Concept Acquisition for Dialog Agents

Brandon Rohrer
Sandia National Laboratories
USA
brrohre@sandia.gov

***ABSTRACT**: Dialog agents capable of autonomously acquiring new concepts are likely to be more powerful than those relying on a fixed set of preprogrammed concepts. kx-trees provide a novel unsupervised learning method for concept acquisition. Through online, incremental, divisive, binary-tree-based clustering, it organizes raw sensory experiences into low-level concepts. Using the same mechanism, it can organize low-level concepts into higher level concepts to create concept hierarchies of arbitrary depth. This paper contains a description of kx-trees' operating principles, illustrative examples, and a discussion of how they relate to existing clustering algorithms and similar biologically-inspired tools.*

Key words: Dialog agents, binary- tree clustering, Unsupervised learning, Concept hierarchy

Received: 29 May 2010, Revised 28 June 2010, Accepted 5 July 2010

1. Introduction

The ability to interpret dialog in terms of concepts greatly increases the perceived usefulness and fidelity of dialog agents. For instance, identifying a novel word "Liam" with the concept of <boy's name> could allow a dialog agent to ask "When did you last speak with him?" without having any other prior knowledge. Parsing dialog according to the concepts associated with words and phrases is a common strategy in dialog agents and has proven itself very effective. Part of speech tagging is the most common example of this; <noun> and <verb> are examples of extremely general concepts. In most, if not all, agents using concepts, the number and nature of concepts are determined at the time of system creation by the programmer. The work described here provides a mechanism for acquiring new concepts based on an agent's experience. It allows an agent to be created without any pre-defined concepts, generating as many of them as its experience warrants over the course of its lifetime.

In order to create concepts, instead of relying on hand crafted concept definitions, sensory experiences must be grouped into concept sets. These first level concepts can then be grouped into second level concept sets, which in turn may be grouped into third level concepts, etc. The process of forming elements into groups, known as clustering, is a common machine learning problem, falling into the family of methods known as unsupervised learning. (A wide-ranging review of unsupervised learning methods can be found in [6, ch. 10].)

In this work, sensory experiences (state observations) are grouped into a small number of clusters. Each dimension in the state space represents a single state variable. What distinguishes this work from
previous unsupervised learning methods is that it does not assume that state dimensions are ordered fields. That is, this work does not assume that 1 is less than 2 or that the distance from 1 to 2 is less than the distance from 1 to 100. As a result, this method can be applied to many different data types, including continuous, discretized, categorical, binary, and hybrid data.

The clustering algorithm described here is based on a binary decision tree. When it is initialized, the entire state space represents a single cluster. The state space is then repeatedly subdivided into regions, with some of the regions being designated as clusters and others as non-cluster regions. Each subdivision is represented by splitting a leaf node into two new leaves. A subset of the tree's leaves are the clusters. The dimension along which to divide each leaf is chosen so as to maintain and isolate clusters as much as possible. A tree created by dividing regions along a constant value of a single dimension is a kd-tree. A tree that chooses when and where to divide solely on the basis of the data distribution is said to have the X-property [5]. In order to distinguish between this and other tree-based tools (including Decision Q-Trees [8], S-trees [23], and T-Trees, an extension of Continuous U-Trees [27]) it will be referred to as a kx-tree.

In this paper, I present a detailed description of kx-trees,2 together with two implementations demonstrating their operation. One kxtree is used to find clusters in a two-dimensional state space. A second kx-tree is used with digital images to find visual primitives similar to those found in human cortical area V1.

2. Method

kx-trees are conceptually similar to other decision tree based algorithms, but differ in the specifics of their implementation. Although the most well known implementations of decision trees are CART [2] and C4.5 [19], a concise overview of a wide variety of decision trees is given in [24]. A decision tree is defined by how it answers the three questions 1) When to split a node, 2) Where to split that node, and 3) When to stop splitting a node. (Some trees also answer a fourth question—When and how to prune branches from the tree—but kxtrees do not use pruning.) After describing the structure and populating process of a leaf, this section answers those questions for kx-trees.

2.1 Leaf structure

Each leaf represents an explicitly bounded region of the state space. When defining the initial leaf that encompasses the entire state space, upper and lower bounds on all input dimensions must be known or assumed. As a leaf is divided, the bounds of its children are created using the decision boundary. In this way, the leaves form a nonoverlapping set of regions that completely cover the state space. (See Figure 1) Each leaf is a hyperbox, defined by minimum and maximum values in each dimension of the state space.

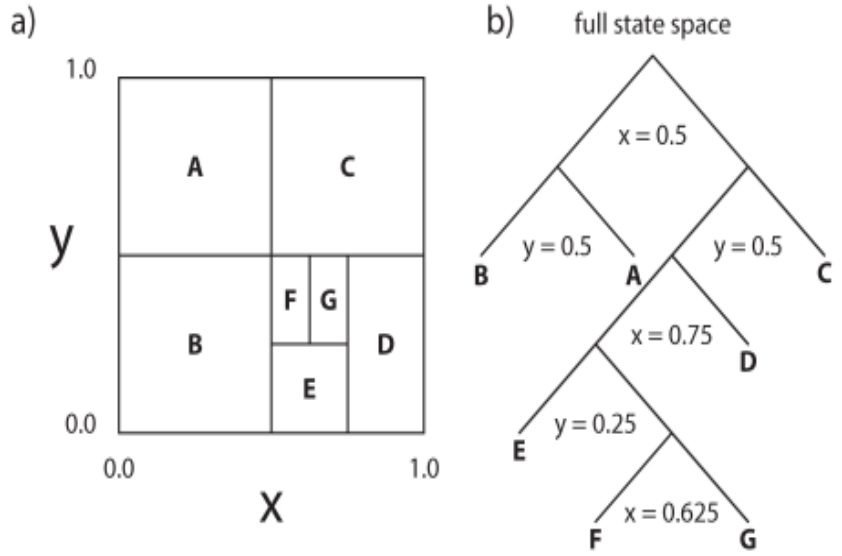

Figure 1. a) A partitioned state space and b) the decision tree representation of that partitioning. The top node in the tree represents the entire state space. It also indicates the division of the space into its left and right halves along the line $x = 0.5$. Leaf nodes in the tree represent individual regions as labeled. Any internal node on the tree represents the union of all of the leaf node regions that fall under it

[1]The unsupervised learning method kx-trees and its operating principles were first presented at the 2010 Linguistic and Cognitive Approaches to Dialog Agents Workshop, [20] from which papers in this issue were taken. An account was subsequently published in the Ninth International Conference on Development and Learning. [21] Relevant portions of that written description are included here.

