

# Compact Representation of Coordinated Sampling Policies for Body Sensor Networks

Shuping Liu<sup>1</sup>

Anand Panangadan<sup>2,3</sup>

Ashit Talukder<sup>1,2,3</sup>

Cauligi S. Raghavendra<sup>1</sup>

<sup>1</sup> University of Southern California  
Ming Hsieh Department of  
Electrical Engineering  
Los Angeles, CA 90089, USA  
1-213-821-0871  
{lius,raghu}@usc.edu

<sup>2</sup> Childrens Hospital Los Angeles  
4650 Sunset Blvd.  
Los Angeles, CA 90027, USA  
1-323-361-2413  
APanangadan@chla.usc.edu

<sup>3</sup> Jet Propulsion Laboratory  
4800 Oak Grove Drive  
Pasadena, CA 91109, USA  
1-818-354-1000  
Ashit.Talukder@jpl.nasa.gov

## Abstract

Embedded sensors of a Body Sensor Network need to efficiently utilize their energy resources to operate for an extended amount of time. A Markov Decision Process (MDP) framework has been used to obtain a globally optimal policy that coordinated the sampling of multiple sensors to achieve high efficiency in such sensor networks. However, storing the coordinated sampling policy table requires a large amount of memory which may not be available at the embedded sensors. Computing a compact representation of the MDP global policy will be useful for such sensor nodes. In this paper we show that a decision tree-based learning of a compact representation is feasible with little loss in performance. The global optimal policy is computed offline using the MDP framework and this is then used as training data in a decision tree learner. Our simulation results show that both unpruned and high confidence-pruned decision trees provide an error rate of less than 1% while significantly reducing the memory requirements. Ensembles of lower-confidence trees are capable of perfect representation with only small increase in classifier size compared to individual pruned trees.

## Keywords

Body Area Network, Energy efficiency, Markov Decision Process (MDP), Policy representation, Supervised Learning

## 1. Introduction

With the availability of miniature sensors, many new applications are possible with a network of embedded sensors. One such application is called a “Body Area Network”. Here, a patient’s vital signs are continuously measured using multiple physiological and metabolic sensors attached to the patient’s body. Data from the sensors is continuously recorded and transmitted over wireless links to remote health-care staff.

One of the main challenges in developing such a system is ensuring that the system as a whole does not run out of energy too soon. Medical sensors used in Body Area Networks consume relatively large amounts of energy due to the presence of electromechanical

components. Thus, increasing the sampling rates of the sensors decreases the lifetime of the system. Moreover, a higher sampling rate also increases the amount of radio transmission and data processing at the sensor. This is an energy-intensive process and hence further reduces the lifetime of that node. On the other hand, lower sampling rates reduce the accuracy of the measurements. In such multi-sensor systems it is possible to trade-off the sampling rates of the sensors in order to extend the lifetime of the system while maintaining accuracy.

Our work is motivated by a human health monitoring application being developed at the Children’s Hospital Los Angeles and the University of Southern California [1]. The sensing data is locally processed and automatically transmitted to remote relevant health-care staff through the cell phone network (Fig. 1). Multiple sensors enable the subject’s health status to be determined with higher accuracy or to reduce the effect of sensor variability through averaging; novel minimally invasive metabolite sensors, such as the interstitial fluid (ISF) alcohol sensor that we have studied (Fig. 2), have significantly higher error rates than the corresponding invasive blood sampling-based sensors [1]. Other sensors include body temperature, heart rate, blood oxygenation and glucose meter. In this system also, the energy consumed at each sensor node depends on the sampling rate at that sensor. For example, the ISF alcohol sensor requires a pump to draw out ISF before a measurement can be made.

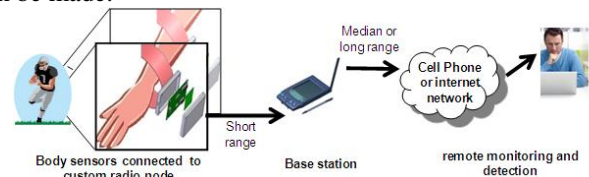


Fig. 1: Two-tier smart health monitoring architecture

Recently, policy-based sensing has attracted academic interest for this type of system [2-4]. A sensing policy determining the optimal sampling rate for all possible states is calculated offline and saved in the sensor memory. The sensors equipped with this sensing policy can intelligently make their own decision about

taking samples by adapting to different environment, so called smart body sensors.

