

# Developmental Learning of Memory-Based Perceptual Models

Yuri A. Ivanov

Fundamental Research Labs  
Honda R&D Americas  
Boston, MA 02111

Bruce M. Blumberg

Media Laboratory  
Massachusetts Institute of Technology  
Cambridge, MA 02139

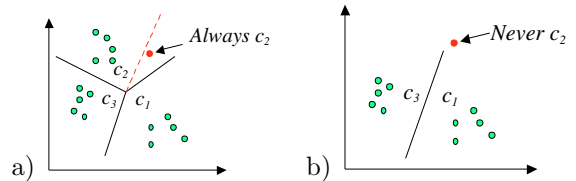
## Abstract

This paper reviews an on-line learning algorithm for incremental learning of memory-based utterance models. The simple core algorithm is augmented with a strategy for selecting the compression set - a subset of the input data that possesses some optimality characteristics. These sets are again found incrementally. Several strategies are formulated and empirically compared on three standard data sets. The algorithm is used as a perceptual learning component of an adaptive autonomous agent.

## 1 Introduction

In our work on autonomous learning agents we have frequently encountered a situation where the agent needs to learn a new category. In a very rough terms, the architecture of the agent should include two main components - a perceptual system, and a policy of action selection. The former is simply a set of clusters in the space of sensory observations, while the latter is some form of mapping of these perceptual categories to actions, that is a set of rules that dictate to the agent to take a particular action,  $a_i$  when it observes the event  $x \in s_j$ , where  $s_j$  is a perceptual category.

Ideally we would like to build an agent that learns both parts from experience, guided by its internal and external biases, much in the way in which animals acquire their skills under training. In fact, we would like to take ideas from animal training quite literally and use the outcomes of action selection as grounding for perception. Consider an example from animal training. When a dog hears an utterance “sit” it has no semantic context. The meaning is further acquired only when the consistent fact is established that the dog gets a treat if it sits down when an acoustic pattern similar to “sit” is heard. That gives a rise to semantic grounding of the class of utterances which are grounded in utility of selecting a particular action, where the utility is



**Figure 1:** Greedy label selection. a) The true class boundary (red dashed line) does not coincide with the empirical estimate (solid black line). Queries falling inside the wedge between true and empirical boundaries always receive the same label. b) When no data has been seen for a class  $c_2$  the query point is never assigned a label  $c_2$ .

expressed in terms of *expected reward*.

The full autonomy of the agent implies that the agent can only learn its perceptual organization from indirect supervision. That is, from a signal that evaluates the appropriateness of its action selection mechanism, regardless of what was its perceived cause. This makes the problem of perceptual learning more difficult than supervised and transductive learning, while still less difficult than fully unsupervised. We term this class of problems where supervision is partial and indirect *weakly transductive*.

The learning of both perceptual and policy components is taking place in the complete agent. In this paper we only present the learning process that occurs in its perceptual space, while the complete system is described in [8]. Other variations on category representation are also explored in [10].

### 1.1 Related Work

We are inspired by ideas of developmental learning advocated by Weng et al., [13]. Working in the developmental setting lead us to the incremental problems to which we devote this paper. The problem of grounding perception has been addressed before. Roy and Pentland, [11] looked at cross-modal

grounding of perception.

Parts of this paper are based on the Support Vector machinery, introduced by Vapnik, [12]. In our work we use the Dynamic Alignment (DynA) Kernel, [10], within the SVM framework, which allows using SVM for classification of unaligned sequences.

The problem of finding *compression sets*, [5], or representative subsets of the data, has been addressed mostly in the setting of batch processing. In particular, ‘‘Hart’s rule’’, [7], is often used and reviewed by researchers in pattern recognition, e.g. [3, 4, 6]. In contrast, our methods are incremental.

## 2 Incremental Learning Algorithm

Imagine the following incremental classification scheme – a classifier assigns the same label to a query point as its nearest neighbor, and gets a feedback from an oracle indicating if the assignment was correct. If it is, the query is stored, otherwise discarded. Such a scheme will be called *greedy*.

