reveal (Hand, Hand-Parts):
Constituent, Entity, Entity, VisUnity, VisUnity, *Separate*
c. Reveal (Node, Node-Port) (inset)
Enumeration, Entity, Entity, VisUnity, VisUnity, *Overlay*

The first five values in turn describe the five features: Relation, Domain_1, Domain_2, ObjType_1, and ObjType_2; and the last one is the target (in *italic*).

In this experiment, we use a decision-tree-based system called C4.5 to achieve our learning goal [Quinlan, 1993].

Experimental Results. We train C4.5 on the 60 presentations using 6-fold cross-validation, a standard procedure when the amount of data is limited. Using the most general feature assignment, we learn five classification rules with an overall classification error 30%:

1. Domain_1 = Concept ∧ ObjType_2 = VisStructure ⇒ Overlay [87.1%]
2. Relation = Enumeration ∧ ObjType_2 = VisStructure ⇒ Overlay [84.1%]
3. Relation = Enumeration ∧ Domain_1 = Concept ⇒ Overlay [84.1%]
4. Domain_1 = Entity ∧ ObjType_2 = VisUnity ⇒ Expose [56.7%]
5. Relation = Constituent ∧ Domain_1 = Concept ∧ ObjType_2 = VisUnity ⇒ Separate [50.0%]

As these rules indicate a strong correlation between the visual encoding of o_2 (ObjType_2) and Reveal techniques, they may not be useful in predicating a Reveal technique since ObjType_2 might be unknown when the technique needs to be decided. By dropping ObjType_2, we get four new rules but with an error of 35.7%:

5. Relation = Enumeration ∧ ⇒ Overlay [74.0%]
6. Domain_1 = Concept ∧ ⇒ Overlay [70.2%]
7. Domain_1 = Entity ∧ Domain_2 = Concept ⇒ Expose [58.7%]
8. Relation = Constituent ∧ Domain_1 = Entity ⇒ Expose [48.8%]

To reduce the classification error, we have systematically varied feature assignments from the most general to more specific ones. Consequently, an assignment of feature ObjType_1 has produced a set of useful rules, which can be used to predict Reveal technique based on the visual encoding of o_1 with a relatively low error (Figure 5).

ObjType_1 **Values & Definitions**	**Rules** (overall error 21.7%)
Icon: an iconic representation **Label**: a textual representation and its variations (e.g., buttons) **Symbol**: a 3D graphical representation **VisStructure**: a visual schematic drawing (e.g., map)	a. ObjType_1 = Icon ⇒ Overlay [85.7%] b. ObjType_1 = Label ⇒ Overlay [82.0%] c. ObjType_1 = Symbol ⇒ Expose [60.0%]

Figure 5. The design rules for Reveal and the salient feature

Compared with the hand-crafted rules currently used in our generation system, the learned rules give similar design suggestions. But the learned rules are more concise (with fewer conditions), and also have a quantitative confidence factor, which is missing in our hand-crafted rules.

Learning a Visual Attribute Mapping

In this example, we demonstrate how to establish a visual mapping between data semantic properties and specific visual attributes (Goal C.2 in Figure 2).

Learning Goal. Specifically, our goal is to learn how to best map data closeness onto visual proximity:

(4). $\text{Closeness}(o_1, ..., o_n) \Rightarrow \text{Proximity}(v_1, ..., v_n)$.

Here v_i is the visual encoding of the ith data object o_i. Like other multidimensional scaling applications [Cox et al., 1994], we must establish optimization functions to model the mapping between data properties (closeness) and visual features (proximity). As will be seen, however our special constraint here is to determine the order of nominal data to reveal embedded patterns, as nominal sets usually are not encoded by a visual sequence [Mackinlay, 1986].

Data and Tool. Our data is a large set of events collected from recorded event logs in a computer network [Ma and Hellerstein, 1999]. Each event has a host name (source of event), event type (e.g., a connection was lost), and a time stamp. In this experiment, we used over 10,000 events generated by 160 hosts with 20 event types in a 3-day period.

To model the data closeness, we first define the semantic distance between two events (event_i and event_j) using Euclidean distance:

$$d_{ij} = w_t(t_i - t_j)^2 + w_h I(h_i, h_j) + w_i I(e_i, e_j) .$$

Here t_i, h_i, and e_i denote the ith event's time, host, and type; and w_t, w_h, and w_e are weights for these three attributes. In our experiment, w_h and w_e are set to **1**, and w_t is the inverse of smallest visible time unit. Function $I(a, b)$ is a binary function that measures distance between two nominal values. $I(a, b)$ is **1** if a and b are the same; otherwise it is **0**.

Suppose that we want to examine event data along two dimensions: time and host. These two dimensions are then encoded using axis positions based on their data characteristics (i.e., time is a quantitative set and host is a large nominal set). We can then define the visual proximity between two events' visual encoding v_i and v_j:

$$v_{ij} = w'_t(x_i - x_j)^2 + w'_h(y_i - y_j)^2.$$

Here w'_t and w'_h use the same values as w_t and w_h in the data distance measure, while x and y encode the positions of event time and host on the X and Y axes, respectively.

Individually mapping every d_{ij} onto v_{ij} is very difficult, if not impossible, as we have over 10,000 events. Moreover, such individual mappings only ensure the local effectiveness of a presentation. Thus we model all mappings at once as a global optimization problem. In particular, we seek an optimal placement of all event hosts on the *Y* axis, as event positions on the *X* axis are already ordered by time. This encoding minimizes the *overall* differences between the data closeness and visual proximity:

$$E = \sum_{i=1}^{n} \sum_{j=1}^{n} (d_{ij} - w v_{ij})^2$$

where w is a weighting factor and n is the total number of events.