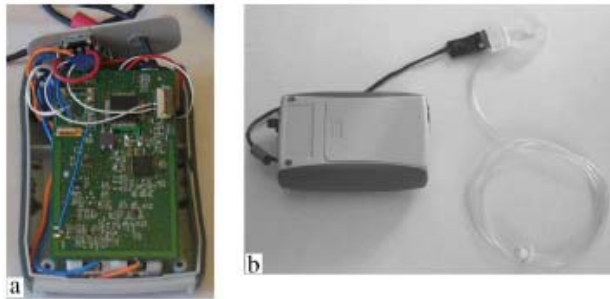


Fig. 2: (a) Internal circuit of ISF sensor containing microcontroller and radio transceivers and (b) ISF alcohol sensor (the tube to the vacuum pump is not connected)

However, the space complexity of the sensing policy is exponential in the number of sensors in the network and the level of discretization of the problem, because the sensing policy includes the actions for all possible situations. Therefore, it is important to represent the sensing policy in a compact way so that policy based sensing could be applied in smart health care application in the real world. In this paper, we describe supervised learning techniques that provide a more compact representation of the decision policy, and which would allow the policy to be stored on embedded body sensors. We have studied various learning mechanisms and evaluated them based on their solution to the memory problem. As far as we know, this is the first effort to represent a large policy in the limited-memory sensors by learning a compact pattern using supervised learning techniques.

## 2. Related Work

There has been extensive research in the past decade on improving energy efficiency in embedded sensor networks at multiple levels, including at the hardware platform, operating system, network protocols and application software [5]. However, all these approaches are either heuristic or deterministic.

Recently, policy-based sensing has been studied [2-4], which calculate an optimal sampling policy offline with a stochastic model. In prior work, we have developed a stochastic MDP framework to optimally coordinate the actions taken by the sensors (sampling rates) to increase energy efficiency and system life time for both single sensor [2] and multi-sensor systems [3]. Yi et al. also developed a constraint MDP policy for the single sensor case [4]. However, little research work has focused on how to represent a large policy on sensors with limited memory size. De Coe et al. [6] integrate all policies in one framework to solve policy conflicts. J. Marecki et al. [7] suggest an algorithm called FANS to represent the

policy obtained from a Partially Observable MDP (POMDP) by employing a finite state machine. Tonti et al. [8] compare three policy representations with semantic web languages: KAOs, Rei, and Ponder. All these works represent the policy in a short and compatible way. But the policy size is still large when the number of sensor nodes in the networks increases, and hence the policies remain infeasible to store on sensors with limited memory capacity. In our current work, we have used supervised learning to compute a compact pattern of the policy and only store the pattern in the sensors, instead of the original policy table. This would enable the policy to be stored even on sensor nodes with limited memory.

## 3. General Multi-agent MDP Model

A Multi-agent Markov Decision Process is defined by a 4-tuple  $(S, A, P, R)$ .  $S$  denotes the finite set of global states. The global state is the joint state of the  $n$  agents,  $(S_1, S_2, \dots, S_n)$ . The joint action space  $A$  is the action concurrently executed by all agents,  $A = \times_{i \in I} A_i$  where  $A_i$  is the action space of node  $N_i$  and corresponds to the set of possible sensor sampling rates.  $P$  is the transition probability function defining how the global state changes when a joint action is executed,  $P: S \times A \times S \rightarrow [0, 1]$ .  $R$  is the immediate reward function and it depends only on the sensor sampling rates and the event criticality. Intuitively, the sampling rate should be higher during critical events. There is a penalty,  $R_{\text{powerout}}$ , if the system runs out of power before the desired lifetime.

If all sensors are able to sense simultaneously, then the reward is proportional to the sum of the sampling rates. We assume optimal sensor fusion under the assumption that the sensors have the same error variance and zero co-variance. This equation can be modified to reflect different error variances and co-variances. The global policy is defined as a function that determines the action for every state  $s \in S$ . The quality of a policy is defined by the average utility, which is the expected sum of future rewards. The value of a state  $s$  under policy  $\pi$ , denoted by  $V^\pi(s)$ , is the expected (discounted) sum of rewards obtained by following the policy  $\pi$  from  $s$ . The value function can be computed by solving Bellman equations using the Value Iteration algorithm. A policy is optimal if the value of every state under that policy is maximal. Hence if we calculate the maximum value function  $V^{\pi^*}(s)$ , then we can obtain the optimal policy.

