Incremental Exemplar Learning Schemes for Classification on Embedded Devices

Ankur Jain, Daniel Nikovski

TR2008-062 September 2008

Abstract

Although memory-based classifiers offer robust classification performance, their widespread usage on embedded devices is hindered due to the device's limited memory resources. Moreover, embedded devices often operate in an environment where data exhibits evolutionary changes which entails frequent update of the in-memory training data. A viable option for dealing with the memory constraint is to use Exemplar Learning (EL) schemes that learn a small memory set (called the exemplar set) of high functional information that fits in memory. However, traditional EL schemes have several drawbacks which make them inapplicable for the embedded devices; (1) they have high memory overheads and are unable to handle incremental updates to the exemplar set, (2) they cannot be customized to obtain exemplar sets of any user-defined size that fits in the memory and (3) they learn exemplar sets based on local neighborhood structure that do not offer robust classification performance. In this paper, we propose two novel EL schemes, EBEL (Entropy-Based Exemplar Learning) and ABEL (AUC-Based Exemplar Learning) that overcome the aforementioned short-comings of traditional EL algorithms. We show that our schemes efficiently incorporate new training datasets while maintaining high quality exemplar sets of any user-defined size. We present a comprehensive experimental analysis showing excellent classification-accuracy versus memory-usage tradeoffs using our proposed methods.

European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases 2008, Antwerp, Belgium

Incremental Exemplar Learning Schemes for Classification on Embedded Devices

Ankur Jain Daniel Nikovski
Mitsubishi Electric Research Laboratory
201 Broadway, Cambridge, MA 02139
{jain,nikovski}@merl.com

June 23, 2008

Abstract

Although memory-based classifiers offer robust classification performance, their widespread usage on embedded devices is hindered due to the device's limited memory resources. Moreover, embedded devices often operate in an environment where data exhibits evolutionary changes which entails frequent update of the in-memory training data. A viable option for dealing with the memory constraint is to use Exemplar Learning (EL) schemes that learn a small memory set (called the *exemplar set*) of high functional information that fits in memory. However, traditional EL schemes have several drawbacks which make them inapplicable for the embedded devices; (1) they have high memory overheads and are unable to handle incremental updates to the exemplar set, (2) they cannot be customized to obtain exemplar sets of any user-defined size that fits in the memory and (3) they learn exemplar sets based on local neighborhood structures that do not offer robust classification performance. In this paper, we propose two novel EL schemes, EBEL (Entropy-Based Exemplar Learning) and ABEL (AUC-Based Exemplar Learning) that overcome the aforementioned short-comings of traditional EL algorithms. We show that our schemes efficiently incorporate new training datasets while maintaining high quality exemplar sets of any user-defined size. We present a comprehensive experimental analysis showing excellent classification-accuracy versus memory-usage tradeoffs using our proposed methods.

1 Introduction

Rapid advances in embedded computing [19] now facilitate significant computational capabilities in low-power embedded devices. Nodes in a sensor network are now expected to sense as well as *analyze* the data and make on-site intelligent decisions as opposed to simply pushing the data to a central server. Examples of such applications include fault detection in device condition monitoring data streams, face detection in video sensor networks, and intrusion detection at network routing hubs. In this paper, we focus on the data classification problem when the classifier operates on an embedded device. The computational and memory resources of such devices are not advanced enough to run sophisticated classifiers such as Support Vector Machines (SVM) or Neural Networks due to the computationally intensive optimization procedures and model learning steps involved. Moreover sensor data often exhibits evolutionary characteristics such as presence of a new kind of anomaly or a new class of network attacks, the training data for which are available after actual deployment either in batches or as a data stream. This entails incremental updates to the classifier in operation. Memory-based classifiers, especially density estimation based classifiers [10] are an excellent choice for such applications due to the practical benefits of ease of implementation, incremental nature and model free operation, and theoretical advantages of implicit smoothing, robustness to noise and automatic relevance weighting of the data. However, an embedded device is unlikely to be able to hold a large training data in memory (which could potentially keep increasing in size as new training data arrives). A viable option then is to employ exemplar learning (EL) techniques to find a training subset comprising a few carefully selected *exemplars* of high functional value that fit in memory and effectively delineate the class boundaries [1]. Sometimes, it is even acceptable to incur a small classification accuracy loss to conserve critical device resources. Since these devices are often expected to have a real-time performance, a small exemplar set also results in improved runtime performance. Effective EL schemes are thus essential for any memory-based classifier operating under resource constraints.

1.1 Exemplar learning on embedded devices

Exemplar learning (also known as case-based reasoning [16, 22]) is a combinatorial optimization problem similar to that of feature selection [13]. Due to the high complexity of the optimization procedure involved, it is often preferred to use randomized [15] or greedy methods [18, 3]. Over the years, EL solutions have been proposed using nearest neighbors [1], genetic algorithms [5], vector quantization [11] and density estimation based classification techniques [8].

EL research has mostly advanced in the context of nearest neighbors under the assumptions of Aha's algorithms [1] and more effective pruning approaches that operate on intricate neighborhood characteristics have been proposed [4]. Wilson et al. [18] provide a comprehensive survey of popular EL approaches in their work. They also proposed a suite of EL algorithms (DROP1-DROP5), which to the best of our knowledge are the most widely accepted approaches. In the rest of this section, we identify the issues with applying the DROP algorithms in an embedded device setting:

1. **Incremental Update:** DROP algorithms are computationally intensive, offline, and are not incremental in nature. They require the original training dataset in the memory throughout the execution of the exemplar learn-

ing procedure[1]. This makes these approaches inapplicable for embedded devices where memory is limited, and for applications where the training data is updated regularly.

