Learning combination of evidence functions in object recognition

Diane J. Cook[*] Kevin Woods, Lawrence O. Hall, Louise Stark[†] Kevin Bowyer
Department of Computer Science and Engineering
University of South Florida
Tampa, FL 33620
hall@csee.usf.edu, kwb@csee.usf.edu

Abstract

This paper describes a learning system, OMLET, which helps to automate the construction of function-based object recognition systems. OMLET is designed to learn fuzzy membership functions which approximate the allowable ranges of physical dimensions such as width, area, relative orientation, etc. The learning is done by iterative error reduction on a set of labeled training examples. Results from a chair domain show that the system is capable of learning the fuzzy membership functions.

1 Introduction

Function-based object recognition systems classify objects based upon their possible function rather than by matching against a set of known specific object models. For example, a function-based recognition system would recognize any object that satisfies the functional requirements "provides sittable surface" and "provides stable support" as being a chair. The potential advantages of function-based recognition systems have been discussed by a number of researchers in AI and computer vision [1, 3, 4, 5, 6, 9, 10]. The OMLET learning system is designed to facilitate the construction of such systems by allowing some of the many parameter values that are normally hand-crafted by trial and error to be automatically learned from a set of examples.

The current version of OMLET is designed to learn the approximate ranges for the physical measurements that go into the definitions of the functional requirements. For example, the width, depth, height and area (functional properties) of the "sittable surface" (a functional requirement) of a chair must each lie within some specified range. Different equally good chairs may have somewhat different values for each of these measurements. But if the area and width are increased enough, then the goodness of the object as an example of chair will decline and eventually it will be a better example of the category bench than it is of the category chair. This transition from chair to bench is fuzzy. The goal of OMLET is to learn an appropriate fuzzy membership function for the measurements of functional properties that are used to define the functional requirements. The membership function needs to both accurately describe the labeled training examples and provide acceptable generalizations to unseen examples.

OMLET is designed to work in concert with GRUFF [8, 7], a function-based computer vision system. Input to GRUFF is expected in the form of rules which constitute a complete boundary surface description of a valid 3-D object. The system is asked to attempt to categorize the object as belonging to a subcategory, with a cumulative ***association measure***. GRUFF uses the fuzzy probabilistic t-norm, t-conorm pair [2] in combining evidence. Output includes recognition of the (sub)category of which the object is believed to be a member (if it is of a known kind), identification of the portions of the object structure that fulfill the required functions, and a measure of the overall goodness for the fulfillment of the functions required by the object (sub)category. Given actual range measurements and examples that match GRUFF's, OMLET will learn the appropriate fuzzy membership functions for each of the parameter ranges. We show results from a domain of chairs [8]. The advantage of this system is that it will allow function based computer vision systems to be developed without the time intensive trial and error process of actually determining what the fuzzy membership values for the physical ranges need to be.

2 Omlet

OMLET is a learning system for ranges of function-based object recognition systems, and in the test mode it acts as a function-based recognition system. Knowledge primitives form the building blocks of the OMLET system and represent chunks of parameterized procedural knowledge that implement some basic concept about shape, physics, or causation. Each knowledge primitive takes some (specified portions of a) 3D shape description as its input, along with values of parameters for the primitive, and returns an ***evaluation mea-***

[*]Currently with Dept. of Computer Science and Engineering, Univ. of Texas at Arlington, Arlington, TX

[†]Currently with: Dept. of CS, Univ. of the Pacific, Stockton, CA

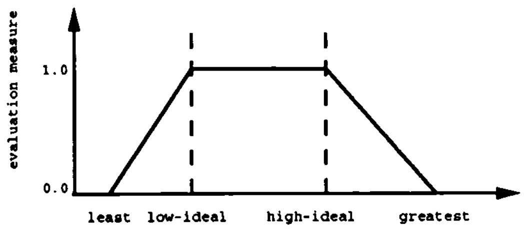
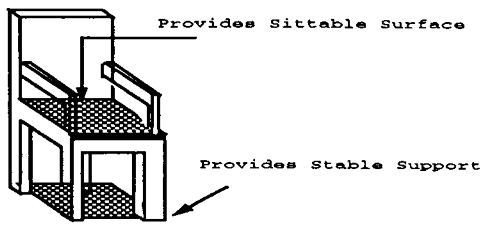


Figure 1: Fuzzy membership function defined by range parameters



```
(<- ( conventional_chair ?b ?c ?e ) 0.4 PAND
      ( provides_sittable_surface ?b ?c ?e )
      ( provides_stable_support ?b ) )

 (<- ( provides_sittable_surface ?b ?c ?e ) 0.4 PAND
      (dimensions AREA 0.135 0.22  ?c )
      ( WIDTH/DEPTH 1.0 ?c )
      ( dimensions CONTIGUOUS_SURFACE 1.0 1.0 ?c )
      ( dimensions HEIGHT 0.4 0.6 ?c )
      ( clearance ABOVE ?b ?c )
      ( clearance IN_FRONT ?b ?e ) )
 (<- ( provides_stable_support ?b ) 0.4 PAND
     ( stability SELF ?b ) ) )
```