## 4. Decision Tree-based Learning

Supervised learning is a machine learning technique for learning a function from training data with true labels. The training data consist of both inputs (*attributes*) and

outputs (*labels*). The learning task is to learn a function (hypothesis) that can predict outputs for any valid and is evaluated using test data. To achieve this, the learner has to generalize from the presented data to unseen situations in a "reasonable" way. In our problem formulation, we want to learn a compact representation of the optimal MDP policy. Therefore, we set both training data and test data to be the policy table. The inputs (attributes) are the global states and the outputs (labels) are the actions taken for these states. The objective is to reduce the training/test error as much as possible. We use the decision tree supervised learning technique for this purpose.

A decision tree is a hierarchical model for representing the discrete function obtained by the learning process. It is composed of internal decision nodes and terminal leaves. Each decision node  $m$  implements a test function  $f_m(\mathbf{x})$  with discrete outcomes labeling the branches. Given an input, at each node, a test is applied and one of the branches is taken depending on the outcome. This process starts at the and is repeated recursively until a leaf node is hit, at which point the value written in the leaf constitutes the output [9,10].

Let  $DT$  be a decision tree and define the objective function to be the number of misclassification errors on the training/test data,

$$J(DT) = |\{(x, y) \in TS: DT(x) \neq y\}|,$$

Where  $\mathbf{x}$  is the vector of attributes for the inputs,  $y$  is the label for the output.  $TS$  is the training data set.

A supervised learning for decision trees computes a decision tree  $DT$  that minimizes  $J(DT)$ . Decision can be unpruned or pruned. The original tree that is obtained by the learning algorithm is unpruned. In order to reduce the number of the internal nodes on a decision tree, we can prune some internal nodes by an algorithm with takes a desired confidence level as input and produces a pruned tree while still keeping the error  $J(DT)$  relatively small.

#### a) Unpruned Decision Tree Supervised Learning

A learning algorithm for an unpruned decision tree supervised learning is shown in Algorithm 1.  $\{1, 2, \dots, X\}$  is the value set of attribute  $x_j$ . The function  $\text{Majority}(x_i)$  returns the label which is assigned to the majority of the attributes in the training set under the root  $x_i$ . We desire to find a decision tree  $DT$  that minimizes the error  $J(DT)$ .

$\text{ChooseBestAttribute}$  in Algorithm 1 is an important step to find the best decision tree within a reasonable time. A mutual information based method is given in Algorithm 2. The mutual information between attribute  $x_i$  and  $x_j$ ,  $I(x_i, x_j)$ , is calculated as

$$I(x_i, x_j) = H(x_i) - \sum_{a \in x_j} p(x_j = a) H(x_i | x_j = a)$$

Where  $H(x_i)$  is the entropy of attribute  $x_i$ .

---

#### Algorithm 1: $\text{GrowTree}(TS, x_i)$

---

1. **Inputs:** tree root  $x_i$ , and the training data set  $TS$
  2. **Output:**  $h$  that minimizes  $J(h)$
  3. **For**  $label \in \{1, 2, \dots, A\}$  **do**
  4.     **If**  $(y = label \text{ for all } < x, y > \in TS)$
  5.         **Return** new leaf(label)
  6.      $x_j = \text{ChooseBestAttribute}(TS)$
  7.     **For**  $a \in \{1, 2, \dots, X\}$  **do**
  8.          $S_a := \text{all } < x, y > \in S \text{ with } x_j = a$
  9.         **If**  $(S_a = \emptyset)$
  10.             **Return** new leaf(majority( $x_i$ ))
  11. **Return** new node( $x_j, \text{GrowTree}(S, x_{j1}), \text{GrowTree}(TS, x_{j2}), \dots, \text{GrowTree}(TS, x_{jX})$ )
- 