Each leaf represents a single region. Each node on the tree represents a super-region, the combination of the regions defined by its children, which themselves may be either leaves or branch nodes. Most tree-based learning tools are supervised learning algorithms, rather than unsupervised. Observations of input states are typically accompanied by categories, in classification problems, or values, in regression problems. kx-trees differ from these. The leaves of a kxtree do not provide an estimate of a reward, value function, or class membership. Instead, kx-trees identify locally dense regions of state activity. In addition, kx-trees subdivide clusters once they exceed a predetermined size. The motivating assumption behind kx-trees' partitioning heuristic is that useful concepts capture a roughly fixed number of observations. If a concept represents very few observations, it provides only limited abstraction, and if it represents a very large number of observations, it may be too general to be useful.

2.2 When to split a node

A kx-tree node becomes eligible to split when the number of observed states it contains exceeds a user-defined threshold, *M*. Each observation that falls within a given region of state space is a *member* of that region. When a node has more than *M* members, it is considered ripe for bisection. However, each member has a finite lifetime, *L*, also defined by the user. As a result, a node is ready to split when more than *M* of the last *L* state observations fall within its region of state space.

2.3 Where to split a node

The objective of a kx-tree is to find clusters of observations and bound them tightly. Each cluster represents a concept, a feature, or a closely related group of states. Tighter bounds give a more concise concept definition. Therefore, when a node is bisected, the dimension along which to split and the value at which to split it are chosen such that the cluster is preserved as much as possible. For all admissible splits, the split which maximizes the disparity between the number of members in each child is selected. If more than one candidate split achieves the maximum value, then the split value is randomly selected from among them. (See Figure 2.)

More formally, for an admissible split (*d*, *x*) along dimension *d* at value *x*, the quality of the split, *q* is given by

$$q_{d,x} = |m_a - m_b| \quad (1)$$

where m_a is the number of members from the parent node that would fall under one of its children for that split and m_b is the number of members that would fall under the other child node. Quality is symmetric with respect to m_a and m_b. The winning split fulfills the condition

$$(d, x)_{winner} = \max_{d \in D} (\max_{x \in X_d} q_{d,x}) \quad (2)$$

where *D* is the set of all state space dimensions and X_d is the set of all admissible split values for dimension *d*. The admissible split values in X_d divide the region along dimension *d* roughly in half. X_d is generated by taking a number of evenly-spaced values along *d*. If d_{min} and d_{max} are the limits of *d* in the region to be divided, then *N* admissible split values can be generated:

$$d_{low} = d_{min} + \frac{1}{4}(d_{max} - d_{min}) \quad (3)$$

$$d_{high} = d_{min} + \frac{3}{4}(d_{max} - d_{min}) \quad (4)$$

$$x_i = d_{low} + (d_{high} - d_{low}) \frac{i}{N+1}, \forall i \leq N \quad (5)$$

The resulting values of X_d fall between, but do not include, one quarter and three quarters of the range of *d*. *N* is a user-selected constant.

2.4 When to stop splitting a node

Due to the fact that members have a limited lifetime, an explicit stopping criterion is typically unnecessary. The splitting condition of accumulating more than *M* members within any interval *L* becomes harder to achieve as regions get smaller. With any continuous distribution of observations, the member accumulation rate will approach zero as region size approaches zero. However, when processing discontinuous distributions (as with categorical or discretized data), observations may be

grouped at a single value, prompting endless repeated splits. This can occur whenever a region's range in a given dimension is less than the discrete resolution in that dimension. A method for handling this degenerate case is to specify a threshold for the minimum region size in each dimension. This can either be a constant $\in$ that is the same for each dimension or a vector E with separate values for each dimension. Further divisions are not allowed along any dimension for which $d\max - d\min < \in$. The same effect could be achieved by adding a small amount of noise ($\in$) to each observation.

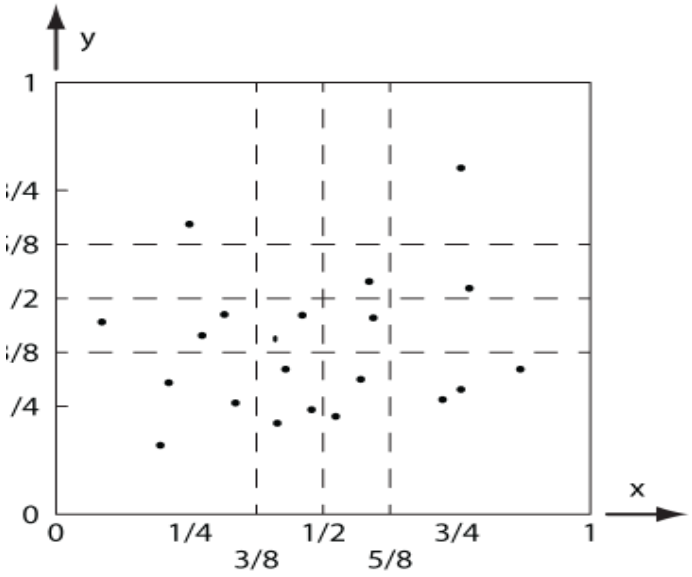

Figure 2. Illustration of the process of choosing where to split a node. For a node in a two dimensional state space, bounded by $\{x, y\} = \{0,1\}$ with $N = 3$, there are six possible ways to divide the node. Using Equation 5, the three potential splits on each axis occur at 3/8, 1/2, and 5/8. The quality of each split is determined by the imbalance that results between the number of observations falling into each child node. The greater the imbalance, the higher the quality of the split. For instance, applying Equation 1 to the split (x, 1/2) yields $q_{x,1/2} = |m_a - m_b| = |12 - 9| = 3$. Calculating the quality of all potential splits reveals that (y,5/8) best preserves the clustering of the observations. It would be selected as the split location for this example