2. **Ordered Removal:** Given an appropriate ranking of the training instances, the user can choose high-ranked instances that fit in the available memory. However, for all DROP schemes, the size of the exemplar set is determined at runtime and cannot be user defined. To the best of our knowledge, the ranking heuristic proposed by Xi et al. [20] is the best known approach that can be used to rank instances in DROP algorithms. However, as we will show in our experimental analysis, this scheme is adhoc and does not perform well.

3. **Robustness:** DROP algorithms learn exemplars based on the local neighborhood structure and make instance removal decisions based on the error rate performance. This approach is sensitive to noise and class-imbalance issues. We argue that using global methods such as density estimation and performance analysis based on ROC (Receiver Operating Characteristic) curves results in high-quality exemplar sets.

In this paper, we propose two novel incremental EL schemes based on the Parzen kernel density estimation classifier (PKDE) [10] that address the issues presented above. Our algorithms are motivated by the greedy selection scheme proposed by [3], where given a training dataset, we remove one instance at a time that is least likely to contribute towards the classification decision. Our proposed methods are as follows:

- EBEL (Entropy Based EL) – This method removes instances from the training set based on their *information content* as opposed to the error rate. Instead of using an adhoc ranking scheme, it removes a training instance whose removal causes the least amount of drop in the conditional entropy of the class indicator variable insuring minimum loss of information. EBEL is incremental, has low computational overheads and offers effective ordered removal.

- ABEL (AUC Based EL) – This method prunes data based on AUC (Area under ROC curve) performance. ABEL uses a *validation set* and prunes an instance if its removal offers the least drop in the AUC computed for this validation set. Though computationally more intensive than EBEL, we show that using the AUC's equivalence to the Wilcoxon-Mann Whitney statistic [21] and our proposed incremental method for updating the validation set scores, the runtime performance of ABEL can be reduced significantly.

While both the algorithms are incremental and suitable for an embedded device, ABEL is more demanding on computational resources. This is primarily

[1]DROP1 can be used in an incremental setting but it is known to have a significantly worse performance than other DROP algorithms

because of the overhead of maintaining the validation set in memory and repeated AUC computations. However, unlike EBEL, it offers robustness to class imbalance in the training data. As we will show in our experimental analysis, the two algorithms outperform traditional EL methods offering excellent tradeoff between runtime performance and exemplar set quality.

2 Incremental Exemplar Learning

We represent the original training set by $\mathcal{T}$ s.t. $|\mathcal{T}| = N$. An exemplar set with n instances is denoted as $\mathcal{S}_n$ s.t. $\mathcal{S}_n \subseteq \mathcal{T}$ where $\mathcal{S}_N = \mathcal{T}$. Given an exemplar set $\mathcal{S}_N$, our objective is to be able to incrementally compute smaller exemplar subsets $(\mathcal{S}_n|(1 \leq n < N) \wedge (S_n \subseteq S_{n+1}))$, by removing one instance at a time, such that the drop in classification accuracy in transitioning to $\mathcal{S}_n$ from $\mathcal{S}_{n+1}$ is as small as possible. Furthermore, unlike the DROP algorithms, this transition step should use information available from exemplar set $\mathcal{S}_{n+1}$ only and not from the original training set $\mathcal{T}$. This methodology naturally incorporates any new batch (or stream) of training data. When the system is in operation with exemplar set $\mathcal{S}_k$ of size k and a new training subset $\mathcal{T}_{new}$ (with possibly new target classes) arrives, we proceed with pruning the $\{\mathcal{S}_k \cup \mathcal{T}_{new}\}$ dataset, until a new exemplar subset $\mathcal{S}_k^{'}$ of operational size k is obtained. We present the details of using effective data structures that help us efficiently obtain $\mathcal{S}_n$ from $\mathcal{S}_{n+1}$ in Section 2.2.

2.1 The Parzen Kernel Density Estimation Classifier

PKDE is a well-known and one of the most widely accepted non-parametric density estimation technique. Given s samples $\{X_1, X_2, \cdots X_s\}$ drawn from a density function $f(x) \in \Re^1$, the Parzen density estimate is given by:

$$\hat{f}(x) = \frac{1}{s\lambda} \sum_{i=1}^{s} \kappa \left(\frac{x - X_i}{\lambda} \right), \tag{1}$$

where $\kappa(.)$ and λ are called the kernel function and the kernel bandwidth respectively. The kernel bandwidth is interpreted as a *smoothing parameter* and the kernel function as the *relevance weighting function*. A popular choice for the kernel function is the Gaussian kernel, $\kappa(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}||x||^2}$. If the kernel bandwidth is chosen as a function of the sample size s.t. $\lambda = \lambda(s)$, then the density estimates obtained using Eq. 1 are asymptotically unbiased when the following conditions hold true:

$$\lim_{s \to \infty} \lambda(s) = 0, \quad \text{and} \quad \int_{-\infty}^{\infty} \kappa(x) dx = 1. \tag{2}$$

A detailed analysis on the properties of the PKDE can be found in [14]. We now present our notations and their usage in PKDE based classification. For a given class c and training subset $\mathcal{S}_n$, we define set the $\mathbf{Z}_n^c$ as follows:

$$\mathbf{Z}_n^c = \{z | (z \in S_n) \wedge (class(z) = c)\}. \tag{3}$$

Given a subset $\mathcal{S}_n$, the density estimate of an instance x belonging to class c is given by:

$$f_c^n(x) = \frac{1}{|\mathbf{Z}_n^c|\lambda} \sum_{y \in \mathbf{Z}_n^c} \kappa \left(\frac{x - y}{\lambda} \right). \tag{4}$$