---

#### Algorithm 2: $\text{ChooseBestAttribute}(TS)$

---

1. **Inputs:** the training data set  $TS$
  2. **Output:**  $x_j$  so that  $\tilde{J}_j$  is minimized,
  3. **Choose**  $j$  to minimize  $\tilde{J}_j$ , computed as follows,
  4.     **For**  $v \in \{1, 2, \dots, X\}$  **do**
  5.          $TS_v := \text{all } < x, y > \in TS \text{ with } x_j = v$
  6.          $p_v := |TS_v|/|TS|$
  7.          $n_v := |TS_v|$
  8.          $n_{v,y} := \# \text{ of examples in } S_v \text{ with class } y$
  9.          $p_{v,y} := n_{v,y}/n_v$ , prob. of examples from class  $y$  in  $TS_v$
  10.          $H(y|x_j = v) := -\sum_y p_{v,y} \log p_{v,y}$
  11.      $\tilde{J}_j = \sum_{i=1}^X p_i H(y|x_j = i)$
  12. **Return**  $x_j$
- 

We use a simple example to show the calculation of  $I(x_i, x_j)$ . Let attribute  $x_j$  take two values,  $\{0, 1\}$ , with probability,  $p(x_j = 0) = 0.3$  and  $p(x_j = 1) = 0.7$ . The label  $y$  takes only two values,  $\{0, 1\}$ , as shown in Fig. 3. Considering 50 training examples, if the label  $y$  is classified by attribute  $x_i$ , 30 examples are classified correctly and 20 examples are misclassified. The entropy of  $T$  is given by,

$$\begin{aligned} H(x_i) &= - \sum_{i \in x_i} p(x_i = i) \log_2 p(x_i = i) \\ &= -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} = 0.9710 \end{aligned}$$

If we grow the decision tree by attribute  $x_j$ , the correctly-classified and misclassified examples for  $x_j = 0$  and  $x_j = 1$  are shown in Fig. 3. The conditional entropies in this case are

$$H(x_i | x_j = 0) = 0.9183 \text{ and } H(x_i | x_j = 1) = 0.8113$$

Then we can calculate  $I(x_i, x_j)$ :

$$\begin{aligned} I(x_i, x_j) &= 0.9710 - 0.3 \cdot 0.9183 - 0.7 \cdot 0.8113 \\ &= 0.1276 \end{aligned}$$

The task is to choose the attribute  $x_j$  that has the highest mutual information in order to minimize the

expected remaining uncertainty  $\tilde{J}_j$ .

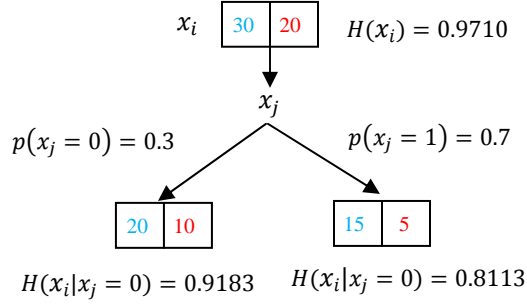


Fig. 3: The calculation of mutual information

### b) Pruned Decision Tree Supervised Learning

There are many ways to prune a decision tree. Here we use *pessimistic pruning* [11] of a decision tree. For example, given a case where 20 examples are correctly classified and 10 examples are misclassified, the error rate on training data is  $e=10/30 = 0.33$ . By using the normal approximation, the binomial confidence interval is,

$$e - z_{\alpha/2} \cdot \sqrt{\frac{e(1-e)}{30}} \leq e \leq e + z_{\alpha/2} \cdot \sqrt{\frac{e(1-e)}{30}}$$

The pessimistic method approximates the error with the upper bound of this interval. The pruning algorithm prunes a node if its pessimistic error is greater than the sum of pessimistic errors of all its children. Otherwise, it is not pruned.

## 5. Simulation Results

We first calculate the optimal policy offline by using a multi-agent MDP model as described in section 3. When the policy is available, we next approximate it by a decision tree supervised learning algorithm so that we can obtain a compact pattern of the policy. Both training data set and test data set are set to be the whole policy table.

