Density-Adaptive Learning and Forgetting

by

Marcos Salganicoff

University of Pennsylvania
Philadelphia, PA 19104-6228

December 1993

IRCS Report 93-50

In Proceedings of Tenth International Conference on Machine Learning, Amherst, MA, June 1993.

Density-Adaptive Learning and Forgetting

Marcos Salganicoff
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
sal@grip.cis.upenn.edu

Abstract

We describe a density-adaptive reinforcement learning and a density-adaptive forgetting algorithm. This learning algorithm uses hybrid k-$D/2^k$-trees to allow for a variable resolution partitioning and labelling of the input space. The density adaptive forgetting algorithm deletes observations from the learning set depending on whether subsequent evidence is available in a local region of the parameter space. The algorithms are demonstrated in a simulation for learning feasible robotic grasp approach directions and orientations and then adapting to subsequent mechanical failures in the gripper.

1 Introduction and Motivation

In many learning applications, it is often the case that the learner perceives a continuum of real-valued perceptual input attribute valuations, in which case the state-space is infinite, rather than a set of discrete-valued attributes. Additionally, it is often the case that reinforcement for a given instance (the outcome) is binary. For example, in a robotic domain, either an object is grasped successfully or dropped. Therefore, in that context, it may be more meaningful to learn the conditional probability of receiving the reinforcement value, rather than to estimate reinforcement as a real-valued quantity.

We describe a technique, *Density-Adaptive* Reinforcement Learning (DARLING), for identifying regions of a real-valued parameter space where the lower-bound probability of succeeding (receiving *immediate* reward) subject to a $1-\alpha$ confidence value is above some minimum probability required for the task.

Also, as noted previously by [18, 7, 11], an important assumption taken by many learning methods is that the concept (e.g. the environment and task) to be learned is stationary over time. By stationary, we mean that the true underlying process which maps exemplars to outcomes is unchanging. A non-stationary concept can be manifested in terms of time varying state-transition functions, or time-varying reward functions. This may, in turn, be due to gradual or sudden failures in sensors and actuators, as well as changes in the behavior of the external world.

In artificial neural-network approaches, weights are updated on-line and non-stationarity presents less of a problem, since the weight updating rules will eventually change weights so that they minimize prediction error on the most recent pool of exemplars. The problem is more acute in cases where a memory-based approach for learning is used and stored exemplars are used directly to form predictions, such as nearest-neighbor and tree based approaches. In the non-stationary case, the learning set will contain observations which are obsolete and will be significantly biased by the representative exemplars from obsolete concepts. Consider also, that often, artificial neural networks are trained on a *fixed* learning set which is repeatedly presented in random order, and therefore the same requirement to delete obsolete observations is present.

We describe a density adaptive *forgetting* technique to delete obsolete observations using exponential weight-decay based on a nearest neighbor criteria. The approach uses a decay coefficient to decrement an experience's weighting, similar to that of [8, 11, 9]. However, this coefficient is a function of the similarity of that given experience to subsequent experiences, rather than a fixed value. The weight of an exemplar is decayed and deleted when it goes below some minimum value, and is superseded by the newer observations that led to its deletion. This procedure can be used as a front-end to a variety of learning algorithms, the only prerequisite being that the input attribute space can support a distance metric. It can also be easily modified to keep the size of a learning set bounded if limited storage is available.

We demonstrate the utility of these learning and forgetting algorithms in simulation for the assessment of a robotic grasp's suitability to an object with a given parametric superellipsoid attribute description [19] and pose. However, the method can be applied to any situation where there are a fixed set of actions to evaluate, a reward and a real-valued input attribute space.

This work differs from previous efforts in learning for robotic grasping in terms of action and perceptual representation, as well as the learning methods employed. Dunn [4] employed a two dimensional polygonal representation, and a random search for successful grasps during learning, followed by a 2-D model matching during execution. Tan [20] employed a feature-based sonar depth representation and a cost sensitive extension of ID-3 with 3-D objects. Bennett [2] worked in robotic grasping of polygonal 2-D puzzle piece task using explanation-based learning and domain theories about uncertainty and grasping. Mel [10], Ritter [15] and Cooperstock have used $\Sigma - \Pi$ neural-networks, self-organizing feature maps and backpropagation, respectively, for learning visually-guided control of robot arms for grasping.