Figure 2: Functional definition of a conventional chair

sure between zero and one. The evaluation represents a degree of membership in this category, or degree to which the shape satisfies the particular invocation of the primitive. Some of the knowledge primitives include range parameters which may be specified upon invocation: least, low-ideal, high-ideal, and greatest. These parameters are used to define a fuzzy membership function, as in Figure 1, for calculating the *evaluation measure* for the invocation of the primitive. Although the definition and the values representing low-ideal and high-ideal for each range may be supplied to the system, the arms of the trapezoid (the least and greatest points) are not known *a priori*. Let $z1$ represent the *least* value, $n1$ represent the *low-ideal* value, $n2$ represent the *high-ideal* value, and $z2$ represent the *greatest* value.

Rules are used as the representation language in the OMLET system. The rules indicate how the functional properties are combined to define a functional requirement, and similarly how the functional requirements are combined to give the function-based definition of an object category. An example of a function-based definition is shown in Figure 2. Given a set of examples, OMLET will attempt to build a general proof tree that matches the rules. The proof tree is a structure that mimics the way evidence is combined using the rule base, and facilitates the implementation of the learning algorithm. The physical measurements of the functional properties of an example object are input to the fuzzy membership functions in the leaves of the proof tree. The output at a leaf node is the evaluation measure for the individual functional property. The evaluation measures are incrementally combined during the evaluation of a shape, using the *probabilistic and* ($a \times b$) and *probabilistic or* combiners. The resulting combined measure is output at the root node of the proof tree as the membership value of the input example for the given object category.

The input to OMLET is a set of goals for specific labeled examples from some object category (such as "chair"). The input may include goals for examples from object subcategories as well (such as "arm chair"). The goal includes the example's category (or subcategory), the physical elements of the 3-D object that fulfill the functional requirements, and an evaluation measure.

Several approaches to learning the parameters of the fuzzy membership functions have been examined. For all the approaches, an example object pattern p is presented to OMLET with the expected output membership value. Next, OMLET attempts to prove that the object is known (a chair for our purpose). If the chair description matches the function-based definition, the output measure is calculated using the *probabilistic and* and *probabilistic or* combining functions. If the output membership value is sufficiently different from the expected output value, then the ranges that were included in the definition need to be adjusted. It is the manner in which the ranges are adjusted that distinguishes the three different learning approaches discussed here.

The error for a training example is defined as the difference between the expected output measure and the actual output measure. The error is propagated down the function-based proof tree with each element at an and or or junction getting an equal share of the error. For example, if the error at a three element and node is d, then each of the three elements will receive a portion of the error equal to the cube root of d (i.e. the inverse of the *probabilistic and* function). Similarly, the inverse of the *probabilistic or* function is used to

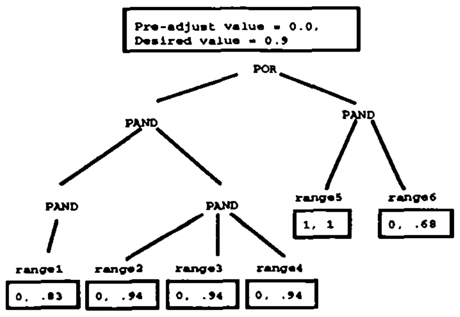


Figure 3: A proof tree

propagate the error value at an or node. It should be noted that the desired evaluation measure is actually propagated in the OMLET implementation, however the error is directly computable since the actual and target values are always known at each node in the proof tree. Eventually, a portion of the overall error reaches the ranges defined by the fuzzy membership functions. An example case of a function definition that combines six range's values using four probabilistic **ands** and one probabilistic **or** is shown in Figure 3. In this figure, the membership values before and after adjustment are shown for the entire object and for each range.

When the error reaches the individual ranges for a training example, the input to the fuzzy membership function (i.e. the x axis value) and the desired evaluation measure can be used to make adjustments to the membership trapezoids in an attempt to reduce the error. Two methods for adjusting the arms of membership trapezoids are described in the following subsections. Initially, it is assumed that the normal ranges ($n1$, $n2$) are prespecified and not permitted to change during training. The last subsection describes some methods for learning the normal ranges from the training data.