The greedy scheme encounters two problems, illustrated in figure 1. Figure 1 a) shows that all points landing in the wedge between the true and the empirical boundaries are mislabelled, regardless of how frequently the labels in that region are rejected by the oracle. Figure 1 b) demonstrates the situation where the classifier has not yet seen any data for one of the classes. This class is never discovered since no neighbor having this class’ label will ever exist. In order to alleviate these problems, an *exploration* strategy should be adopted by the algorithm.

### 2.1 Exploration Strategy

To implement the exploration strategy, the learning algorithm uses a probabilistic label<sup>1</sup> assignment, while incrementally building the model for each class. The full classifier model  $\mathcal{M}$  is a collection of class models,  $\mathcal{M} = \{\mathcal{M}_k\}_{k=1}^K$ , each of which is a collection of correctly classified samples,

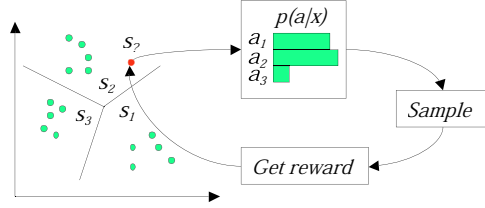
$$\mathcal{M}_k = \left\{ x_i^{(k)} \right\}_{i=1}^{C_k}. \text{ That is:} \quad (1)$$

$$\mathcal{M} = \left\{ x : x \in \left\{ x_i^{(k)} \right\}_{i=1}^{C_k} \right\}_{k=1}^K$$

where  $0 \leq C_k \leq C_{max}$  – number of samples added so far, and  $C_{max}$  is the maximum number of samples in the model, if applicable.

For a discriminant function for a class  $k$ ,  $g_k(\cdot)$ , separating in- and out-class samples, a *pseudo-distance* is calculated by evaluating  $g_k(\cdot)$  at the

<sup>1</sup>In this paper the terms ‘‘action’’, ‘‘state’’ and ‘‘label’’ are used interchangeably. In [9] we explore problems involving a more complicated policy with local state models.



**Figure 2:** Illustration of the sample selection procedure. A new query point,  $x^n$ , is used to compute the label probability distribution,  $p(a|x^n)$ . The action is selected by sampling  $p(a|x^n)$ . The reward dictates whether or not to include the query  $x^n$  into the model.

query point,  $x^n$ . This results in a  $K$ -vector of distances from  $x^n$  to each of the class models:

$$\mathbf{d}(x^n) = \{g_k(x^n)\}_{k=1}^K \quad (2)$$

The algorithm assigns a label  $s_i$  to a class with a probability  $P(s = s_i|x^n)$ , computed in proportion to these pseudo-distances:

$$p(a|x^n) = p(s|x^n) = \frac{e^{-\beta\bar{r}\mathbf{d}(x^n)}}{\sum_s e^{-\beta\bar{r}\mathbf{d}_s(x^n)}} \quad (3)$$

where,  $\beta$  is a free parameter; and  $\bar{r}$ , the expected discounted reward.

This distribution is sampled to select a label,  $s^n = s_\ell$ , and an action  $a^n = a_\ell$ . If the action of the agent is rewarded,  $x^n$  is added to the class model,  $\mathcal{M}_\ell$ , and the value of the expected discounted reward is updated:

$$\bar{r}_t = \bar{r}_{t-1} + \alpha(r - \bar{r}_{t-1}) \quad (4)$$

where  $r$  is the momentary reward and  $\alpha$  is a learning rate set to some small value,  $0 \leq \alpha \leq 1$ . The procedure is illustrated in the figure 2 and is summarized below:

#### Sample Selection Algorithm

1. Receive an observation,  $x^n$ ;
2. Compute distances  $\mathbf{d}(x^n)$  (eqn. 2);
3. Calculate  $p(a|x^n)$  (eqn. 3);
4. Select an action  $a^n = a_\ell$  by drawing a sample from  $p(a|x^n)$ ;
5. Update  $\bar{r}$  with the reward,  $r$  (eqn. 4);
6. If  $r = 1$  add  $x^n$  to the model  $\mathcal{M}_\ell$ ;
7. Proceed to step 1

## 2.2 Compression Sets

In a memory-based algorithm, both memory and processing power requirements grow with increase of the number of stored samples. This calls for methods of compressing the data set.