2.5 Clusters

kx-trees seek to isolate groups of members into clusters. A leaf node becomes a cluster when it collects more than $M/2$ members. Thus, when a node is divided, one of the children will become a cluster and the other will not (although it may eventually become one too), and the parent node ceases to be a cluster. Once a leaf node is designated a cluster, it remains one, even if the number of members it contains falls below $M/2$. Cluster nodes represent concepts, the categories into which most of the data fall, or features which are common in the data.

Once clusters have been formed, they serve as a means of interpreting new states. A single state observation is a point in state space; if it falls within the hyperbox of a cluster, the data point can be represented more abstractly as that concept. But typically, clusters do not provide complete coverage of the state space. In order to find a conceptual interpretation of points that do not fall within a cluster, a similarity or distance measure can be used to identify the nearest clusters. The similarity measure used by kx-trees represents the fraction of dimensions in which an observation matches a cluster. The similarity,σ, between a state s and a cluster c in an n-dimensional space, S, is given by

$$\sigma(s,c) = \frac{\sum_{i=1}^{n} in(s,c,i)}{n} \quad (6)$$

Where

$$in(s,c,i) = 1, \text{ if } s_i \geq c_{i\,min} \text{ and } s_i < c_{i\,max}$$
$$= 0 \text{ otherwise} \quad (7)$$

where s_i is the value of s, c_i min is the minimum value of c, and c_i max is the maximum value of c, all in the i^{th} dimension. The corresponding distance measure,δ, is given by

$$\delta(s,c) = 1-\sigma(s,c) \quad (8)$$

If s falls within c for all dimensions, then $\delta = 0$ and $\sigma = 1$. If s falls within c in only half of the dimensions, $\delta = \sigma = 1/2$. The distance measure takes no account of how far outside a cluster a state may fall. Only matching and non-matching are reflected in the distance. In this way, the distance measure is similar to Hamming distance, but is extended to apply to real valued vectors and is normalized to fall within the range [0, 1]. In fact, for binary state spaces, the modified Hamming distance of Equation 8 reduces to the actual Hamming distance divided by the number of state space dimensions, n. It should be noted that this is a departure from many other unsupervised learning approaches, which use Euclidean distance. The modified Hamming distance and modified Hamming similarity avoid making the assumption that the state space is well scaled or that its dimensions are ordered fields, and this allows it to be applied to a wider range of problems.

2.6 Coordinate transformation to feature space

Combined with the modified Hamming similarity, clusters can be used to reinterpret states. The vector of similarities between a state and each of the clusters translates that state into a new state space, S', in which each dimension represents a cluster. The new state, s', is given by

$$s' = [\sigma(s,c_1), \sigma(s,c_2),...,\sigma(s,c_{n'})] \quad (9)$$

where n' is the total number of clusters and the dimensionality of the new state space.

Multiple clusters may lie very near each other in S, such that a state falling completely within one cluster may have high similarity to other clusters as well. This would cause dimensions in S' to behave as if they were highly non-orthogonal.

A pseudo-orthogonal state space can be formed by using a winner-take-all projection onto S':

$$s' = [0,0,0,..., \sigma(s,c_i),... 0,0,0] \quad (10)$$

where similarity for the i^{th} dimension is a maximum:

$$\sigma(s,c_i) \geq \sigma(s,c_j), \forall j \neq i \quad (11)$$

If this condition holds for more than one dimension, the winner is randomly selected from among them. By forcing s to project onto only one dimension in S', the relative contributions of neighboring clusters remain clearly distinguishable.

Transforming observations from S to S' re-expresses them in terms of more complex features of the input space. This process converts low level data into somewhat higher level features. In the cases where $n' < n$, it also results in a dimensionality reduction.

It is noteworthy that the inputs and the outputs of kx-trees' coordinate transformation are both in the same form: vectors with realvalued elements. As a result, multiple kx-trees can be arranged hierarchically with the output of one serving as input to another. It is expected that with each level in the hierarchy, the feature representation would become increasingly abstract. This phenomenon is the subject of current research.

3. Simulations and Results

3.1 Simulation 1: Clustering example

The first simulation reported here is a simple but illustrative example. In it, a pair of arbitrarily selected two-dimensional normal distributions were used to generate random points in a two-dimensional state space. The defining parameters for the kx-tree were chosen as follows:

Anomaly lifetime,	$L = 100$
Maximum number of members,	$M = 40$
Number of split candidates per dim.,	$N = 101$
Minimum category size,	$\in = 0$

The kx-tree processed the points as they were generated, creating a partition of the state space and finding the cluster leaves associated with it.

The results for simulation 1 are shown in Figure 3. The two normal distributions produced two groups of points, and the recursive partitioning of the space produced cluster nodes corresponding to the densest portion of each group. The tree in Figure 4 provides an alternative representation of the partition in Figure 3 and shows two major branches, one corresponding to each of the clusters. Figure 5 shows how the state space, *S*, maps onto the feature space, *S'*, defined by the clusters. Simulation 1 was run until a stable tree structure was achieved, that is, no new nodes were created for a substantial period of time. Simulation 1 provides an illustration of kx-tree's operation on a simple problem and is intended to provide an intuitive basis for understanding the results of simulation 2.