If the total number of classes is denoted by C, then the prior π_c^n of each class is:

$$\pi_c^n = \frac{|\mathbf{Z}_n^c|}{n}. \tag{5}$$

For an arbitrary instance x, we can now compute its posterior probability of belonging to a particular class c given the training dataset $\mathcal{S}_n$ as follows:

$$\begin{aligned} p(c|x)_{\mathcal{S}_n} &= \frac{\pi_c^n f_c(x)}{\sum_{j=1}^{C} \pi_j^n f_j(x)}, \\ &= \frac{\sum_{y \in \mathbf{Z}_n^c} \kappa \left(\frac{x-y}{\lambda} \right)}{\sum_{y \in S_n} \kappa \left(\frac{x-y}{\lambda} \right)}. \end{aligned} \tag{6}$$

For a binary classification problem, if the two classes are denoted by c^+ and c^-, then the merit-score[2] $\gamma_n(x)$ of an unknown instance x given the training set $\mathcal{S}_n$ can be obtained as follows:

$$\gamma_n(x) = p(c^+|x)_{\mathcal{S}_n} - p(c^-|x)_{\mathcal{S}_n}. \tag{7}$$

The merit-score values can then be used for classification accuracy analysis using ROC curves, AUC computations or classifier threshold selection [7].

2.2 EBEL: Entropy-Based Exemplar Learning

EBEL is a greedy algorithm motivated by the leave-one-out cross validation (LOOCV) scheme. For each instance $x \in \mathcal{S}_n$, EBEL computes the conditional entropy of the class output variable using $\mathcal{S}_n \setminus \{x\}$ as the training set and x as a

[2]In the rest of this paper, it is assumed that a high merit-score is associated with a greater chance of an instance belonging to the positive class c^+.

testing instance. It then chooses to discard the instance whose removal results in the least conditional entropy loss. If $\mathcal{C}$ denotes the class output variable, $p(x)$ the prior on a training instance x and $H(\mathcal{C}|x)$ the entropy of the class variable conditioned on the presence of instance x then EBEL prunes instances according to the following rule:

$$\mathcal{S}_{n-1} = \mathcal{S}_n \setminus \{\underset{x \in \mathcal{S}_n}{\text{argmin}}[H(\mathcal{C}|x)]\} \text{ where,} \tag{8}$$
$$H(\mathcal{C}|x) = -p(x) \sum_{c=1}^{C} p(c|x)_{\mathcal{S}_n \setminus \{x\}} \log(p(c|x)_{\mathcal{S}_n \setminus \{x\}}).$$

Assuming that the training data is independent and identically distributed (i.i.d.), the prior $p(x)$ on an instant x is a constant and can be ignored while making the conditional entropy computations.

The intuition behind EBEL's pruning rule comes from Fano's inequality [17] which shows that the classification error of a classifier $\mathcal{G}$ is lower bounded by the conditional entropy as follows:

$$P(\mathcal{G}(x) \neq \mathcal{C}) \geq \frac{H(\mathcal{C}|x) - \log(2)}{\log(C-1)}. \tag{9}$$

Hence, choosing the subset with least conditional entropy is least likely to reduce the classification accuracy.

We are now faced with the challenge of finding the most suitable candidate for pruning in an incremental and a computationally efficient manner. We show that by storing only the sum of the pairwise kernel values of the training data (instead of the complete $N \times N$ pairwise kernel-value matrix), each outgoing instance can be found in time linear to the size of the exemplar subset. Given set $\mathcal{S}_n$, we store the sum of the kernel values of each instance $x_j \in \mathcal{S}_n$ with all other instances of its own class as follows:

$$\nu^n[c,j] = \sum_{x \in \mathbf{Z}_n^c \wedge x \neq x_j} \kappa\left(\frac{x - x_j}{\lambda}\right). \tag{10}$$

The conditional entropy of the class output variable with each instance $x_j \in \mathcal{S}_n$ can then be obtained as:

$$H(C|x_j) = -\sum_{c=1}^{C} \nu^n[c,j] \log(\nu^n[c,j]). \tag{11}$$

If instance y is chosen for removal using the pruning rule of Eq. 8, then the ν-matrix can be updated as follows:

$$\nu^{n-1}[class(y), j] = \nu^{n}[class(y), j] - \kappa\left(\frac{x_j - y}{\lambda}\right). \tag{12}$$

As we have already shown in Eq. 6, the contents of the ν-matrix are sufficient to compute the conditional probabilities and hence the conditional entropy values in Eq. 8.

So far, we have ignored the effect of sample size (*i.e.*, exemplar set size) on kernel bandwidth. Given the original training set $\mathcal{T}$, the kernel bandwidth can be obtained using popular bandwidth optimization schemes, such as LOOCV [10] that minimizes mean square error (MSE). However, as we have shown in Eq. 2, the kernel bandwidth is not independent of the sample size, which in our case is continuously changing. Since, performing the LOOCV after each pruning step is computationally infeasible, we need to further investigate the relationship of the kernel bandwidth to the sample size.