2 Learning Algorithm

First, we describe the learning algorithm, *Density Adaptive* reinforcement learning (DARLING) [16]. Here the term *Density* refers to the local density of observations in the attribute space, defined as the number of exemplars per unit volume of attribute space.

The DARLING algorithm takes inspiration from decision tree approaches embodied in Classification and Regression Trees [3] and ID-3 [14], along with the geometric learning approaches described by Omohundro [13]. The goal of the algorithm is to identify regions of the input attribute space having a lower-bound estimated probability, p_-, of succeeding (receiving reward) that is greater than some specified minimum probability p_{min} required for the task. The algorithm produces a classification tree (see Figure 1) with real-valued splits that approximates those regions. We desire that the tree approximate those regions of the parameter space with minimum over- and under-estimation.

2.1 Density-Adaptation

The algorithm first builds a *k-D*-tree [5] based on the distribution of the exemplars in the parameter space, *ignoring* the outcome labels of each exemplar. This step adaptively partitions the exemplar set into a set of bins each with a roughly uniform number of exemplars. Therefore, a *k-D*-tree takes the exemplar distribution in the attribute space, and partitions the input space such that the probability of a future observation landing in any one of the partition bins approaches equi-probability, assuming it is drawn from the same distribution as the learning set. The attribute space

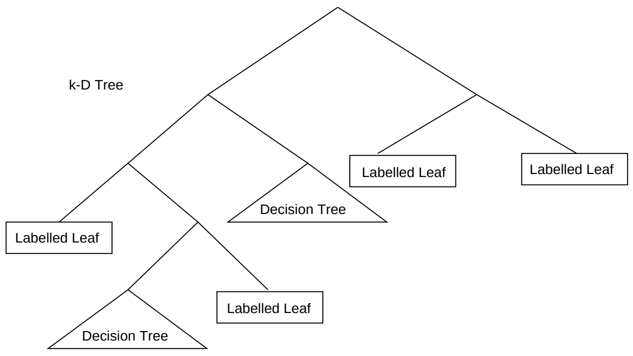

Figure 1: The DARLING Tree. It consists of a *k-D*-tree with leaf nodes that cover regions of the parameter space. The leaf nodes are either labelled classifications or the roots for shallow 2^k decision-trees that further partition and identify subregions.

is effectively transformed so as to equalize the original distribution of exemplars so that it is uniform [12]. The higher the local density of exemplars, the larger the number of bins per unit volume (area) of attribute space, and the smaller the average spatial extents of those leaves. Smaller spatial extent leads to higher effective resolution (see Figure 2).

Many decision-tree algorithms operating in real-valued domains are greedy, they rank the attributes and splits at the current node under construction based on some locally computable figure of merit. At one extreme are algorithms such as ID-3[14] and CART [3]. Their splitting criteria are based on the expected information gain among attributes which is purely a function of instance labelling and the orderings of these labellings as projected along the current attribute axis. They do not take into account metric information such as the physical locations of exemplars in the parameter space. The DARLING algorithm is at the other extreme since it first builds a *k-D* trees. The *k-D*-tree is also generated using a greedy algorithm for ranking attributes. However, it completely ignores the labelling of the exemplars, and picks attributes according to which attribute for the current node has the greatest spatial spread as projected onto the current attribute axis, only looking a metric information about the learning set. Immediately after this phase, the DARLING algorithm looks at outcome label information to build shallow decision trees.

2.2 Node Splitting and Labelling

As stated previously, *k-D*-tree partition does not ensure that the homogeniety of each bin is high, since its construction ignores the outcome of each exemplar, and is driven only by their locations. It may be that a node under evaluation does not meet the p_{min} stopping criteria. However upon splitting that node, it

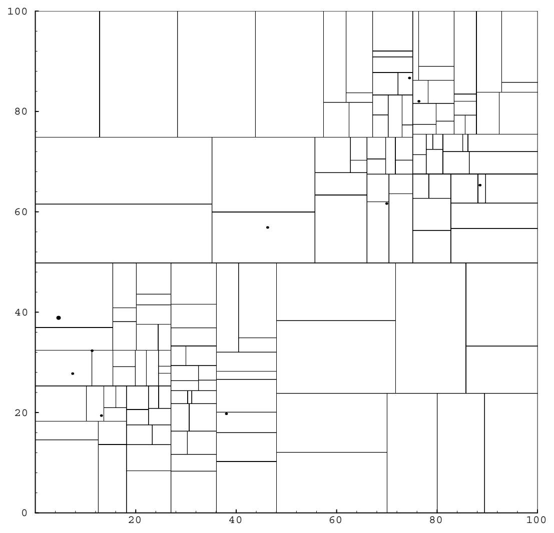

Figure 2: Density adaptation using *k-D* trees. The *k-D*-tree partition of a mixture of Gaussians and a uniform distribution. It can be seen that the size of the leaves decreases as the local density of samples increases, yielding the desired adaptive resolution property.

may yield a set of children nodes among which some are acceptable.