Unlike in the batch setting, in the on-line algorithm, only the data kept in class models is available. At each step the decision needs to be made a) whether to include the successful query point in the prototype set; and b) which prototype to discard from the model to free up space for the new one.

In this respect our algorithm introduces and compares several strategies of sample compression – a) based on utility; b) based on empirical risk; c) based on class boundaries; and d) based on margin samples.

### 2.2.1 Utility Compression Rule

Each prototype<sup>2</sup>,  $x_j^{(i)}$ , of each class model,  $\mathcal{M}_i$ , is augmented with a *usage count*,  $Z$  - a number of times the prototype has been found to be the closest to a query,  $x^m$ ; and a *lifetime count*,  $L$  - a number of queries to the classifier since the prototype  $k$  was inserted into the model. The *utility index* of the prototype,  $U$ , is computed as a ratio of the corresponding usage count to its lifetime:

$$U(x_j^{(i)}) = \frac{Z(x_j^{(i)})}{L(x_j^{(i)})} \quad (5)$$

When the sample  $x^n$  is considered for addition to the model  $\mathcal{M}_i$ , and the model contains  $C_{max}$  prototypes - a maximum allowed number for the model, then the prototype with the lowest utility index is replaced with  $x^n$ .

Utility Compression Rule
<ol style="list-style-type: none"> <li>1. Increment <math>L</math> of every stored sample;</li> <li>2. If the reward <math>r &gt; 0</math> and the winning model index is <math>i</math> <ol style="list-style-type: none"> <li>(a) Increment the usage count, <math>Z</math> of the nearest sample of <math>\mathcal{M}_i</math>, <math>x_j^{(i)}</math>;</li> <li>(b) If <math> \mathcal{M}_i  &gt; C_{max}</math> discard from it the sample with the smallest <math>U</math>;</li> <li>(c) Append <math>x^n</math> to the set <math>\mathcal{M}_i</math>;</li> </ol> </li> </ol>



<sup>2</sup>Parenthesized superscript denotes the index of a class model, while the subscript indexes the sample within the class model.

### 2.2.2 Minimum Risk Compression Rule

After the sample  $x^n$  is classified into the class  $C_i$  and added to the model  $\mathcal{M}_i$  the algorithm removes a sample which is found to be redundant. That is, the  $k$ -th sample of the  $i$ -th set,  $x_k^{(i)}$ , which is the best predicted by the remaining samples in the model. Misclassification risk for the sample  $x_k^{(i)}$  is computed as the probability of classifying it into a class  $j \neq i$ , given all the present model samples with  $x_k^{(i)}$  removed from the set  $\mathcal{M}_i$ :

$$\begin{aligned} r(x_k^{(i)}, \{x_{NN}^{(j)}\}_{j=1}^K) &= p(C_k \neq i | x_k^{(i)}, \{x_{NN}^{(j)}\}_{j=1}^K) \\ &= 1 - \frac{e^{-\zeta \|x_k^{(i)} - x_{NN}^{(i)}\|^2}}{\sum_{j=1}^K e^{-\zeta \|x_k^{(i)} - x_{NN}^{(j)}\|^2}} \end{aligned} \quad (6)$$

The sample  $x_k^{(i)}$  minimizing the risk in eqn. 6 is removed from  $\mathcal{M}_i$ :

$$\mathcal{M}_i = \mathcal{M}_i \setminus \arg \min_{x_k^{(i)}} \left[ r(x_k^{(i)}, \{x_{NN}^{(j)}\}_{j=1}^K) \right] \quad (7)$$

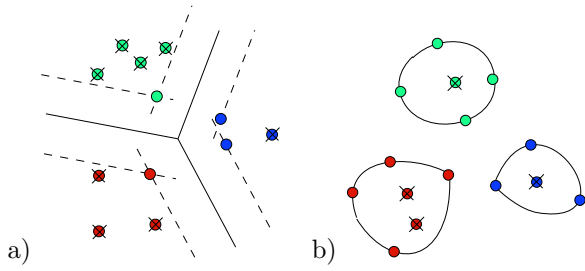
It is clear that the removed sample is the one with the maximum value of the posterior probability,  $p(a|x_k^{(i)})$ .