If the sample's true density $\hat{f}(x)$ is continuous in the r^{th} order and does not change with the sample size, then the optimal bandwidth $\lambda(n)$, that minimizes the MSE is related to the sample size n as follows [14]:

$$\lambda(n) = \frac{\hat{f}(x)\displaystyle\int_{-\infty}^{\infty}\kappa^2(y)dy}{(2nrk_r f^r(x))^{\frac{1}{2r+1}}}, \tag{13}$$

where k_r is called the characteristic component of the kernel function $\kappa(.)$. Fortunately, most of the terms in the above equation are independent of the sample size, and for a typical case of $r = 2$, we can obtain the optimal bandwidth for a sample of size n as:

$$\lambda(n) = \lambda(N)\left(\frac{N}{n}\right)^{0.2}, \tag{14}$$

where $\lambda(N)$ is the bandwidth obtained using the training dataset of size N, that minimized the MSE. Although this prevents a significant computational effort of repeated bandwidth optimization with changing sample size, a bandwidth update still entails the recomputation of the ν-matrix in Eq. 12. Since, the bandwidth varies slowly with the sample size as $n^{-0.2}$, we update the ν-matrix periodically when $N_{last}/n > \frac{\sqrt{N_{last}}}{\alpha}$, where N_{last} was the size of the sample when the last bandwidth update occurred and α is user-specified sensitivity parameter. This formulation insures that the runtime complexity of the incremental steps in EBEL is always linear in the size of the training set being pruned. Thus, ignoring the initial bandwidth optimization and ν-matrix computation costs, the runtime complexity of EBEL for obtaining an exemplar set of size k from a training set of size N is $O((N-k)\alpha^2 N)$.

Algorithm 1 outlines the details of the EBEL procedure. When the system is already using set $\mathcal{T}_{old}$ for memory-based classification and is updated with

Algorithm 1 EBEL

Inputs: Existing Training dataset: $\mathcal{T}_{old}$, New Training dataset: $\mathcal{T}_{new}$, Desired size of training subset: k, Number of classes: C, Kernel function: κ, bandwidth optimized for $\mathcal{T}_{old}$: $\hat{\lambda}$, Bandwidth update sensitivity parameter: α
Outputs: Desired Training subset: $\mathcal{S}$ $s.t.$ $|\mathcal{S}| = k$

```
1: $\mathcal{S} \leftarrow \mathcal{T}_{old} \cup \mathcal{T}_{new}, N \leftarrow |\mathcal{T}_{old}| + |\mathcal{T}_{new}|, \lambda \leftarrow \hat{\lambda}\left(\frac{|\mathcal{T}_{old}|}{N}\right)^{0.2}$
2: Initialize $\nu$-matrix $s.t.$, $\nu[c,q] \leftarrow \sum_{(x\in\mathcal{S})\wedge(class(x)=c)\wedge(x\neq x_q)} \kappa\left(\frac{x_q - x}{\lambda}\right)$
3: while $|\mathcal{S}| > k$ do
4:    $\mathcal{I} \leftarrow \underset{x_i\in S}{\text{argmin}} -1 * \sum_{c=1}^{C} \frac{\nu[c,i]}{\sum_{c=1}^{C}\nu[c,i]} \log\left(\frac{\nu[c,i]}{\sum_{c=1}^{C}\nu[c,i]}\right)$ {$x_{\mathcal{I}}$'s removal offers least entropy drop}
5:    for $i \leftarrow (j|x_j \in \mathcal{S})$ do
6:       $\nu[class(x_{\mathcal{I}}), i] \leftarrow \nu[class(x_{\mathcal{I}}), i] - \kappa\left(\frac{x_i - x_{\mathcal{I}}}{\lambda}\right)$ {Update $\nu$-matrix}
7:    end for
8:    $\mathcal{S} \leftarrow \mathcal{S} \setminus \{x_{\mathcal{I}}\}$ {Remove $x_I$ from $\mathcal{S}$}
9:    if $\frac{N}{|\mathcal{S}|} > \frac{\sqrt{N}}{\alpha}$ then
10:      $\lambda \leftarrow \hat{\lambda}\left(\frac{N}{|\mathcal{S}|}\right)^{0.2}$, $N \leftarrow |\mathcal{S}|$ {Update bandwidth}
11:      goto Step 2 {Recompute $\nu$-matrix}
12:   end if
13: end while
```

a new batch of training data $\mathcal{T}_{new}$, Algorithm 1 finds an exemplar set $\mathcal{S}$ s.t., $|\mathcal{S}| = k$. The size k of the desired exemplar set is provided as an input parameter. The bandwidth value λ corresponding to $\mathcal{T}_{old}$ is assumed to have been optimally obtained offline. Note that the same algorithm can be used to obtain the exemplar set of size k from the original training dataset $\mathcal{T}$ by setting $\mathcal{T}_{new} = \emptyset$ and $\mathcal{T}_{old} = \mathcal{T}$. The algorithm first combines training samples from both the new and the old training data and then updates the corresponding kernel bandwidth accordingly (Step 1). After computing the ν-matrix, it prunes set $\mathcal{S}$ until its size reaches the desired value of k. The instances are pruned one at a time (Steps 4-8) and the ν-matrix is updated if the sample size becomes smaller than that allowed by the user parameter α (Steps 9-12). The resulting exemplar set $\mathcal{S}$, replaces the older training set $\mathcal{T}_{old}$ and is used for classifying the future unlabeled instances.