Since we are interested in finding regions of the attribute space having a pessimistic probability of success greater than some p_{min}, we use the outcome ratios to compute a probability interval estimate [6] for the underlying probability of receiving a success in that leaf. Two thresholds, p_{min} and p_{max}, are required for splitting. If the lower bound of the probability interval is above the p_{min} threshold, then it is accepted. It the upper bound p_+ is below the p_{max} value, then it is immediately rejected. If neither condition holds then the *k-D* tree leaf is further split.

This *k-D*-leaf splitting could also be accomplished using a variety of standard identification tree algorithms depending on the learning task. The critical notion is that the *effective* resolution of a decision tree of some depth l is greater when its domain is smaller. Because we are inserting these decision trees into the domain of *k-D*-tree leaves, the spatial distribution of exemplars governs the resolution of the individual decision trees. The decision trees may be shallow, yet still have high spatial resolution if they are embedded into a leaf with small spatial extent. In this implementation, we embed 2^k-trees (generalized quadtrees) into the *k-D*-tree. They are built to varying depths based the $(1-\alpha)$ acceptance criteria described below.

```
Algorithm **DARLING**(point_set)

    (* perform density adaptation *)
    Generate $k$-$D$-tree for point_set
    **for** all leaves of $k$-$D$-tree
    **begin**
        Compute $p_+$ and $p_-$ for current_leaf according to equation 2 (* can label as success *)
        **if** $p_- > p_{min}$ **then** cur_leaf.outcome := success
        **else if** $p_+ < p_{max}$ **then** cur_leaf.outcome := failure (* label as failure *)
        **else** generate_2k_tree( new( 2k_node ), cur_leaf ) (* build $2^k$-tree if indeterminate *)
    **end**
```

Figure 3: **The DARLING Algorithm**

2.3 Deciding When to Split

The idea of the splitting criteria is to drive the probability interval estimates towards the extremes of 0 or 1, which indicate good homogeneity in outcome of exemplars in the current leaf partitions. The upper and lower confidence bounds for the probability estimate interval, p_+ and p_- respectively, are computed using the same $1-\alpha$ confidence bound computation utilized by Kaelbling [6].

We desire an interval that contains the true probability value with confidence $(1-\alpha)$ given x successes (rewarding outcomes) out of n exemplars in the leaf. This yields

$$p_- \leq p_i \leq p_+ \qquad (1)$$

where

$$p_\pm = \frac{\frac{{g_{\frac{\alpha}{2}}}^2}{2n} + \frac{x}{n} \pm \frac{g_{\frac{\alpha}{2}}}{\sqrt{n}}\sqrt{\frac{{g_{\frac{\alpha}{2}}}^2}{4n} + \frac{x}{n^2}(1-\frac{x}{n})}}{1 + \frac{{g_{\frac{\alpha}{2}}}^2}{n}} \qquad (2)$$

where, p_- and p_+ are the lower and upper bounds of the probability interval estimate, respectively and $g_{\frac{\alpha}{2}}$ is the confidence interval coefficient (either tabulated or computed). The complete algorithm is given in Figures 3 and 4.

2.4 Robustness to Noise

The difference in error immunity between k-nearest-neighbor algorithms and the DARLING algorithm is shown in Figure 5. The quantity p_{flip} is the probability that the outcome of a given exemplar in the learning set is mislabelled by having its labelling flipped. The quantity p_{error} is an estimate the combined misclassification rate due to false positives and false negatives conditioned over all predicted and true members of the class. It is a normalized measure of hypothesis'

```
Algorithm generate_2k_tree( cur_node, leaf cur_leaf )

    Compute $p_+$ and $p_-$ for cur_leaf
    if $p_- > p_{min}$ then (* terminate *)
        begin
            cur_node.outcome := success
            cur_node.children := $\lambda$
        end
    else if $p_+ < p_{max}$ then (* terminate *)
        begin
            cur_node.outcome := failure
            cur_node.children := $\lambda$
        end
    else (* split further *)
    begin
        for all cur_node.children
            begin
                cur_node.child = new( node )
                generate_2k_tree( cur_node.child, partition(cur_leaf.child) )
            end
        end
```