For simplicity, we consider the 2-sensor system in our simulation. It is easy to extend to more than two sensor systems, which give the similar results. The parameters are as follows:  $T=20, E=10, H=10, A=10$ . We compare the unpruned decision tree and pruned decision tree with different confidence ( $C$ ) in term of error rate, shown in Fig. 4. The error rate is the percentage of incorrect predictions on test data as compared to the optimal decision policy calculated using the MDP. The results show that the unpruned decision tree and a pruned decision tree with high confidence ( $C=1$ ) give the best error rate, which is less than 1%. The higher the confidence factor, the less likely the algorithm is to prune a node. Thus when the confidence factor is lowered, the number of nodes is lessened, at the cost of increasing error. The tree size is

approximately 510 nodes, which is small enough to be stored on an embedded sensor. If we count each entry in the MDP optimal policy table as being equivalent to a node in a decision tree, the relative number of nodes in the optimal MDP policy table, unpruned decision tree, and pruned decision tree ( $C=1$ ) are shown in Fig. 5. The results show that the number of nodes required for representing the optimal policy reduces from 20001 to 500 when we employ the decision tree to learn the optimal policy table. The compression ratio is up to 39 in terms of the number of nodes. The reason is that decision tree model saves only the pattern of the policy by learning the whole policy while the sensing policy saves actions for all possible states. On the other hand, the maximum depth of the tree is about 14. Thus, the number of steps required to access the correct action from the policy table during execution increases from 1 (for directly accessing the full policy table) to 14. We believe that this increase in execution time is offset by the decrease in storage.

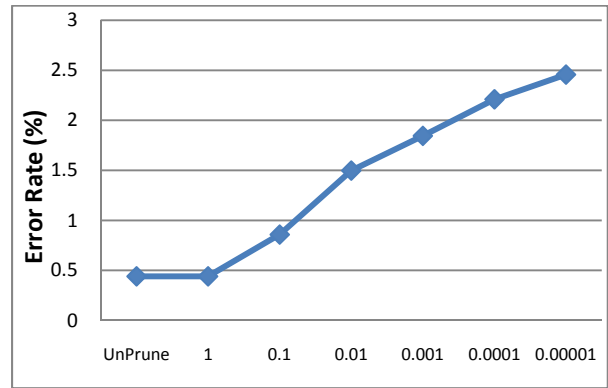


Fig. 4: Effect of confidence factor on error rate (decision tree)

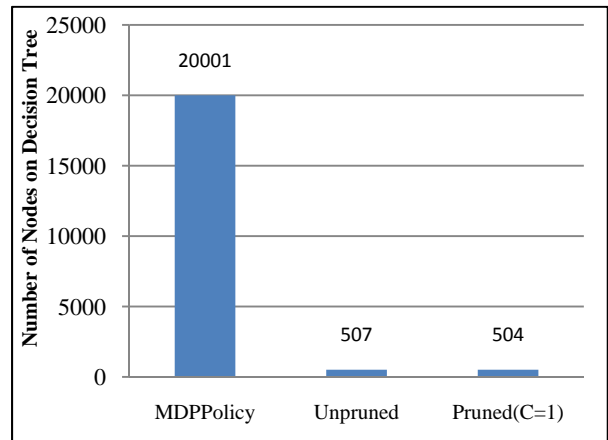


Fig. 5: The comparison of node number for the optimal policy table, unpruned and pruned ( $C=1$ ) decision tree

It is possible to achieve a zero error rate by using ensemble learning methods [12] on pruned decision

trees with low confidence ( $C=10^{-5}$ ) without increasing the classifier size by more than approximately 20% (Fig. 6). The idea of ensemble learning methods is to select a whole collection, or ensemble, of hypotheses from the hypothesis space and combine their predictions. The dramatic reduction in error rate is expected when using boosting meta-learners like AdaBoost [13], since decision trees have fairly low bias and fairly high variance. Boosting reduces the amount of error due to variance, thus decreasing the overall error. Fig. 6 shows the memory size for the optimal policy table, AdaBoost with low confidence pruned decision tree ( $C=10^{-5}$ ), high confidence pruned decision tree ( $C=1$ ), and the unpruned decision tree.