2.3 ABEL: AUC-Based Exemplar Learning

Although EBEL is an efficient EL scheme, it has certain disadvantages that it shares with the traditional EL methods: (1) it ignores the class population ratios while making the instance removal decisions. This problem is critical to incremental exemplar learning where removal of each instance or addition of new training datasets continuously skews the class populations; (2) an instance plays the dual role of a training instance (as long as it is not pruned) as well as that of a testing instance (while computing conditional probabilities in Eq.8). Hence, at different stages of the pruning, the performance is validated (*i.e.*, the entropy is computed) against datasets differing in composition and size, reducing the statistical significance of the results.

In this section, we propose the ABEL algorithm, that addresses the aforementioned issues. ABEL ignores the dependencies within the training data samples and operates by monitoring the AUC (Area under ROC curve) performance of the unpruned training data computed over a consistent *validation set.* The validation set is extracted from the training data at the beginning of the pruning procedure, and the instances are then removed such that the AUC performance drop as a result of the removal is minimized. Given a training set $\mathcal{S}_n$, and a validation set $\mathcal{V}$ such that $\{\mathcal{V} \cap \mathcal{S}_n\} = \emptyset$, ABEL prunes an instance x according to the following rule:

$$\mathcal{S}_{n-1} = \mathcal{S}_n \setminus \{\underset{x \in S_n}{\operatorname{argmax}}[\mathcal{A}_{\mathcal{G}}(\mathcal{V}, \mathcal{S}_n \setminus \{x\})]\}. \tag{15}$$

where, $\mathcal{A}_{\mathcal{G}}(X, Y)$ denotes the AUC value using testing dataset X, training dataset Y when classifier $\mathcal{G}$ is used to score the testing instances. ABEL provides a simple yet effective EL framework that generalizes to any classifier. However, given $\mathcal{S}_n$ each pruning step comes at the increased computational cost of n AUC computations and the memory overhead for holding the validation set. In the rest of this section, we first discuss how the complexity of any AUC computation can be reduced using the Wilcoxon-Mann-Whitney statistic [21] followed by a discussion on how the update scheme presented in Section 2.2 can be used to enhance the runtime performance of ABEL when used with a PKDE classifier.

Although the ROC is considered to be a continuous curve, in practice it turns out to be a step function [7]. Hence, *given* the merit scores of the instances in the validation set, the AUC can be computed directly (without actually generating the ROC curve) in $O(|\mathcal{V}|^2)$ time. However, for a finite set of instances, it can be shown that the AUC (denoted by $\mathcal{A}$) is equal to the normalized Wilcoxon-Mann-Whitney (WMW) statistic [9]:

$$\mathcal{A} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{p} 1_{(\gamma_i^+ > \gamma_j^-)}}{np}, \tag{16}$$

where $\Gamma^+ = \{\gamma_1^+, \gamma_2^+, \cdots, \gamma_p^+\}$ are the merit-scores of the positive class instances and $\Gamma^- = \{\gamma_1^-, \gamma_2^-, \cdots, \gamma_n^-\}$ are the merit-scores of the negative class instances in $\mathcal{V}$. ($|\mathcal{V}| = n + p$).

Statistic $\mathcal{A}$ is the estimator of $P(\Gamma^+ > \Gamma^-)$ (*i.e.*, the probability that all positive class instances get a higher merit-score than the negative class instances) and if all the merit scores are sorted in nondecreasing order s.t. δ_i is the rank

Algorithm 2 ABEL

Inputs: Existing Training dataset: $\mathcal{T}_{old}$, New Training dataset: $\mathcal{T}_{new}$, Desired size of training subset: k, Classifier to use: $\mathcal{G}$, Validation-set's fractional size: η
Outputs: Desired Training subset: $\mathcal{S}$ *s.t.* $|\mathcal{S}| = k$

```
1: $\mathcal{S} \leftarrow \mathcal{T}_{old} \cup \mathcal{T}_{new}$
2: Initialize $\mathcal{V}$ $s.t.$ $(\mathcal{V} \subseteq \mathcal{S}) \wedge (|\mathcal{V}| = \eta * |\mathcal{S}|)$ {Extract the validation set}
3: $\mathcal{S} \leftarrow \mathcal{S} \setminus \mathcal{V}$ {Ensure $\mathcal{S} \cap \mathcal{V} = \emptyset$}
4: while $|S| > k$ do
5:     $\mathcal{S} \leftarrow \mathcal{S} \setminus \mathrm{argmax}_{x_i \in \mathcal{S}} \mathcal{A}_{\mathcal{G}}(\mathcal{V}, \mathcal{S} \setminus \{x_i\})$ {Remove instance of least AUC loss}
6: end while
```

of γ_i^+, then $\mathcal{A}$ can be further simplified to [21]:

$$\mathcal{A} = \frac{1}{np}\left(\sum_{i=1}^{p} \delta_i - \frac{p(p+1)}{2}\right). \tag{17}$$

Eq. 17 shows that given the merit-scores of the validation set instances, the AUC value can be computed in $O(|\mathcal{V}|\log(|\mathcal{V}|))$ time. However, obtaining the merit-scores necessitates classification of all the instances in the validation set. If $O(\mathcal{G})$ denotes the time taken by a classifier $\mathcal{G}$ to classify a testing instance, then the runtime complexity of removing one instance using ABEL is $O(|\mathcal{V}| * (\log(|\mathcal{V}|) + \mathcal{G}))$.