2.1 Adjusting After Each Epoch

Since only the arms of the trapezoid are permitted to change, the $z1$ and $z2$ values of the membership trapezoids are adjusted to reduce the error amount. This means that if the object's membership value is too low, a range, such as range1 in Figure 3, will have one of the arms of the membership function adjusted. The z1 value will become smaller or the z2 value will become greater to give the the observed range value a greater membership value. The user can define a *learning rate* that controls how quickly the ranges will adjust to fit

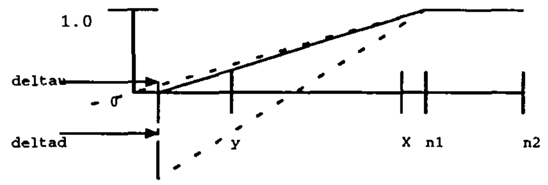


Figure 4: The biased adjustment problem

a given training example. This procedure adjusts all of the trapezoids to locally minimize the mean-squared error over all training examples. That is, the trapezoidal membership functions are adjusted after each training example is presented. The training set may be used to continually adjust the ranges until a satisfactory level of accuracy is achieved.

Error is assigned equally to each range on a level. The ranges are adjusted after all the training examples have been presented, i.e. at the end of a training epoch. Making global changes may well allow smoother error reduction and allow the best adjustment on the average. When an error is found for a training example and propagated back down the proof tree to the appropriate ranges, the x value and desired y value for each range is stored in a list of "target" points. Additionally, the arm (right or left) to which the target point belongs is noted. At the end of a training epoch, an average x value and an average desired y value are computed for each arm of the trapezoid, and a single adjustment is made to each trapezoid arm.

There are a couple of issues that arise in adjusting the arms. First, examples with a large membership value (close to 1.0) can have a profoundly larger affect on the trapezoid adjustment than examples with smaller membership values. To illustrate this point, consider Figure 4, which is approximate. x has a high membership value of 0.97, while y has a small membership value of 0.12. At x we would like to decrease its membership value by 0.001, while at y we would like to increase its membership value by 0.001. Decreasing the membership value for x will cause a large shift of $z1$ to the right, while increasing the membership value for y will cause only a small shift of $z1$ to the left. This behavior heavily biases the system toward examples with large membership values. To compensate for this bias, we multiply each example's error by an amount inversely proportional to its membership value. In this way we lessen the bias of examples with large membership values and increase the bias of examples with small membership values.

Second, because OMLET is not capable of determining which trapezoids are to blame for generating an incorrect measure, some trapezoids are adjusted when they should not be. There are cases in which the system can detect a wrong adjustment. If one training example has an expected measure greater than 0, then we know that all of its range values lie between $z1$ and $z2$. Therefore, OMLET should never change the arms of a trapezoid to exclude these examples which have already been presented to the system. To ensure that the arms of a trapezoid never exclude positive examples, OMLET keeps track of the farthest outlying points which are used in examples with measure greater than 0. If a later training example causes the system to bring $z1$ or $z2$ closer to $n1$ or $n2$, OMLET will make sure that they are not moved in beyond the farthest point to be included. With these adjustments, the overall error for OMLET has been significantly reduced for fairly large datasets.

2.2 Least Squares Line Fit

The second variation of learning the membership trapezoid arms uses the list of target points to adjust the ranges after each training epoch. A least squares line fit is used to fit each arm of the trapezoid to the set of target points. Since this method attempts to make a direct estimation of the desired positions of the trapezoid arms rather than incrementally moving the arms in an effort to reduce the average error, the bias adjustment problem described above does not exist. As with the first method, better results are achieved if the trapezoid arms are prevented from being positioned such that positive training examples evaluate to zero.

2.3 Learning the Normal Ranges

If the normal ranges of the membership trapezoids are not known a priori, then OMLET is capable of learning them from the training examples. All membership functions are initialized with $z1 = min$, $n1$ and $n2 = 0.5$, and $z2 = max$, where min and max are the minimum and maximum inputs to the membership functions, respectively. Thus, membership functions are trapezoids with a normal range that is initially a point. For the first training epoch, a list of target points for each range is collected by propagating the desired evaluation measure down to the individual ranges. Note that the fuzzy ranges are not used in any computations, and therefore the initialization is somewhat arbitrary. For each range, an approximate location of the normal range is made. This is done by moving the normal range (actually a point defined by $(n1 = n2, 1.0)$) incrementally from $z1$ to $z2$ and noting the error over

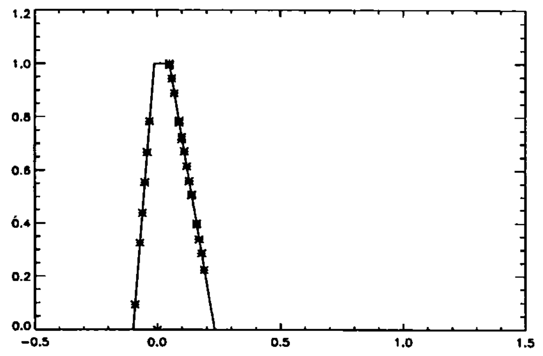


Figure 5: Using least square line fit and normal range learning to fit trapezoids to target points

the entire training set for each position. The position which produces the minimum error is chosen as an initial guess for the normal range. It is only important at this stage to place the point for the normal range somewhere within the actual normal range.