Experimental Results. Using a greedy algorithm with a linear complexity, we are able to find an optimal host placement on the *Y* axis. This placement in fact provides an optimal mapping between data closeness and visual proximity. In other words, this mapping ensures that all events that are semantically close to each other also appear visually close to each other (Figure 6b). In comparison, we use Figure 6(a) to show event relations when hosts are randomly placed on the *Y* axis. This experiment also demonstrates how an optimization approach may be used to find optimal visual mappings (design rules), when an exact mapping is difficult to acquire.

Conclusions & Future Work

In summary, we have addressed several fundamental aspects toward applying machine learning to graphics rule acquisition. We first outline three learning spaces to identify different learning problems related to rule formulation. Using a design-process-driven approach, we illustrate how to systematically formulate specific learning goals. To describe various rule conditions and actions, we present two feature spaces and discuss how to extract relevant features. Based on the characteristics of various learning strategies, we examine their applicability in our learning domain. Moreover, we have also designed and conducted two experiments to demonstrate how practical learning techniques can be used to acquire different design rules.

One of the main tasks on our agenda is to develop a learning algorithm that can take advantage of structured features (e.g., the domain of our visual feature objType is a tree), since existing tools focus only on numerical or nominal features. As we deal with potentially a large set of features in graphics synthesis, we would like to explore how to automatically extract salient features. We also intend to create a large graphics corpus to facilitate fine-grained graphics design analysis using machine learning. To evaluate the quality of learning results, we would like to conduct formal studies to compare the designs created by expert-derived and machine-learned rules.

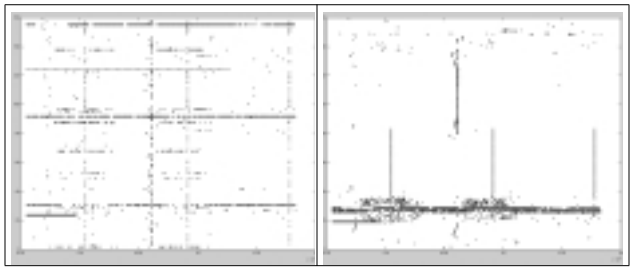

Figure 6. Data closeness helps revealing data patterns

References

Breiman, L., Friedman, J., and Stone, C. (1984). *Classification and Regression Tree*. Wadsworth.

Casner, S. (1991). A task-analytic approach to the automated design of graphic presentations. *ACM Trans. on Graphics*, 10(2):111–151.

Cox, T., Cox, M., and Cox, D. (1994). *Multidimensional Scaling*. CRC Press.

Duda, R. and Hart, P. (1973). *Pattern Classification and Scene Analysis*. Wiley.

Keller, P. R. and Keller, M. M. (1993). *Visual Cues: Practical Data Visualization*. IEEE Computer Society Press and IEEE Press.

Kosslyn, S. (1989). Understanding charts and graphs. *Applied Cognitive Psychology*, 3:185–226.

Lohse, G., Biolsi, K., and Rueter, H. (1994). A classification of visual representations. *Communications of the ACM*, 37(12):36–49.

Ma, S. and Hellerstein, J. (1999). EventBrowser: A flexible tool for scalable analysis of event data. In *DSOM*.

Ma, S. and Ji, C. (1999). Performance and efficiency: Recent advances in supervised learning. *Proc. of the IEEE*, 87(9):1519–1536.

Mackinlay, J. (1986). Automating the design of graphical presentations of relational information. *ACM Trans. on Graphics*, 5(2):110–141.

Marks, J. (1991). A formal specification scheme for network diagrams that facilitates automated design. *J. of Visual Languages and Computing*, 2(4):395–414.

Michalski, R. and Stepp, R. (1983). Learning from observation: Conceptual clustering. In Michalski, R., Carbonell, J., and Mitchell, T., editors, *Machine Learning: An AI Approach*, chapter 11, pages 331–364. Morgan Kaufman.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.

Pan, S. and McKeown, K. (1998). Learning intonation rules for concept to speech generation. In *Proc. ACL '98*, volume 2, Montreal, Canada.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufman.

Roth, S. F. and Mattis, J. (1990). Data characterization for intelligent graphics presentation. In *Proc. ACM CHI '90*, pages 193–200.

Roth, S. F. and Mattis, J. (1991). Automating the presentation of information. In *Proc. IEEE Conf. on AI Applications*, pages 90–97.

Seligmann, D. and Feiner, S. (1991). Automated generation of intent-based 3D illustrations. *Computer Graphics*, 25(4):123–132.

Tufte, E. R. (1997). *Visual Explanations: Images and Quantities, Evidence and Narrative*. Graphics Press, Cheshire, CT.

Wildbur, P. and Burke, M. (1998). *Information Graphics: Innovative Solutions in Contemporary Design*. Thames and Hudson.

Winn, W. and Holliday, W. (1982). Design principles for diagrams and charts. In Jonassen, D., editor, *The Technology of Text*, volume 1, pages 277–299. Educational Technology Publications.

Wurman, R. (1996). *Information Architects*. Graphics Press, New York.

Zhou, M. and Feiner, S. (1998a). Automated visual presentation: From heterogeneous information to coherent visual discourse. *J. of Intelligent Information Systems*.

Zhou, M. and Feiner, S. (1998b). Visual task characterization for automated visual discourse synthesis. In *Proc. ACM CHI '98*, pages 292–299.