In Algorithm 2, we have outlined the ABEL procedure which is similar to Algorithm 1 in terms of input and output parameters. However, it prunes instances using a more robust methodology than EBEL and requires an additional parameter η that determines the size of the validation set. The algorithm first extracts the validation set from the training data, given its fractional size η (Step 2) and then uses only the remaining instances as the training samples (Step 3). The optimal value of η depends on the memory constraints and the complexity of the classification problem. While a large value of η results in a high memory overhead, a small value reduces statistical significance of the AUC-scores of the validation set elements resulting in worse instance removal decisions. In our experimental analysis, we found that setting $\eta = 0.1$ offered a reasonably good tradeoff between the two extremes for a wide variety of problems.

Though ABEL provides an effective EL framework that generalizes to any classifier it suffers from high computational complexity. Its runtime performance can be significantly improved when used with the PKDE classifier. As we have already shown in Section 2.2, the PKDE merit-scores of the validation set elements can be incrementally updated in $O(\alpha^2|\mathcal{V}|)$ time. Since each AUC computation takes $O(|\mathcal{V}|\log(|\mathcal{V}|))$ time, by setting $|\mathcal{V}| = \eta N$, the runtime complexity of ABEL for obtaining an exemplar set of size k from a training set of size N is $\mathrm{O}(N(N-k)(\eta\log(\eta N) + \alpha^2))$.

3 Experimental Evaluation

We compare the performance of our proposed schemes against the popular DROP1 and DROP2 algorithms proposed by Wilson et al., [18] coupled with the ranking scheme proposed by Xi et al., [20]. While DROP1 is incremental in nature and can be used in the embedded device setting that we are interested in, DROP2 (including its other variants DROP3-DROP5) requires access to the entire training data during its EL stages and is inapplicable in an incremental setting. We include it in our experimental evaluation to have a comprehensive analysis and to benchmark our proposed schemes against the state of the art.

We conducted several experiments on a variety of machine learning datasets from the UCI [2] and UCR [12] machine learning repositories. In this paper, we report critical results on the following datasets:

1. Magic Gamma [2]– This dataset was generated by the registration of high energy gamma particles in a ground-based atmospheric Cherenkov gamma telescope using the imaging technique. The dataset has 10 attributes and $5,000$ instances. The classification task was to distinguish images caused by primary gamma rays from those caused by other cosmic rays.

2. Vehicle Silhouette [2] – This dataset consists of vehicle silhouettes (represented using 18 features) of four types of vehicles (Opel, Sara, Bus or Van). In our experimental setup, the classification task was to distinguish the "Bus" class from the other vehicle classes. The dataset has 846 instances.

3. Face(all) [12] – This dataset contains $1,690$ instances from 14 classes that correspond to human faces (from both genders) as a 131 point time series. In our experimental setup, the classification task was to distinguish the "Face 1" class from the other face classes.

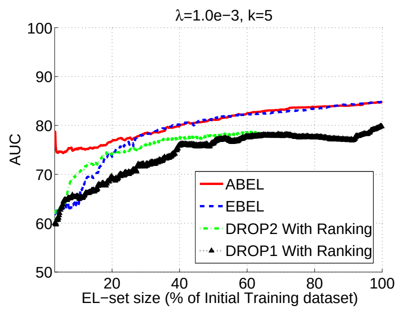
4. Statlog [2] – The dataset consists of the multi-spectral values of pixels in a satellite image of a landscape. The classification task is to use the spectral values to predict the soil type in the landscape image. The dataset has six different classes with $6,435$ instances and 36 features. In our experiments, the objective was to distinguish the "Grey damp soil" class from the other classes.

Having multiclass datasets help us robustly evaluate the performance of the proposed EL schemes in an incremental setting, where data belonging to a particular class may cease to exist in the exemplar set as a result of pruning, or new classes appear as a result of training data updates. We compared classification accuracy on the ROC curve using the AUC metric [6]. For all our experiments, the initial kernel bandwidth for PKDE and the neighborhood size for the DROP algorithms were obtained offline using LOOCV technique. Throughout our experiments, we have used $\alpha = 2$ and $\eta = 0.1$ unless stated otherwise. The validation set for ABEL was always randomly selected from the initial training set. In the rest of this section, we denote the initial training dataset by $\mathcal{S}_N$

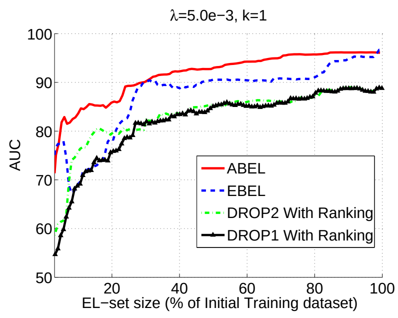
(e.g., for the Vehicle dataset $N = 846$) and the exemplar sets learned from this set as $\mathcal{S}_n$.

Figure 1 shows the classification performance using our proposed schemes as we vary the desired exemplar-set size for different datasets. The results were aggregated over a 5-fold cross validation scheme. The x-axis shows the exemplar-set size ($|\mathcal{S}_n|$) as the percentage of the original training dataset ($|\mathcal{S}_N|$) and the y-axis represents the AUC-value obtained using $\mathcal{S}_n$ as the training data for the classifier. The size of the validation set was kept fixed at $|\mathcal{V}| = \eta * |\mathcal{S}_N|$ for all the datasets. As seen in the figure, reducing the value of $|\mathcal{S}_n|$ results in classification accuracy drop. Although the individual classification performance on different datasets is different, for high values of $|\mathcal{S}_n|$, the change in the classification accuracy for all EL methods is insignificant. However, the classification accuracy drops rapidly when the required exemplar-set size is significantly small (*i.e.*, less than 10%). As seen in Figures 1(c)-1(d), both ABEL and EBEL perform significantly better than the incremental DROP1 algorithm. The non-incremental DROP2 scheme shows a better performance with small exemplar set sizes on Magic Gamma (Figure 1(a)) and the Vehicle datasets (Figure 1(b)). Since DROP2 uses the entire training dataset for making an instance removal decision as opposed to using only the exemplar set data, the effect of an instance removal is validated against a significantly larger dataset resulting in better performance. However, the performance of DROP2 is always worse than that of ABEL even though it used only a small percentage ($\eta = 0.1$) of the training data for validation. The performance of DROP algorithms is especially worse on the Face(all) data with DROP1 becoming unstable at low memory conditions dropping to AUC values of as low as 50 (Figure 1(c)).