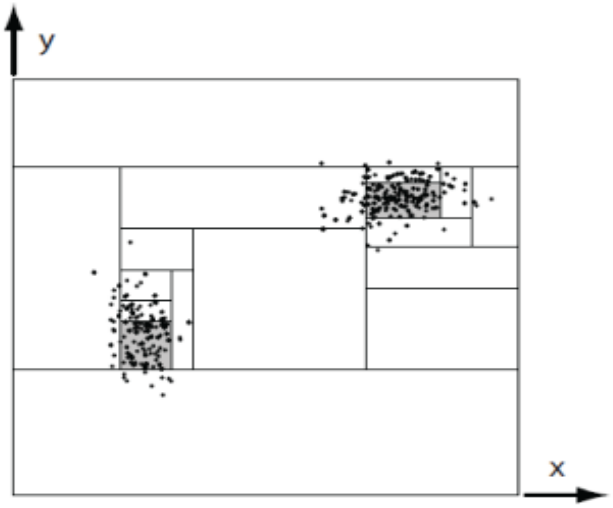

Figure 3. Two dimensional state space as partitioned by a kx-tree. The two normal distributions produced clearly separated groups of points. Each bisection is shown by a line on the bisection boundary. Each region represents a leaf node. Cluster nodes are shaded gray and correspond to the densest part of each data distribution

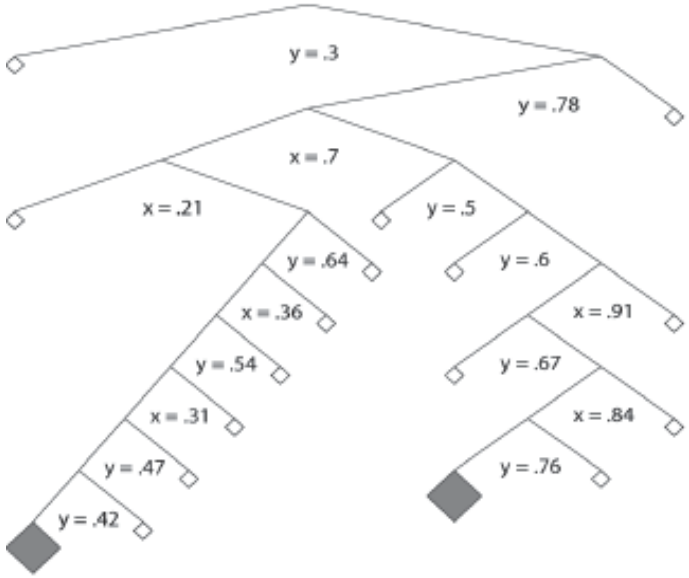

Figure 4. Tree representation of the partitions shown in Figure 3. The root node of the tree represents the entire state space. Each branching represents a split at the line given by the corresponding equation. The right branch represents the region to the right or above the split. Unfilled boxes on leaf nodes show regions that are not clusters. The two gray filled boxes show the clusters

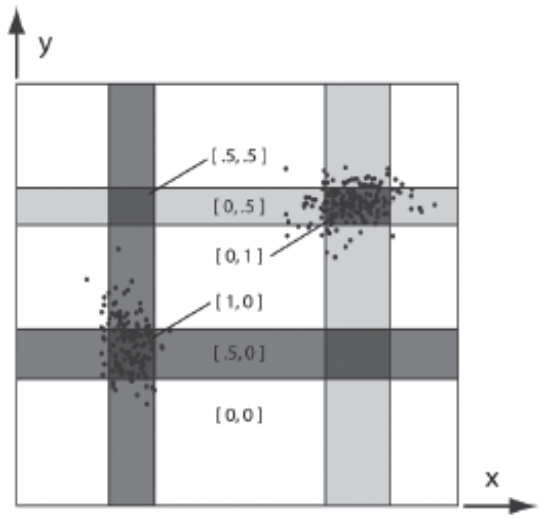

Figure 5. Mapping from state space *S* to feature space *S'*
Values of *s'* are given for each region in *s*, by color

3.2 Simulation 2: Vision

In the second simulation, kx-trees were applied to grayscale images to find simple visual features. The simulation was designed to be similar to the human visual processing system. The images were taken from the CalTech-256 image database [9]. Images were randomly selected, Gaussian blurred with a pixel radius of 6, then downsampled by a factor of 4 (such that the resulting image had 1/16 the number of pixels). This was done to remove the small-scale artifacts of JPEG compression. Then, images were convolved with a center-surround difference filter in order to mimic the retinal and thalamic processing that results in center-surround receptive fields for neurons entering layer 4 of cortical area V1 [12]. Each pixel value in the resulting center-difference image was obtained by taking the original pixel value and subtracting a fraction of the value of each of its neighbors: one-sixth for the four pixels with which it shares a border and one-twelfth for its four diagonal neighbors. The resulting difference value had the range of [-1, 1]. Three-by-three pixel groups were taken in a systematic raster tiling from the difference image and processed using a kx-tree. After the image had been completely scanned, a new image was selected and the process repeated. The nine-dimensional state space was partitioned and clusters were identified. The defining parameters for the kx-tree were chosen as follows:

Anomaly lifetime,	$L = 1000$
Maximum number of members,	$M = 70$
Number of split candidates per dim.,	$N = 1$
Minimum category size,	$\in = 1.0$

Choosing $N = 1$ ensured that all splits would be perfect bisections, and choosing $\in = 1.0$ ensured that no dimension would be split more than once. The resulting partition had a maximum of $2^9(512)$ distinct regions.

The result of running simulation 2 for 200,000 observations was a set of clusters in the nine-dimensional state space, shown in the top panel of Figure 6. As shown in the bottom panel, these can be notionally compared to the receptive fields identified in the human visual cortex, which have exhibited strong characteristics of directionality [12]. While the correspondence is too loose to be conclusive, it suggests that kx-trees may produce feature representations that are similar to those produced by the human neocortex.