Minimum Risk Compression Rule ( $r > 0$ )
<ol style="list-style-type: none"> <li>1. If <math> \mathcal{M}_i  &gt; C_{max}</math> compute the risk for every sample in <math>\mathcal{M}_i</math>;</li> <li>2. Discard the sample with the smallest value of risk;</li> <li>3. Append the query <math>x^n</math> to the set <math>\mathcal{M}_i</math>;</li> </ol>



### 2.2.3 Margin Compression Sets

Another approach to finding useful compression sets for the given set of models is based on the observation that the areas of the space where the classification boundaries are constructed are more important to be included in the models than the data lying in the areas distant from the decision boundary. The figure 3 a) illustrates the intuition behind this scheme. It shows that for making a classification decision the data located far from the margin between classes does not contribute to the decision boundary, and, therefore, can be safely removed.



**Figure 3:** a) Margin compression sets are found by discarding the data lying away from the margins; b) Boundary compression sets are the points lying on the cluster boundary of the classes.

The solution for the problem of identifying the points lying on the margin (so called Support Vectors) is readily available from a technique known as Support Vector Machine (SVM). For the full discussion of the technique the reader is directed to the literature (e.g. [12]).

In linear SVM the solution to the classification problem is based on finding a hyperplane that separates the data with minimum error. Often a non-linear transformation is applied to the data prior to the parameter estimation and classification, giving rise to other types of SVMs - RBF, polynomial, etc.

The optimal separating plane is determined by a vector  $w$  that minimizes the margin functional:

$$F(w, \xi) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i \quad (8)$$

subject to the constraint:

$$y_i(w \cdot z_i + b) \geq 1 - \xi_i \quad (9)$$

where  $C$  is fixed and  $\xi_i$  is a slack variable.

The feature space can be chosen in a special way, so that the inner product in that space is represented by a kernel function in the input space,  $(z_i \cdot z_j) = (\Phi(x_i) \cdot \Phi(x_j)) = K(x_i, x_j)$ . The resulting discriminant function, with values for the Lagrange multipliers,  $\alpha_i$  provided by quadratic optimization, is given as follows:

$$g(x^n, D) = \sum_{i: x_i \in \{SV\}} \alpha_i y_i K(x_i, x^n) - b \quad (10)$$

Data lying on the margin boundary will have their corresponding  $\alpha_i \neq 0$ . These are the vectors that are important for making the classification decision that the algorithm needs to keep. The margin compression rule computes a new set of support vectors every time a model includes a new sample and possibly reduces it:

#### Margin Compression Rule ( $r > 0$ )

1. Append the query  $x^n$  to the set  $\mathcal{M}_i$ ;
2. Find a set of support vectors,  $\mathcal{M}_i^{SV} \subseteq \mathcal{M}_i$  as a solution to the classification problem  $\mathcal{M}_i$  vs.  $\{\mathcal{M}_j\}_{j=1, j \neq i}^N$ ;
3. Replace the set  $\mathcal{M}_i$  with the reduced set  $\mathcal{M}_i^{SV}$ .

#### 2.2.4 Boundary Compression Rule

The boundary compression rule is based on finding the support of a cluster and identifying the points lying on its boundary (see fig. 3 b)). The cluster support set is found with the use of a “kernel trick” - mapping the data to some higher-dimensional (feature) space and finding the smallest enclosing sphere for it. Then, the points lying on the surface of the sphere in the feature space can be shown to lie on the cluster’s exterior in the original space (see [1]).

The boundary points are found by minimizing the radius of the enclosing sphere,  $R$  from the following constraint:

$$\|\Phi(x_j) - a\|^2 < R^2 + \xi_j \quad (11)$$

with slack variables  $\xi_i \geq 0$  and  $a$  the sphere center. Again, the solution is found by quadratic optimization, with Lagrange multipliers,  $\alpha_i > 0$ , indicating that the corresponding points lie on a cluster boundary. The Boundary Compression strategy estimates the new set of cluster boundary points at every step when the model includes a new sample, and possibly reduces it:

#### Boundary Compression Rule ( $r > 0$ )

1. append the query  $x^n$  to the set  $\mathcal{M}_i$ ;
2. Find a set of boundary points  $\mathcal{M}_i^B \subseteq \mathcal{M}_i$  as a solution to the minimum enclosing sphere problem;
3. Replace the set  $\mathcal{M}_i$  with  $\mathcal{M}_i^B$ .