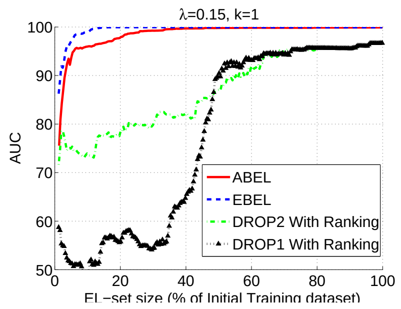
In order to verify that the performance enhancements obtained using our methods were primarily due to a better EL scheme that generalizes well to other classifiers, we evaluated the classification performance with the resulting exemplar sets using other classifiers as well. In Figure 2, we show the results obtained using the popular SVM classification algorithm. We present the entire ROC curves obtained using exemplar sets with 90% instances removed from the original dataset ($|\mathcal{S}_n| = 0.1|\mathcal{S}_N|$). The exemplar sets were obtained under the experimental conditions used in Figure 1. For each fold of the cross validation step, we first applied the four EL schemes to remove 90% of the training instances. The data in the resulting exemplar set was then used to train the SVM with the Radial Basis Function (RBF) kernel. The final ROC curve was obtained after vertical averaging [7] of the individual ROC curves from each fold. While the individual performance on different datasets using different EL methods is of less importance, Figure 2 should be used to compare the *relative* performance of the EL methods. For example, the ROC curve of EBEL for the Face(all) dataset (Figure 2(c)) shows a better classification accuracy than that of ABEL, a trend also observed earlier in Figure 1(c). This trend in the relative performance for all datasets is similar to that seen in Figure 1, with ABEL and EBEL outperforming the DROP algorithms. The results shown in Figures 2(a)-2(d) indicate that our EL schemes provide high quality exemplar sets that generalize to other classifiers equally well.



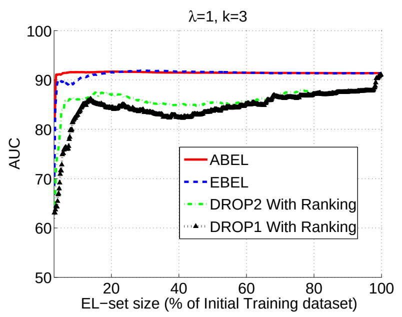
(a) Magic Gamma



(b) Vehicle



(c) Face(all)



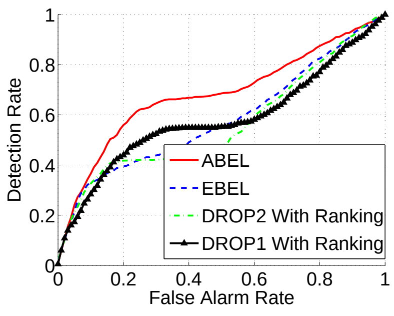
(d) Statlog

Figure 1: AUC performance against exemplar set size.

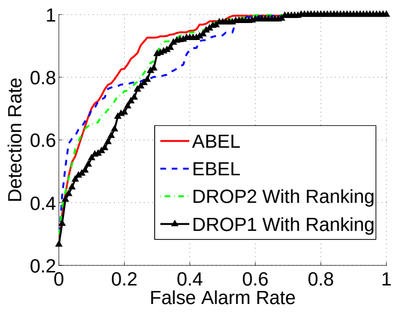
EL Method	Memory Overhead	CPU Usage			
		Face(all) x=180μsec	Vehicle x=59μsec	Magic Gamma x=$2,600\mu sec$	Statlog x=$2,200\mu sec$
DROP1	$O(n)$	x	x	2.2x	x
DROP2	$O(N+n)$	1.6x	1.8x	4.1x	1.8x
EBEL	$O(n)$	5.6x	6.1x	x	x
ABEL	$O(\eta N)$	252x	141x	126x	125x

Table 1: Computational resource demand for different EL schemes.

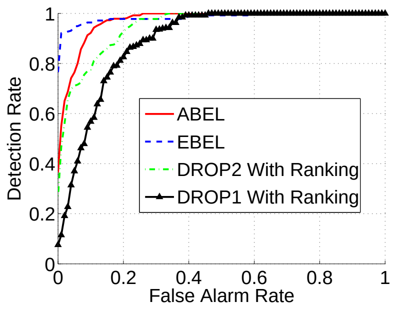
We now analyze our proposed EL methods in terms of their computational resource demand. While DROP1 and EBEL methods do not store any data in the memory other than the exemplar set being used for classification (or in the process of being pruned), the DROP2 algorithm incurs the overhead of maintaining the entire training data ($\mathcal{S}_N$) and ABEL the overhead of maintaining the validation set ($\mathcal{V}$). While EBEL and ABEL will incur the overhead of maintaining the ν-matrix in memory for their efficient execution, the DROP algorithm maintains a kd-tree structure for efficient neighborhood lookups. In Table 1, we have summarized the overall memory overhead (other than holding the exem-