Figure 4: **The Adaptive 2^k-tree Construction Algorithm**

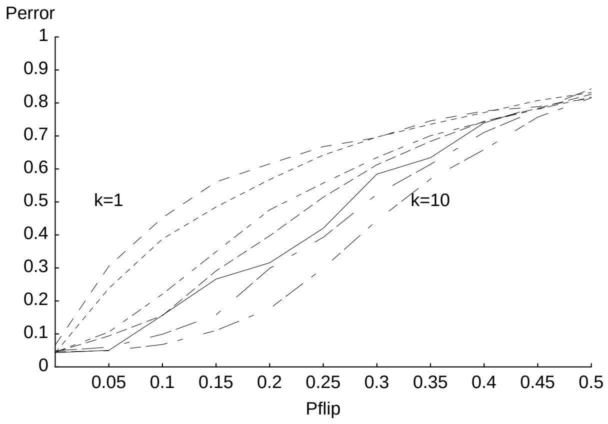

Figure 5: A comparison of several k-nearest-neighbor performance curves (dashed) to the DARLING curve (solid) for $\alpha = .5, p_{min} = .6$ and $p_{max} = .4$ and a unit circle test concept. The noise breakdown curve for the DARLING algorithm is similar to that of the 5-nn algorithm.

agreement with the true concept, computed by taking 1000 random exemplars uniformly distributed over the input domain.

The similarity in performance is due to the fact that the labelling of a given leaf in the DARLING tree is based on a probability estimate that is pooled from a number of observations over the leaf's domain, which is similar to the mechanism employed in k-nearest neighbors. In Figure 5 the bin size is b=10, which is approximates k=10 nearest-neighbors. However, some of the k-D-tree bins are split in to 2^k trees so they have smaller bin-size. This explains why the error break-down curve of DARLING falls somewhere in between that of the k=1 and k=10 cases for the nearest-neighbor learners.

3 Density-Adaptive Forgetting

As mentioned previously, an important distinction should be made in the taxonomy of learning systems between learning in a domain with stationary concepts versus learning in a domain where concepts may change. Additionally, we cannot expect the learner to have infinite storage capacity. This implies it must have some forgetting mechanism, in order to store only a bounded number of examples at any given time.

In particular, memory-based learning algorithms do not track changing concepts well, as noted by Moore [11]. Consider the following scenario: a learner has been collecting exemplars on line for several years and storing them (assume it has a huge store), and suddenly the dynamics of the environment change. In this case, the learner will adapt very slowly, since there will be a large number of observations from the past that are no longer representative of the current concept. In fact, the learner will have a permanent bias since the obsolete observations will always be in its database.

We develop a forgetting algorithm for accomplishing the deleting of experiences based on the principle of locality of observations. This states that observations should be forgotten *only* if there is subsequent information in their locality of parameter space.

This mechanism is implemented by associating a weight w to each observation. Each weight is decremented at a rate proportional to the number and proximity of succeeding exemplars to the corresponding observation. This is in contrast to other weighted forgetting mechanism where all weights are decremented by multiplication with factor, γ, between 0 and 1, each time a new observation is input to the system [11], independent of its location. These approaches have the disadvantage that the entire parameter space must be constantly refreshed, otherwise all data vanishes in regions not subsequently populated by exemplars. This creates an undue burden on acquiring new exemplars since, as the dimensionality of the input parameter space increases (and more exemplars are needed), the forgetting rate must be decreased which impairs tracking performance.

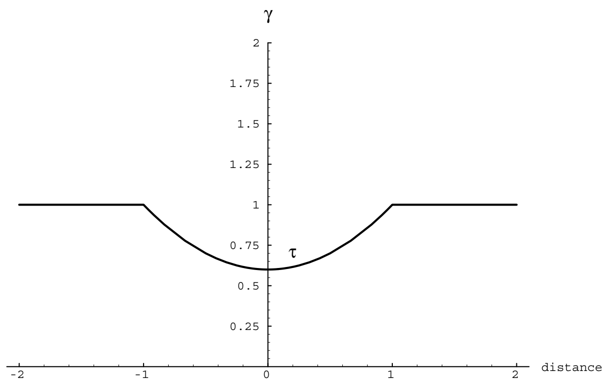

Figure 6: The influence function for decaying observation in the neighborhood of previous observations. The $\gamma = \gamma(X, X_{\{k\}})$, is used as a forgetting coefficient for the kth nearest neighbor which is at distance $d(X, X_{\{k\}})$ from the new observation X. It is a function of the scale parameter $d(X, X_{\{m\}})$ at which γ reaches unity, and τ which is the forgetting rate.