A difference image can be reconstructed from the feature space representation by breaking it into 9-pixel tiles, transforming each tile from *S* to *S'*, and then transforming them back to *S*. The forward transformation given in Equation 9 was used here. The inverse transformation from *S'* to *S* was achieved by finding the centroid of each cluster hyperbox, weighting it by the value of that dimension in *s'* , and linearly summing the results for all clusters. The results of the inverse transformation illustrate which elements are captured in the feature representation. A richer feature set will result in a higherfidelity reconstructed image. The results of three such reconstructions are shown in Figure 7.

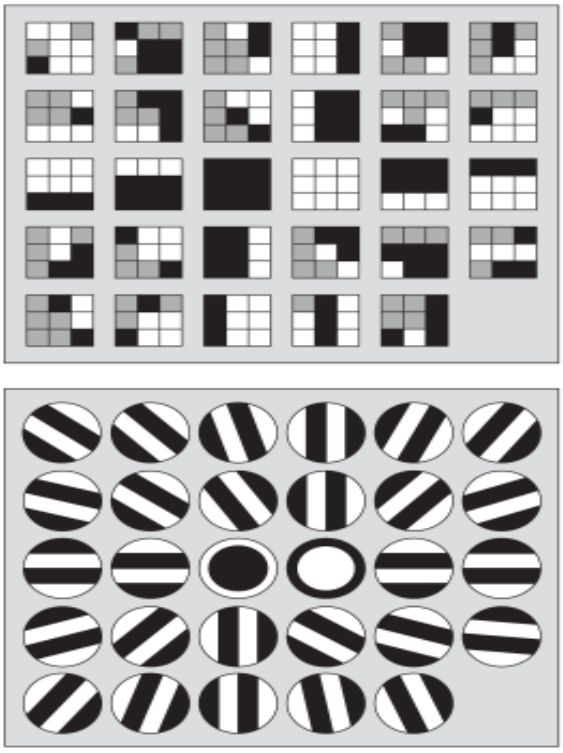

Figure 6. Top: Graphical representation of the 29 clusters found in the space of 9 pixel groups. In each cluster, dimensions (pixels) that have been bisected are represented as white or black, depending on whether the cluster falls above or below the bisection. Pixels in gray have not been bisected. Bottom: Strongly oriented stimuli, matched to each cluster above. Most of the clusters have unbisected pixels, and hence ambiguous receptive fields. While not unique, these stimuli would each be a match for their corresponding cluster in the top panel

4. Discussion

Concept acquisition can extend the capabilities of dialog agents, making them more useful and more human-like in their dialog generation. In this work, concepts are defined as clusters of lower level observations. While this definition does not satisfy all theories of what a concept is, it does provide a useful basis for beginning an investigation of the topic. Formulated this way, the problem of concept acquisition becomes a clustering or unsupervised learning problem. Acquired concepts can be high level and very abstract, or low level and quite specific. In either case, clusters of state observations are identified and used to represent the underlying states.

4.1 Getting from sensory experiences to useful concepts

The primitive visual concepts created in the second simulation are not rich enough to aid a dialog agent. They are only an illustration of the first step of an iterative process which, if carried out to sufficient depth, may be able to create concepts useful in dialog. For example, the concept clusters identified in the simulation are reminiscent of line segments and the empirically demonstrated receptive fields of neurons in the primary visual processing area of the cerebral cortex, V1. Applying a kx-tree to these concept elements may lead to higher level concepts, spanning greater portions of the visual space, corresponding to edges, corners, and curves. Elements of this type have been shown to be receptive fields for neurons in cortical area V2. Repeating the process may lead to complete shapes, and then to objects, which neurons in V4 and higher cortical vision areas have been shown to respond to. Concepts corresponding to objects are sufficiently advanced to be useful in associating with and interpreting dialog.

Although the preceding example focused on vision, it could have applied equally well to multi-modal associations. kx-trees are agnostic to the physical phenomena and semantic content represented by the sensory information and concepts that they cluster. Instead of clustering only visual input, the kx-tree might have clustered an image of a cockatoo with an audio

recording of its song and the ASCII string “bird.” Multi-modal clusters of this nature provide a straight-forward mechanism for a dialog agent to parse its input in terms of the concepts it represents and to make the appropriate semantic connections.

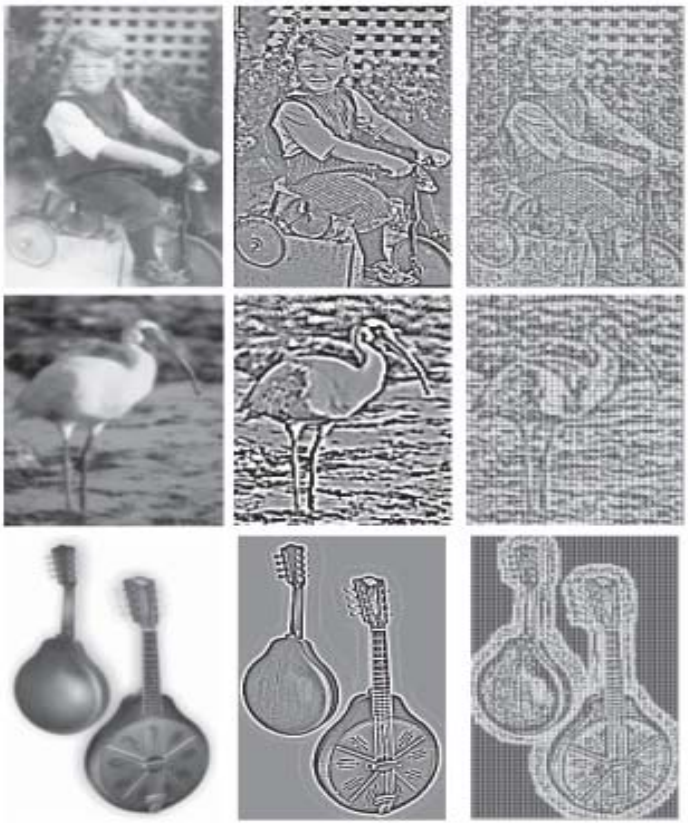

Figure 7. Original grayscale images (left, taken from [9]), difference images (center), and reconstructed difference images (right), based on the features from Figure 6. The difference images show the derivative of the gradient in the images and mimic the information retained by the center-surround processing performed by the human visual system. The reconstructed images retain the larger-scale features from the difference images