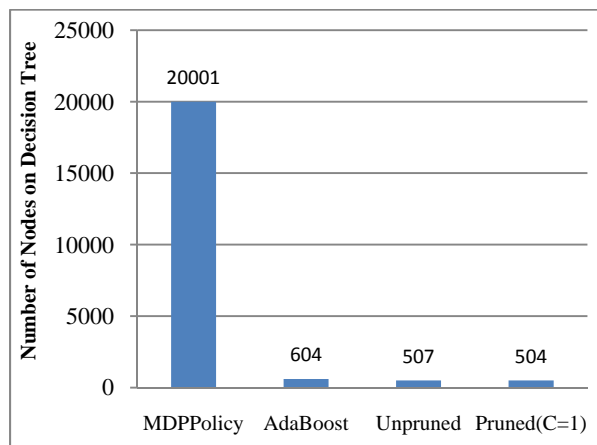


Fig. 6: Comparing the number of nodes in the optimal policy table, AdaBoost with low confidence pruned decision tree ( $C=10^{-5}$ , error rate=0), high confidence pruned ( $C=1$ ), decision tree (error rate =0.45%), and unpruned decision tree (error rate =0.45%)

## 6. Conclusion

We showed how a decision tree-based supervised learning technique can be used to calculate a compact representation of a global sampling policy. This sampling policy is to be used for coordinated sampling among sensors in a Body Area Network. This compact representation is suitable for storage in sensors with limited memory. The results suggest that high-confidence pruned decision trees and unpruned decision trees are best for deploying in a limited-memory system, while still achieving very low error rates (<1%). It is also possible to achieve a zero error rate by using ensemble learning methods on pruned decision trees with low confidence ( $C=10^{-5}$ ) without increasing the classifier size by more than about 20%. The large reduction in error rate is expected when using boosting meta-learners such as AdaBoost.

## 7. References

- [1] M. Venugopal, K. E. Feuvrel, D. Mongin, S. Bambot, M. Faupel, A. Panangadan, A. Talukder, and R. Pidva, "Clinical Evaluation of a Novel Interstitial Fluid Sensor System for Remote Continuous Alcohol Monitoring," *IEEE Sensors Journal*, vol. 8, pp. 71-80, 2008.
- [2] A. Panangadan, S. M. Ali, and A. Talukder, "Markov Decision Processes for Control of a Sensor Network-based Health Monitoring System," in *Proceedings of the Seventeenth Innovative Applications of Artificial Intelligence Conference Pittsburgh: AAAI Press*, Menlo Park, California, 2005, pp. 1529-1534.
- [3] S. Liu, A. Panangadan, A. Talukder and C. S. Raghavendra, "MDP Framework for Sensor Network Coordination," *The 8<sup>th</sup> ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN) Poster Session*, San Francisco, CA, US, 2009.
- [4] Y. Wang, B. Krishnamachari, Q. Zhao and M. Annavaram, "Markov-Optimal Sensing Policy for User State Estimation in Mobile Devices," *the 9<sup>th</sup> ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN10)*, Stockholm, Sweden.
- [5] C. S. Raghavendra, K. M. Sivalingam, and T. Znati, "Wireless Sensor Networks," Springer, 2006, pp. 3-107.
- [6] J. L. De Coe, P. Karger, D. Olmedilla and S. Zerr, "Policy Representation & Reasoning," presentation slide, Leibniz Hannover University, April 2008.
- [7] J. Marecki, T. Gupta, P. Varakantham and M. Tambe, "Not All Agents Are Equal: Scaling up Distributed POMDPs for Agent Networks," In *Proceedings of AAMAS*, May 2008, Estoril Portugal.
- [8] G. Tonti, J. M. Bradshaw, R. Jeffers, R. Montanari, N. Suri and A. Uszok, "Semantic Web languages for policy representation and reasoning: A comparison of KAoS, Rei, and Ponder," in *proceeding of the International Semantic Web Conference (ISWC 03)*. Sanibel Island, Florida 2003.
- [9] J. R. Quinlan, "Induction of Decision Trees," in *Journal of Machine Learning*, 1, 1996: 81--106.
- [10] E. Alpaydin, "Introduction to Machine Learning," *The MIT Press*, Cambridge, Massachusetts, London, Enland.
- [11] L. A. Breslow and D. W. Aha, "Simplifying Decision Trees: A Seurvey," *Knowl. Eng. Rev.*, 12(1), 1997: 1--40.
- [12] Y. Freund and R. E. Schapire, "A Short Introduction to Boosting," in *Journal of Japanese Society for Artificial Intelligence*, 14(5), 1999: 771--780.
- [13] R. Polikar, "Ensemble Based Systems in Decision Making," *IEEE Circuits and Systems magazine*, 6(3), 2006: 21--45.