In our approach, we decrement each exemplar's weight by a factor γ, proportional to the proximity of a subsequent observation, based on a truncated m-nearest neighborhood influence function. A new observation is initialized with a weight $w = 1$. Each time a new exemplar is input, the weightings $w_{\{k\}}$ of kth nearest observations, $X_{\{k\}}$, $(k \leq m)$, within a neighborhood of the m nearest-neighbors of the new exemplar X are decreased by multiplication with γ.

$$w'_{\{k\}} = \gamma\left(X, X_{\{k\}}\right) w_{\{k\}} \tag{3}$$

When a given observation's weighting falls below some threshold value θ, it is deleted from the learning set. The quantity $\gamma\left(X, X_{\{k\}}\right)$ is computed based on the following truncated quadratic function (depicted in Figure 6):

$$\gamma\left(X, X_{\{k\}}\right) = \begin{cases} \tau + (1-\tau)\frac{d^2_{\{k\}}}{d^2_{\{m\}}} & \text{if } d^2_{\{k\}} \leq d^2_{\{m\}} \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

Here $X_{\{i\}}$ is the location of the ith nearest neighbor, X is location of the new observation, and $d_{\{i\}} = d^2(X, X_{\{i\}})$ is the Euclidean distance from the ith nearest neighbor to the new observation. Since the radius $d_{\{m\}}$ is that of the sphere containing the m nearest neighbors, this adapts the decay radius of influence to the local density of exemplars around the new exemplar.

The parameters τ, m and θ determine how many nearby subsequent observations are necessary in the neighborhood of a previous observation before it is deleted from the learning set. The weights decay in a stochastic fashion as illustrated in Figure 7, where an observation decays only if it happens to be one of the m nearest-neighbors of a succeeding observation. A straightforward way to implement a bound on the number of exemplars stored is to set m, the number of nearest-neighbors, equal to some fixed fraction β, $(0 < \beta \leq 1)$, of the total number of observations currently in the learning set [16]. By doing this, the total number of observations that are kept in memory can be set to reach some asymptotic value based on the selection of the τ, β and θ parameters (See Figure 8).

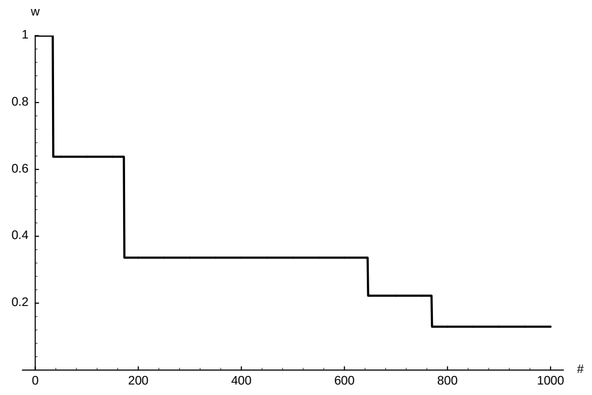

Figure 7: A plot of an actual weight decay as a function of the number of succeeding observations. Whenever the observation is within an m-th nearest neighbor interval, its weight value decreases as a function of the distance between the exemplar and the succeeding observation.

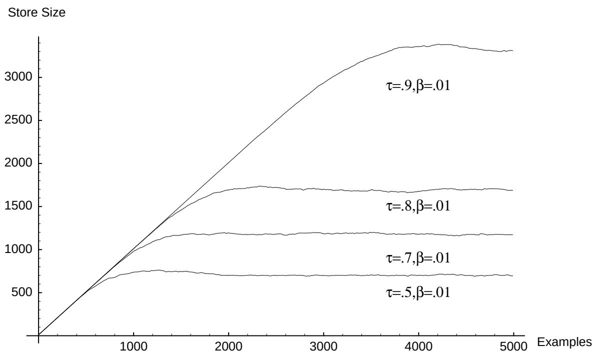

Figure 8: The asymptotic number of exemplars held in the memory store as a function of β and τ for a two dimensional domain with uniform exemplar distribution.