4.2 Practical application of kx-trees to dialog agents

Concept acquisition is only a part of a fully functioning dialog agent. In addition to kx-trees, a dialog agent would need 1) a mechanism for recording observed dialog patterns and their outcomes and 2) a mechanism for selecting among candidate utterances when forming responses. An architecture containing these elements was developed for another application: control of robot movement. [22] In that work, kx-trees generated concepts, sequences of which were recorded in a world model. A planner used those recorded sequences of concepts to predict likely outcomes to various actions, and based on those predictions selected which actions to take. In order to take full advantage of the concepts created by kx-trees, a dialog agent would have to be constructed on similar principles. Common practices of specifying patterns that utterances may fall into, such as <subject>-<verb>-<object>, presuppose the existence and definition of certain concepts. When all concepts are generated during the agent’s operation, preprogrammed patterns of this type cannot be used. This puts the dialog agent at the disadvantage of having to learn all such useful patterns, but also at the great advantage of not being limited to the patterns that its creator saw fit to include.

A dialog agent using kx-trees and learning useful patterns through experience may be able to solve some of the more challenging problems of dialog interpretation and generation. For instance, implicatures, including rhetorical expressions and expressives, are problematic to interpret since the intended meaning is not contained in the denotative semantics of the utterance. Contextual information, such asthe speaker’s tone, the speaker’s facial expressions, and recent events may all provide critical clues to the intended meaning. The difficulty of this problem is illustrated by the fact that some human listeners are unable to interpret some implicatures, particularly those containing irony or sarcasm, even while having access

to all the relevant contextual data. Due to the fact that a kx-tree based dialog agent cannot be given templates with which to interpret dialog, it is forced to learn appropriate interpretations through experience. This process would be similar to that used by a human, who, after being reassured by a friend, "I'm fine," later discovered that the friend was disappointed and angry. Experience would teach this individual to attach some semantic significance to the phrase "I'm fine" other than that which it suggests on its face. Similarly, the dialog agent may be able to learn to correctly interpret implicatures of all types.

The ability of a dialog agent using kx-trees to learn semantics from experience is dependent on their ability to integrate sensory information from a wide variety of sources and modalities. As described earlier, this is the way that kx-trees generate concepts, and so would be inherent in any agent employing kx-trees. Concepts are built from low level data and are only given labels when the experience of the agent (through hearing or reading) learns to associate one. It should be noted that the concepts can be useful to the agent before they are labeled. During concept acquisition, non-verbal cues such as facial expressions, breathing patterns, and body language could also be identified as useful modes of communication. They could even be used by the agent, if it were given a sufficiently capable hardware embodiment. In this way, a dialog agent using kx-trees may provide a unified approach to automatically learning and using both verbal and non-verbal communication.

4.3 Implications of set-based concepts

There are several notions from cognitive psychology and philosophy of what constitutes a concept. The definition of a concept that is used here is operational and distinct from these. For the purposes of this work, a concept is a set of related sensory experiences. For instance, a set of images depicting dogs can be treated collectively as the concept <dog>. Concepts can also be composed of other concepts. For instance, <dog>, <cat>, <hamster>, and <goldfish> can all be grouped together to form the higher level concept <pet>.

The notion of concepts as sets sensory experiences and other concepts has several important implications. First, a concept may comprise heterogenous elements. For example, the concept <dog> may include images of dogs, images of the printed word "dog", audio recordings of dogs barking, audio of the spoken word "dog", the tactile sensations of fur and of being licked, and the smell of dog in need of a bath. For dialog agents, a concept of this type provides connections between the word and the various sensory associations that humans might make with it.

Second, a concept may be semantically ambiguous. The concept <bark> for instance, in addition to including the written and spoken word "bark", could include both audio of dogs barking and thetactile sensations arising from touching the covering of a tree. For dialog agents, this property of concepts allows them to maintain their semantic ambiguity, which may be necessary in order to correctly interpret some dialog, such as puns and riddles.

Third, a concept or sensory experience may be a member of multiple concepts. For example, the concept <dog> may be a member of the higher level concepts <animal>, <pet>, <security system>,and <friend>. The implication of this structure for dialog agents is that reference to one concept (<dog>) can reference all its related senses. This is in contrast to a strictly structured concept map, such as WordNet [25] where each concept is the child of exactly one parent concept.

4.4 Concept acquisition vs. symbol grounding

Symbol grounding is the problem of assigning concrete sensory patterns to abstract categories. [10] Concept acquisition and symbol grounding are both approaches to associating sensory experiences with more abstract concepts. The difference between them is that concept acquisition builds concepts from data while symbol grounding begins with a number of concepts and relates them to data. Concept acquisition works from the bottom up and symbol grounding from the top down. Top down efforts to express concepts in terms of raw data began in the early days of applying artificial intelligence (AI). AI tools that performed symbolic, categorical, and logical manipulations to achieve their results could only be applied to physical problems if those problems could be mapped into the symbolic domain in which the tool was operating. Because the tool had been designed in the symbolic domain, the symbols that needed grounding had also been chosen by the designer. These systems were not capable of generating novel symbols or concepts based on the inputs they received. Their representational capacity was inherently limited.