#### 2.2.5 Summary of Compression Set Rules

The summary of the compression rules is shown in the table 2.2.5. It lists the strategy of the set reduction for each classifier, followed by the pseudo-distance metric for computing the probability of action selection,  $p(a|x^n)$ , as shown in eqn. 3. :

	Compression Set of $\mathcal{M}_i$	Decision Metric
KNN	$\mathcal{M}_i \setminus \arg \min_{x_j^{(i)}} \left\{ \frac{Z(x_j^{(i)})}{L(x_j^{(i)})} \right\}$	$d_k = \ x^n - x_{NN}^{(k)}\ ^2$
KNE	$\mathcal{M}_i \setminus \arg \max_{x_j^{(i)}} \left\{ \frac{e^{-\zeta \ x_j^{(i)} - x_{NN}^{(i)}\ ^2}}{\sum_{k=1}^K e^{-\zeta \ x_j^{(i)} - x_{NN}^{(k)}\ ^2}} \right\}$	$d_k = \ x^n - x_{NN}^{(k)}\ ^2$
SVC	$\mathcal{M}_i \setminus \left\{ x_j^{(i)} : \ \Phi(x_j^{(i)}) - a_i\ ^2 < R_i^2 + \xi_j \right\}$	$d_k = - \left( \sum_{j=1}^{ \mathcal{M}_k } y_j \alpha_j K(x_j^{(k)}, x^n) - b_k \right)$
SVS	$\mathcal{M}_i \setminus \left\{ x_j^{(i)} : [w_i \cdot \Phi(x_j^{(i)}) + b_i] > 1 - \xi_j \right\}$	$d_k = \ x^n - x_{NN}^{(k)}\ ^2$
SVL	$\mathcal{M}_i \setminus \left\{ x_j^{(i)} : [w_i \cdot x_j^{(i)} + b_i] > 1 - \xi_j \right\}$	$d_k = - \left( \sum_{j=1}^{ \mathcal{M}_k } y_j \alpha_j (x_j^{(k)} \cdot x^n) - b_k \right)$
SVM	$\mathcal{M}_i \setminus \left\{ x_j^{(i)} : [w_i \cdot \Phi(x_j^{(i)}) + b_i] > 1 - \xi_j \right\}$	$d_k = - \left( \sum_{j=1}^{ \mathcal{M}_k } y_j \alpha_j K(x_j^{(k)}, x^n) - b_k \right)$

**Table 1:** Summary of compression rules and sampling strategy. For each classifier, the table shows the set reduction technique in the second column, followed by the pseudo-distance metric based on which the probability of label assignment is computed.

**KNN:** Utility reduction strategy (eqn. 5) with Euclidean distance as the decision metric.

**KNE:** Minimum risk strategy (eqn. 7) with Euclidean distance as the decision metric.

**SVC:** Cluster Boundary rule with the discriminant function (eqn. 10) providing the metric.

**SVS:** “Silly” SVM with the Margin rule and a Euclidean decision metric in the input space<sup>3</sup>.

**SVL:** Linear SVM with the Margin rule and the discriminant decision metric (eqn. 10).

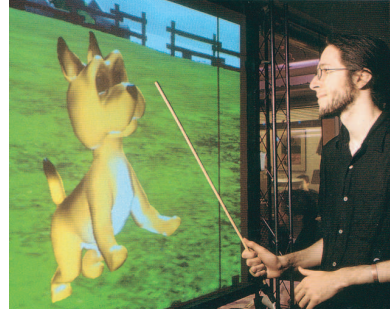
**SVM:** RBF Kernel SVM with the Margin rule and the discriminant decision metric (eqn. 10).