3.1 An Example: Adapting to Action Errors

To illustrate the algorithms discussed, we use a simple simulation developed to test the ability of the DARLING algorithm to adapt to the situation where one of the fingers on a two fingered robotic gripper is jammed at its extreme position. The task of the learner is to determine whether approaching from the object's z-axis direction when its z-axis is pointed upwards will succeed as a function of object dimensions, as schematically illustrated in Figure 9. After the failure, the gripper is still functional, it can pick up objects by squeezing with its operational finger and compressing the object against the jammed finger. However, the range of objects that can actually be picked up is decreased dramatically, and is now limited to a small interval.

The selection map (see Figure 10) for a given approach-orientation combination makes a prediction as to when the corresponding interaction will succeed. The prediction is made in terms of the parametric description of the object, its reduced superquadric description, which is essentially a bounding box representation. For example, the success or failure of the z-axis up, z-axis approach (z_{up}-z_{app}) is a function of the extents of the object perpendicular to the approach axis, namely a_x and a_y. The white areas of the selectivity map (Figure 10(a)) indicate the set of objects with a_x, a_y dimensions that are predicted to succeed in the ideal case, while the grey areas indicate objects whose dimensions would predict failure. The width of the white region is 38mm, which is the maximum width that the jaws of the gripper can open, which, in turn, determines the widest object than can be grasped.

When the finger is jammed, then the underlying selection maps change so that the width of objects now graspable ranges from 25 mm to 38 mm (see Figure 10(b)), rather than 0 to 38 mm. The transition is therefore from the concept depicted in the 10 (a) to that in 10 (b). Figure 11 shows the learned selectivity maps created by the DARLING tree. The transition to the broken finger from the operational one occurs at $n = 2000$. At this point, the prediction performance is seen to degrade significantly (see Figure 12). The error then decreases with further observations as the forgetting algorithm gradually deletes the older observations and supersedes them with observations that reflect the current state of affairs of the environment.

4 Conclusion and Future Extensions

As can be seen in the simulation, a significant number of objects must be attempted (in practice around 200 are needed) before the learner begins to converge on the correct underlying selection map. This is due to the fact that the algorithm we have chosen is non-parametric in terms of the description of the concepts

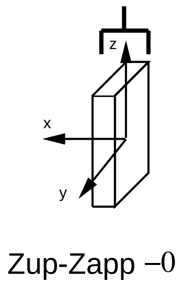

Figure 9: Approaching an object in the z-axis up configuration from the z-axis direction

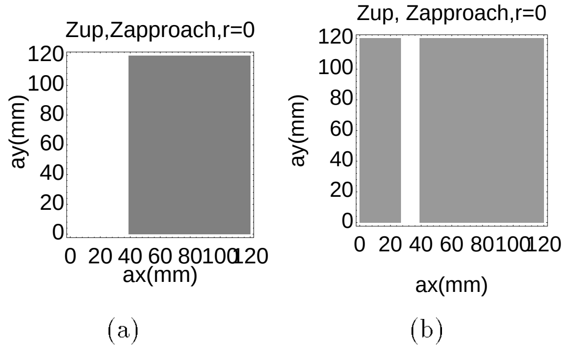

Figure 10: The beginning and ending selectivity maps

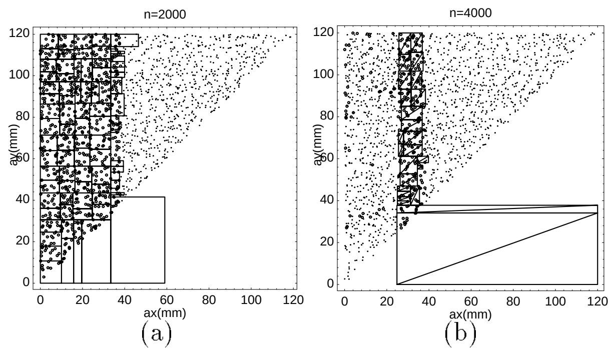

Figure 11: Adapting to a Malfunction. At $n = 2000$ in the simulation, one of the fingers in the gripper jams at its extreme position, causing the range of objects that are graspable to decrease. The selectivity map learned tracks the changes of the underlying maps depicted in 10. The learning and forgetting algorithms had the parameter values $\alpha = .5$, $p_{min} = .6$, $p_{max} = .4$, $\tau = .8$ and $\beta = .01$. The upper diagonal distribution of the points is due to the canonical superquadric representation where $a_x \leq a_y \leq a_z$.