Bottom up concept creation suggests a different approach to designing reasoning agents. If an AI tool or agent begins operation without a dictionary of symbols, then it also cannot have preprogrammed rules for manipulating or computing with those symbols. For example, if a dialog agent is created without parts of speech defined, then it also cannot have templates for how to combine parts of speech into sentences. An agent of this type must also include the capacity to acquire rules and

behaviors. And in cases where a shared symbol dictionary is desired (as is the case when we wish to have a dialog agent and a human agent "mean" the same thing when they use the same word) agents must also include a mechanism for negotiating the semantic content of symbols with other agents. When a dialog agent is capable of 1) creating new concepts from experience, 2) acquiring new behaviors, and 3) negotiating the meaning of words through interaction with human teachers, it lacks the limitations inherent in conceptual top down dialog agents. There is no obvious limit to the sophistication of interpretation and expression of which a bottom up dialog agent might be capable.

4.5 Relation to other work

A decision tree is a natural representation for the iterative division of categories. Each leaf represents a terminal category. Each node represents both the super-category formed by combining its two children, as well as the decision boundary used to differentiate between them. (See Figure 1) Several tree-based adaptive partitioning algorithms have been previously proposed. The Parti-Game algorithm [17] uses a greedy controller to crawl through a partitioned state space. When the controller fails, the last visited subspace is divided. Parti-Game is specifically geared to geometric path planning; it assumes that all paths through the state space are continuous. The G Algorithm [4], U-Tree [16], Continuous U-Tree [26], AMPS [14], and decision trees of Pyeatt and Howe [18] are all approaches used in conjunction with dynamic programming methods or the popular temporal difference method, Q-learning [28], to estimate the value function across the state space. Whenever a subspace's value estimate is shown to be inadequate, the subspace is divided. The G Algorithm handles binary data, U-Trees handle discrete data, and Continuous U-Trees, AMPS, and Pyeatt and Howe's approach handle continuous data.

Taking a broader view, kx-trees are a member of the set of unsu- pervised learning methods, that is, methods that group data that is unlabeled and unclassified. Also referred to as clustering algorithms, there are many unsupervised learning methods developed with different sets of assumptions, but kx-trees provide a novel collection of characteristics. kx-trees are an on-line, hierarchical, divisive clustering algorithm. They are stable in the sense that cluster boundaries do not move. kx-trees are most notable for what they don't assume. Unlike many clustering algorithms, kx-trees operate without assuming 1) how many categories exist in the underlying data, 2) that prior probabilities of any category are known, 3) that prior probabilities are stationary, and 4) that the forms of the class conditional probabilities are known. Once the dimensionality and range of the state space is defined, kx-trees always start from the same initial conditions, so there is no need to carefully pick initial values for cluster parameters. Only the four operating parameters L, M, N, and $\tilde{o}$ need to be selected. One of kx-trees' greatest strengths is that it does not assume that the input space is well scaled or even linear. In fact, kx-trees' distance metric, the modified Hamming distance, does not even assume that each of the input dimensions is an ordered field. As a result, kx-trees can handle inputs consisting of enumerated types, such as *{*poodle = 1, newfoundland = 2, beagle = 3, chihuahua = 4*}*. And their computational demands are modest.

kx-trees' strength comes at a cost, of course. They bound clusters with hyperboxes, so clusters that are not well fit by a hyper-box may not be concisely represented. If M is large, kx-trees can require a relatively large amount of data before useful clusters are generated. They are unstable in the sense that the total number of clusters created can, under certain conditions, grow without bound if not artificially limited. Like other unsupervised learning methods in which the number of clusters is not specified, the validity of the clusters found depends entirely on the appropriateness of the measure of cluster goodness (the node-splitting criterion in the case of kx-trees). And, like other unsupervised learning algorithms of its class, there are no theoretical performance guarantees.

4.6 Relation to deep learning

The problem of concept acquisition in dialog agents is closely related to the problem known in the machine learning community as feature extraction or *deep learning*. [1] Deep learning approaches seek to discover and exploit the underlying structure of a world by creating higher level, lower-dimensional representations of the system's input space. Deep learning algorithms include Convolutional Neural Networks (CNN) [15], Deep Belief Networks (DBN) [11], and the Deep SpatioTemporal Inference Network (DeSTIN) [13]. Deep learning algorithms such as these are alternative approaches, worthy of consideration for automatic concept acquisition, although they differ somewhat from kx-trees. CNNs are designed to work with two-dimensional data, such as images, and they do not apply to arbitrarily structured data, as kx-trees do. By using several layers of Restricted Boltzmann Machines, DBNs are capable of generating sophisticated features that allow it to interpret novel inputs. However, they are typically applied to the supervised learning problem of discrimination, and require a substantial amount of labeled data in order to be adequately trained. Whether DBNs can be applied to the unsupervised learning problem of concept acquisition is unclear. DeSTIN incorporates both unsupervised and supervised learning methods and appears to be fully capable of concept acquisition. It has been published only recently; future papers describing its operation and performance will allow a more detailed comparison with kx-trees.

4.7 Relation to other biologically motivated methods

The performance of kx-trees on identifying visual features similar to the receptive fields of V1 is suggestive, but inconclusive. It indicates that further investigation of kx-trees as a biological learning and concept acquisition mechanism is warranted.

There are several other approaches to the problem of concept acquisition that are motivated by theories of human psychology and neuroscience. Hierarchical Temporal Memory (HTM) [7] uses a hierarchy of Bayesian classification elements to cluster data into concepts ("causes") and to infer likely causes for a given set of inputs. Adaptive Resonance Theory (ART) [3] uses neural networks to bound clusters of states within hyperboxes. Like kx-trees, both HTMs and ART are online methods, meaning that they can incorporate new state observations and update their results incrementally. However, HTMs and ART differ from kx-trees in that they use a Euclidean distance metric to determine the similarity between states. For Euclidean distance to be valid, the state space must be well scaled, that is, a unit step in each dimension must represent the same magnitude of change. This is a somewhat restrictive condition that kx-trees do not need to meet. Because kx-trees do not use Euclidean distances (or any *p*-norm), they do not require well-scaled state spaces.