### 3 Experimental Results

In this section we report results of the application of this algorithm to incremental learning in the complete agent system, as well as present quantitative evaluation of its performance on standard data sets taken from the UCI Machine Learning Repository [2]. Three data sets were used: a) Iris data set; b) Wine data set; and c) Japanese Vowel data set. The following presents them in order. In all experiments the results are averaged over 120 runs with data randomly reordered. For the KNN and KNE set compression methods, the parameter  $C_{max}$  was set to 20.

**Trial by Eire.** The algorithm of this paper was developed for the “Trial by Eire” installation. The installation was used with a large number of users in a variety of noise conditions and showed a fairly

<sup>3</sup>It is “silly” because the decision metric does not match the metric used in the compression rule.

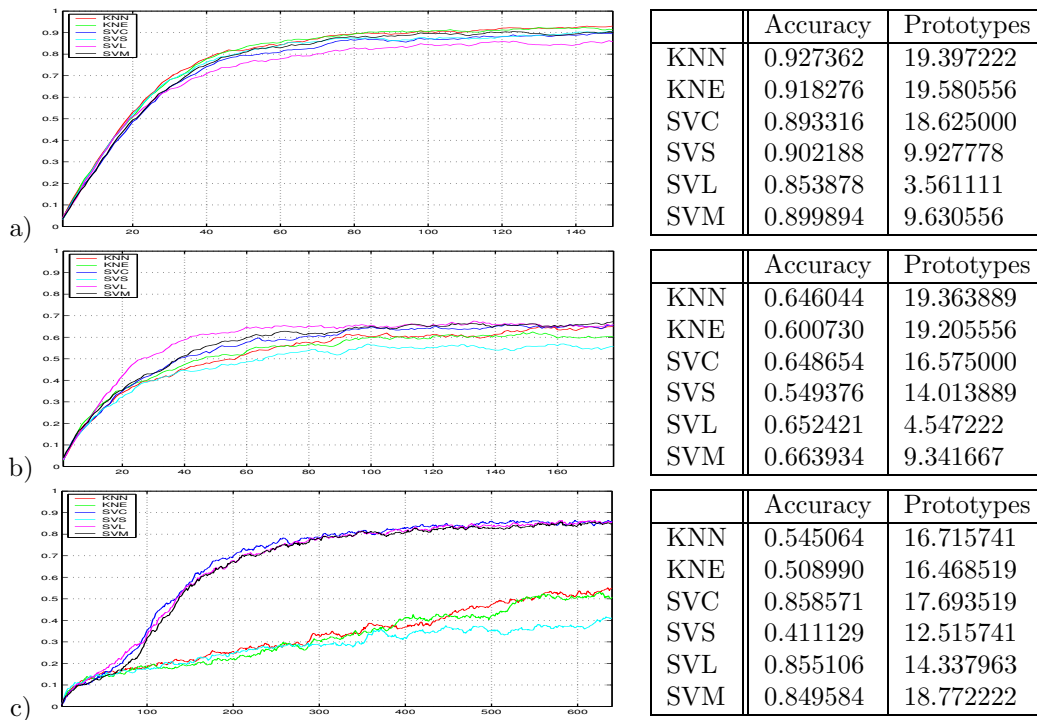


**Figure 4:** Training Duncan.

robust performance even with the simplest utility-based compression. In the live system, the user would have a goal of training the artificial dog, Duncan (see fig. 4), to respond to a set of 5-7 sheep herding voice commands. Although users varied in their gender, and native tongue, the training would normally take 2-3 minutes for the whole set. A quantitative evaluation of the properties of the algorithm follows in the remainder of this section.

**The Iris Set.** The Iris data set consists of 150 4-dimensional measurements of 3 species of the Iris plant. One class is linearly separable from the other two, while the latter are not linearly separable from each other.

The results are presented in the figure 5a). The second column of the table shows the accuracy attained by the incremental classifier after seeing the whole data set. The third column shows the number of retained prototypes per class.



**Figure 5:** Results of running the algorithms on the Iris (a), Wine (b) and Vowel (c) datasets. The plots on the left show average accuracy attained as a function of the number of samples. The tables on the right show the final accuracy and the average number of prototypes retained by each class model.