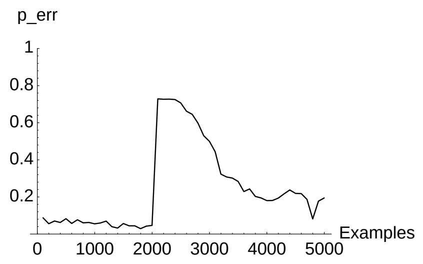

Figure 12: The learning curve for the simulation. It can be seen that the learning curve jumps upward at the $n = 2000$ point, where the finger jams, and then gradually decreases as the system forgets the obsolete observations. The error plotted is an estimate of the probability of the symmetric difference between the true and hypothesized concept.

that it creates and is therefore almost completely data driven. This is the tradeoff between inductive-bias and sample complexity. The less restrictions on the concepts that can be faithfully approximated, the larger the sample size necessary for the learner. We believe that a low-bias approach is warranted in perceptual situations where sensory distortions and miscalibrations can lead to difficult to characterize effects that may not be well approximated by methods with high inductive bias.

There are a number of immediate enhancements that can be made to the learning algorithm. The first is to embed decision trees that use non-axis parallel splits such as perceptron trees [22], or linear discriminants that might be more effective, since they will in general provide much better performance for shallow trees.

Some simple branch-and-bound tests can be implemented to prevent unnecessary subdivision. In particular, if the node's current best case split has a p_{-} estimate which is lower than the current node's p_{-} lower bound, then further subdivision will not yield an improvement and the node should be abandoned.

Additionally, by merging neighboring regions with like labellings, the resulting trees could be simplified, possibly yielding better generalization, and the error breakdown as a function of random mislabelling errors could be improved. The merging process would pool larger numbers of observations, which would yield better noise immunity.

Another drawback is that the search is greedy. It searches for partitions with sufficing pessimistic success probabilities and stops splitting when they are met, rather than for the highest possible lower bounds. It might be the case that further splitting yields a partitioning which has higher lower bound probabilities, even though the current bound is acceptable. A hill climbing lookahead search might yield better solutions.

The forgetting algorithm has the advantage of respecting locality of observations since the metric of a locale is determined by the local density of exemplars. This allows for efficient updating of the learning set so that new observations decay only their neighbors. This especially advantageous when the sampling distributions for the learning set is non-stationary and moves between different areas of the attribute space over time. Using *time-weighted* forgetting, exemplars in inactive areas would be deleted unnecessarily, whereas with density-adaptive forgetting they will persist until new evidence is available to supersede them. On the other hand, if the sampling distribution is stationary then density-adaptive forgetting behaves identically to time-weighted forgetting, so there is no penalty in adopting it over time-weighting forgetting in the first place.

Since the k-D tree built for the DARLING is an efficient structure for nearest-neighbor lookup, returning the nearest neighbors in time $O(\log n)$, where n is the number of observations, it is synergistic with the use of the nearest-neighbor forgetting approach.

Schlimmer *et al.* [18] point to the characteristic of *resiliency* in learning systems as a property that occurs in human and animal learning. Resiliency is the property of a learning system that the longer a behavior is trained, the longer it takes to unlearn subsequently. While algorithms which display this property may be more psychologically suggestive, resiliency can be counterproductive in rapidly changing environments. In particular, if a system operates on-line for very long periods of time before a change occurs, it will take an unacceptably long time to forget the obsolete concept. Fortunately exponentially weighted techniques that actively delete exemplars will not suffer from the same level of resiliency, since they have a natural saturation in the size of the learning set (see Figure 8) which affords turnover.

The forgetting parameters β and τ are task dependent. It is therefore important to be able to estimate the appropriate forgetting parameter for a given task. The more rapidly a task environment changes, the larger β should be and the smaller τ should be. However, there is an obvious tradeoff, the more rapid the forgetting rate, the fewer the asymptotic number of exemplars in the learning set and the worse the overall learning performance. Much work remains in devising automatic techniques for selecting these parameters. Moore suggests a technique for estimating task-specific forgetting parameters using cross-validation in [11].