Some minor adjustments to the learning algorithm are made which will allow the normal ranges to change during training. Using the least squares line fit algorithm, new values for $n1$ and $n2$ can be estimated from where the new trapezoid arms are equal to 1. A couple of restrictions on new $n1$ and $n2$ values assure that the membership functions remain trapezoidal (or triangular if $n1 = n2$). Obviously, $n1$ must be less than or equal to $n2$. In order to make gradual adjustments to the membership functions, $n1$ and $n2$ will only be moved a small portion in the direction of the desired change, rather than jumping directly to the points where the trapezoid arms are equal to one. For some ranges, the training data only provides target points for one leg or a portion of a trapezoid. OMLET is capable of detecting this situation by observing the slope of the fitted line, and adjusting the membership function appropriately. Figure 5 is an example of a fuzzy membership function that OMLET approximated from the training examples.

3 Results

The OMLET rules for a conventional chair are shown in Figure 2. There are three fuzzy ranges involved. They are for AREA, CONTIGUOUS SURFACE, and HEIGHT. Presently we have 88 examples from the chair domain including straight back, arm, and con-

Training Method	Avg Err	Max Err
1. After each epoch	0.018	0.486
2. Lst Sqr Fit (LSF)	0.062	0.526
3. LSF learn normals	0.043	0.501

Table 1: Omlet Learning Results

ventional chairs. These examples have overall membership values from the GRUFF [7] system given the same ranges.

In order to test the ability of OMLET to generalize, a training set consisting of 44 chair examples is created, and the remaining 44 examples are used as a test set. Training is done for 500 epochs, using each of the three methods described above. In addition, results for the least squares line fit method are reported when the normal range is not prespecified, and must be learned by OMLET from the training examples. Since the overall error may oscillate, results are reported for the lowest total error found during training. Table 1 gives the average error per sample, and the maximum error for any sample for the three test situations described above.

In most cases all examples are recognized as chairs with approximately the membership value that was expected. Most membership values were within 0.05 of the expected. The large maximum error of approximately 0.5 is for the same sample for all learning methods. The next highest error value is approximately 0.2. These tests show that it is possible to learn the fuzzy membership functions for a real-world function-based object recognition domain.

4 Conclusion

The OMLET system described here is an effective early approach to providing a tool to help design function-based object recognition systems. It enables the fuzzy membership functions for physical dimensions to be learned from a set of labeled example objects. The objects need only be given consistent approximate membership values in the class to which they belong. The OMLET system can be used to speed the development of function-based object recognition systems by removing the necessity for trial and error determination of appropriate physical ranges.

Acknowledgements: This research was partially supported by a grant from the Whitaker Foundation, National Cancer Institute (CA59 425-01) by NSF grant IRI-9120895 and AFOSR grant F49620-92-J-0223.

References

[1] T. O. Binford. Survey of model-based image analysis systems. *International Journal of Robotics Research*, 1:18–64, 1982.

[2] P. Bonissone and K. Decker. Selecting uncertainty calculi and granularity: An experiment in trading-off precision and complexity. In L. Kanal and J. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pages 217–247. North-Holland Publishing Company, 1986.

[3] M. Brady, P. E. Agre, D. J. Braunegg, and J. H. Connell. The mechanics mate. In T. O'Shea, editor, *Advances in Artificial Intelligence*, pages 79–94. Elsevier Science Publishers, 1985.

[4] M. Brand. An eye for design: why, where and how to look for causal structure in visual scenes. In *SPIE #1825: Intelligent Robots and Computer Vision XI*, pages 180–188, 1992.

[5] M. DiManzo, E. Trucco, F. Giunchiglia, and F. Ricci. Fur: Understanding functional reasoning. *International Journal of Intelligent Systems*, 4:431–457, 1989.

[6] M. Minsky. *The Society of Mind*. Simon and Shuster, 1985.

[7] L. Stark and K. Bowyer. Achieving generalized object recognition through reasoning about association of function to structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13, 1991.

[8] L. Stark, L. Hall, and K. Bowyer. An investigation of methods of combining functional evidence for 3-d object recognition. *International Journal of Pattern Recognition and Artificial Intelligence*. To appear.

[9] L. M. Vaina and M. C. Jaulent. Object structure and action requirements: a compatibility model for functional recognition. *International Journal of Intelligent Mechanisms*, 6:313–336, 1991.

[10] P. H. Winston, T. O. Binford, B. Katz, and M. Lowry. Learning physical description from functional definitions, examples, and precedents. In *National Conference on Artificial Intelligence*, pages 433–439, 1983.