This proved to be a simple data set where nearest neighbor classifiers achieved marginally best performance, while retaining the maximum number of prototypes ( $\sim 38\%$  of the data). Linear SVM performed the worst but only used 3.6 prototypes per class (7.2% of the data).

**The Wine Set.** The Wine data set consists of 178 samples of measurements of 13 parameters of 3 types of Italian wine. The results are presented in the figure 5b).

The RBF and linear SVM compression models showed the best performance at the expense of storing 15.6% and 7.62% of the data respectively. The latter, as well as a good performance of the boundary rule show that the classes were well separable.

**The Japanese Vowel Set.** The last experiment was run on a Japanese Vowel data set. The set consists of 640 sequences of 12 LPC cepstral coefficients. The sequences contain 7 to 29 coefficient vectors each and are collected from 9 male speakers, pronouncing the Japanese vowel “æ”.

The linear SVM was run on the time-normalized data, mapped onto a 252-dimensional space. Other SVMS used the more practical *DynA* kernel, (see [10]). The results are presented in the figure 5c).

The boundary rule, the Linear and RBF SVMs performed significantly better than other contenders (24.9%, 20.1% and 26.4% of data stored).

### 3.1 Discussion

While being so simple the core algorithm contains a sketch of a powerful idea of using whatever little information that is available from the environment to refine perception. What is most interesting here is the different ideas of incremental forming reduced memory-based models.

The Iris data set gave no surprises. Classification rates of all algorithms were relatively high - around 90%. In this situation, only the linear SVM showed a slightly poorer performance taking the hit on the two non-separable classes of the set. The low dimensionality-to-sample-size ratio provided good conditions for the nearest neighbor rules.

Despite the fact that the Wine set is usually called “easy”, the incremental classification of it was not very successful. The problem that the all classifiers encountered was the uneven scale across dimensions. While whitening the data would have improved the results, the necessity to see the data beforehand is undesirable.

On the more interesting Japanese Vowel set the

dramatic improvement in performance achieved by the SVM-based techniques illustrates a good generalization ability that the SVM can provide. The high dimensionality of the space with barely 3 samples per dimension for all 9 classes showed to be an obstacle for the local neighborhood rules.

Perhaps the biggest surprise of all the experiments was the excellent performance of the boundary compression rule. This strategy tends to pick outermost samples of the class, without any concern for outliers. Despite this, on the examined data sets the performance of the boundary compression rule was on par with margin rules.

## References

- [1] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik. A support vector method for clustering. In *NIPS 13*, pages 367–373. MIT Press, 2000.
- [2] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998.
- [3] C Chang. Finding prototypes for nearest neighbor classifiers. *IEEE Transactions on Computers*, 26:1179–1184, 1974.
- [4] P. Devijver and J. Kittler. On the edited nearest neighbor rule. In *ICPR*, pages 72–80, Los Alamitos, CA, 1980.
- [5] S. Floyd and M. Warmuth. Sample compression, learnability, and Vapnik-Chervonenkis dimension. Technical Report UCSC-CRL-93-13, UCSC, 1993.
- [6] S. Geva and J. Sitte. Adaptive nearest neighbor pattern classification. *IEEE Transactions on Neural Networks*, 2:318–322, 1991.
- [7] P. E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14:515–516, 1968.
- [8] D. Isla, R. Burke, M. Downie, and B. Blumberg. A layered brain architecture for synthetic creatures. In *IJCAI*, Seattle, WA, 2001.
- [9] Y. Ivanov, B. Blumberg, and A. Pentland. Expectation maximization for weakly labeled data. In *ICML*, pages 218–225, Williamstown, MA, 2001.
- [10] Yuri A. Ivanov. *State Discovery for Autonomous Learning*. PhD thesis, MIT, 2002.
- [11] D. Roy and A. Pentland. Multimodal adaptive interfaces. Technical Report 438, MIT Media Lab, 1997.
- [12] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.
- [13] J. Weng, C. Evans, W. S. Hwang, and Y. B. Lee. The developmental approach to artificial intelligence: Concepts, developmental algorithms and experimental results. In *NSF Design and Manufacturing Grantees Conference*, Long Beach, CA, 1999.