5. Acknowledgements

This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories. Sandia National Laboratories is a multi program laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

[1] Arel, I., Rose, D. C., Karnowski, T. P. (2010). Deep machine learning— a new frontier in artificial intelligence research', *IEEE Computational Intelligence Magazine*.

[2] Breiman, L., Friedman, J. H.,.Olshen, R. A., Stone, C. J (1984). *Classification and Regression Trees*, Wadsworth International Group, Belmont, California.

[3] Carpenter , G., Grossberg, S. (1987). A massive parallel architecture for a self-organizing neural pattern recognition machine, *Computer Vision, Graphics, and Image Processing*, 37 (1) 54–115.

[4] Chapman, D., Kaelbling, L. P.' (1991). Input generalization in delayed reinforcement learning', *In: Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, p. 726–731.

[5] Devroye, L., Gy¨orfi, L., Lugosi, G (1996). *A Probabalistic Theory of Pattern Recognition*, number 31 in Applications of Mathematics: Stochastic Modeling and Applied Probability, Springer-Verlag, New York.

[6] Duda, R. O., Hart,P. E., Stork, D. G. (2001). *Pattern Classification*, Wiley Interscience, second edn..

[7] George, D., Hawkins, J.(2005).A hierarchical bayesian model of invariant pattern recognition in the visual cortex', *In Proceedings of the IEEE Intl Joint Conference on Neural Networks*, 3, p. 1812–1817, (31 July-4 Aug 2005).

[8] Gilbert, N. ,den Besten, M., Bontovics, A., Craenen, B. G.., W. Divina, F., Eiben, A. E., Griffioen, R.ev´1zi, G. H´ˆorincz, A. L Paechter, B., Schuster, S., Schut, M. C., Tzolov, C., Vogt, P.,Yang, L. (2006). Emerging artificial societies through learning', *Journal of Artificial Societies and Social Simulation*, 9 (2).

[9] Griffin, G., Holub, A., Perona, P.(2007). Caltech-256 object category dataset', Technical Report 7694, California Institute of Technology, (2007). http://authors.library.caltech.edu/7694.

[10] Harnad, S. (1990).The symbol grounding problem', *Physica D*, 42, 335–346.

[11] Hinton, G. E. Osindero, S., Teh, Y. (2006).A fast learning algorithm for deep belief nets, *Neural Computation*, 18, 1527.

[12] Hubel , D. H., Wiesel, T. N (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex', *Journal of Physiology*, 160, 106–154.

[13] Rose, D. , Arel, I., Coop, B. (2009). Destin: A deep learning architecture with application to high-dimensional robust pattern recognition', in *Proc. AAAI Workshop Biologically Inspired Cognitive Architectures (BICA)*..

[14] Kochenderfer,M. J. (2006). *Adaptive Modelling and Planning for Learning Intelligent Behaviour*, Ph.D. dissertation, University of Edinburgh, School of Informatics.

[15] LeCun, Y.,Bottou, L., Bengio, Y., Haffner, P (1998). Gradient-based learning applied to document recognition, *In: Proc. IEEE,* 86 (11) 2278–2324.

[16] McCallum, A. K. (1995). *Reinforcement learning with selective perception and hidden state,* Ph.D. dissertation, University of Rochester, Computer Science Department.

[17] Moore, A.W., Atkeson, C. G. (1995).The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces, *Machine Learning*, 21, 199–233.

[18] Pyeatt, L. D., Howe, A. E. (2001). Decision tree function approximation in reinforcement learning, *In*: *Proceedings of the Third International Symposium on Adaptive Systems: Evolutionary Computationand Probabalistic Graphical Models*, p. 70–77.

[19] Quinlan, J. R.(1993). *C4.5: Programs for Machine Learning*, Morgan Kaufmann.

[20] Rohrer, B. (2010).Concept acquisition for dialog agents, *In: Symposium on Linguistic and Cognitive Approaches to Dialog Agents, Convention of the Society for the Study of Artificial Intelligence and Simulation of Behaviour*.

[21] Rohrer, B. (2010). kx-trees: An unsupervised learning method for use in developmental agents, *In*: *9th International Conference on Development and Learning*.

[22] Rohrer, B., Morrow, J.D., Xavier, P., Rothganger, F.(2010). Becca: A bica for arbitrary robots in unknown worlds, *In: First International Conference on Biologcally-Inspired Cognitive Architectures*.

[23] Rohrer, B., Morrow, J.D., Rothganger, F., Xavier, P.G.(2009).Concepts from data, *In*: *Proceedings of the Brain-Inspired Cognitive Architectures Symposium, AAAI Fall Symposium Series*.

[24] Rokach, L., Maimon, O. (2008). *Data Mining with Decision Trees: Theory and Applications*, volume 69 of *Machine perception and artificial intelligence*, World Scientific Publishing Co., Singapore.

[25] Princeton University (2010). About wordnet', *WordNet*, (2010).http://wordnet.princeton.edu.

[26] Uther,W. T. B., Veloso, M. M.(1998).Tree based discretization for continuous state space reinforcement learning, *In*: *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-98)*, Madison, WI.

[27] Uther,W. T. B., Veloso, M. M. (2003). *Adaptive Agents in Multi-Agent Systems: Adaptation and Multi-Agent Learning*, volume 2636 of *Lecture Notes in Computer Science*, chapter T Tree: Tree-based state generalization with temporally abstract actions, 266–296.

[28] Watkins,C. J. C. H., Dayan, P. (1992). Technical note: Q-learning', *Machine Learning*, 8 (3-4) 279–292.