Once forgetting is implemented, another important question is how often to regenerate the classification tree structures. In this work, new observations are simply inserted into their corresponding *k-D leaf* as they come in, in order to allow the nearest-neighbor forgetting queries to continue. The DARLING tree is rebuilt modulo 100 observations. If on-line performance is needed, an interleaved tree-building schedule may be used, so that the preceding decision-tree is used on-line while its successor is being built either in a background process, or on another processor. Alternatively, [17] discusses some issues in incrementally updating adaptive *k-D*-trees.

This issue might be sidestepped by using the forgetting technique together with other incremental techniques such as ID-5 [21], or using it with nearest-neighbor prediction techniques that do not form explicit decision-trees [1].

Acknowledgements

The work was supported by an NSF Postdoctoral Research Associateship in Computational Science and Engineering (NSF-CDA92-11136), Navy Grant N0014-88-K-0630, AFOSR Grants 88-0244, AFOSR 88-0296; Army/DAAL 03-89-C-0031PRI; NSF Grants CISE/CDA 88-22719, IRI 89-06770; and Du Pont Company.

References

[1] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] S. Bennett. Planning to address uncertainty: An incremental approach employing learning through experience. In *Proceedings of the DARPA Workshop on Advanced Planning and Control Strategies*, pages 313–324, 1991.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees.* Wadsworth International, Belmont, CA, 1984.

[4] G. B. Dunn and J. Segen. Automatic discovery of robotic grasp configuration. In *Proceedings, 1988 IEEE International Conference on Robotics and Automation*, pages 396–401, 1988.

[5] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3:209–226, 1977.

[6] L. P. Kaelbling. *Learning in Embedded Systems.* PhD thesis, Stanford University, Stanford, CA, 1990. Dept. of Computer Science.

[7] A. Kuh, T. Petsche, and R.L. Rivest. Learning time-varying concepts. In *Advances in Neural Information Processing Systems 3*, pages 183–189, Morgan Kaufmann, December 1991.

[8] P. Maes and R. A. Brooks. Learning to coordinate behaviors. In *AAAI-90*, pages 796–802, 1990.

[9] S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55:311–365, 1992.

[10] B. Mel. *Connectionist robot motion planning: A neurally inspired approach to visually guided reaching.* Academic Press, San Diego, CA, 1991.

[11] A. W. Moore. Fast, robust adaptive control by learning only forward models. In *Advances in Neural Information Processing Systems*, Morgan Kauffman, December 1991.

[12] S. M. Omohundro. Efficient algorithms with neural network like behavior. *Complex Systems*, 1:273–347, 1987.

[13] S. M. Omohundro. Geometric learning algorithms. *Physica D, Non-Linear Phenomenon*, 42:307–321, 1987.

[14] J. R. Quinlan. *Learning efficient classification procedures and their application to chess end games*, pages 463–481. Morgan-Kauffman, Los Altos, Ca., 1986.

[15] H. Ritter, T. Martinetz, and K. Schulten. *Neural Computation and Self-Organizing Maps: An Introduction.* Addison-Wesley, Reading, Massachussetts, 1991.

[16] M. Salganicoff. *Learning and Forgetting for Perception-Action: A Projection-Pursuit and Density-Adaptive Approach.* PhD thesis, University of Pennsylvania, 1992. Dept. of Computer and Information Science.

[17] Hanan Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, Reading, Massachussetts, 1989.

[18] J. C. Schlimmer and R. H. Granger. Beyond incremental processing: Tracking concept drift. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 502–507, Morgan Kauffman, Philadelphia, PA, 1986.

[19] F. Solina. *Shape Recovery and Segmentation with Deformable Part Models.* Technical Report MS-CIS-87-111, University of Pennsylvania, Dept. of Computer and Information Science, Philadelphia, Pa., 1987.

[20] M. Tan. A cost-sensitive learning system for sensing and grasping objects. In *Proceedings, 1990 IEEE International Conference on Robotics and Automation*, IEEE Press, 1990.

[21] P. Utgoff. ID-5: An incremental ID-3. In *Proceedings of the Fifth International WOrkshop on Machine Learning*, Morgan-Kauffman, 1988.

[22] P. Utgoff and C.E. Brodley. An incremental method for finding multivariate splits for decision trees. In *Proceedings of the Seventh International Workshop on Machine Learning*, Morgan-Kauffman, 1990.