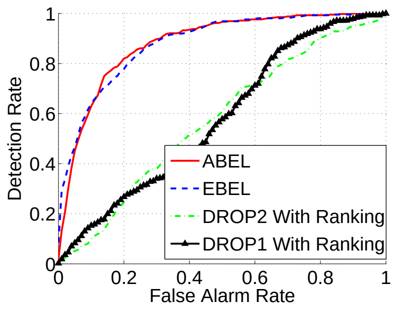
(a) Magic Gamma, $|\mathcal{S}_n| = 397$



(b) Vehicle, $|\mathcal{S}_n| = 68$



(c) Face(all), $|\mathcal{S}_n| = 138$

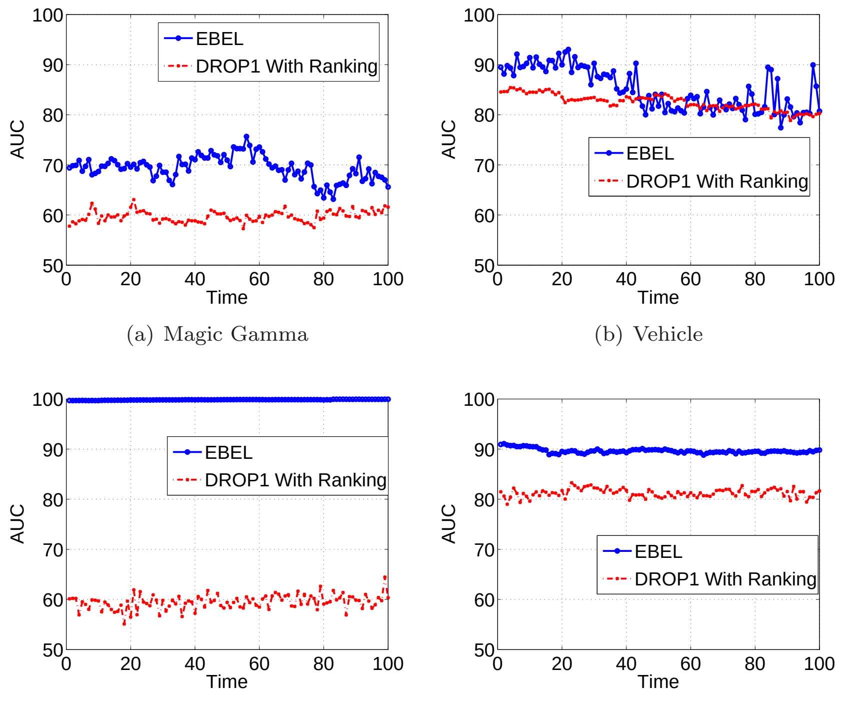


(d) Statlog, $|\mathcal{S}_n| = 355$

Figure 2: Classification performance using the SVM classifier after 90% instance removal.

plar set S_n) and the mean CPU time taken to prune *one* training instance from the training dataset. The results were obtained for the experiments shown in Figure 1[3]. We show the memory overheads in terms of the original training data size ($|\mathcal{S}_N|$) and the exemplar-set sizes ($|\mathcal{S}_n|$). We compare the CPU efficiency of different algorithms on a relative scale where 'x' represents the most efficient algorithm. The individual values of 'x' are shown in Table 1. The CPU usage shown does not include offline overheads of parameter learning, initial kd-tree construction and ν-matrix computations, however data structure update costs are accounted for. We observe that DROP1 and DROP2 are more efficient for small problems (both Face(all) and Vehicle datasets have less than 1,700 instances). However, for larger problems (Magic Gamma and Statlog) EBEL is the most efficient algorithm. Due to the overhead of the AUC computations, ABEL is significantly more CPU intensive than the other methods, and is thus well suited in an offline EL framework. However, EBEL is incremental, more efficient than DROP algorithms and offer robust pruning results thus making it an excellent choice for realtime systems.

[3]All algorithms were implemented in C and the experiments were conducted on an Intel Core Duo, with 1.66 GHz processor and 2 Gigabytes of memory.



(a) Magic Gamma (b) Vehicle

(c) Face(all) (d) Statlog

Figure 3: AUC performance in online setting.

We compare the performance of DROP1 and EBEL in an online setting in Figure 3. For each fold of the cross-validation step, we split the training data such that one half of it is first used to obtain an EL set of $|\mathcal{S}_n| = 250$ samples, and then the other half is used as a training data stream. The training stream updates the EL set with new training instances and the EL algorithm is used again to compute a new exemplar set of 250 samples. We created a training stream of 100 such updates and the AUC performance is calculated after each such update. Figures 3(a)-3(d) show the mean AUC performance against time on the x-axis (each time unit corresponds to one training stream update). As seen in the figures, EBEL consistently outperforms the DROP1 algorithm. Note that new classes may appear in the training stream (for example the Face(all) and the Statlog datasets have 31 and 7 classes respectively) varying the individual performances however it does not affect relative AUC performance enhancements obtained using EBEL. The classification performance of EBEL on the Face(all) dataset (Figure 3(c)) is observed to be consistently near 100% accuracy.

4 Summary and Future Work

In this paper, we have presented two novel exemplar learning schemes EBEL and ABEL. EBEL learns exemplars from an information-theoretic perspective while

ABEL uses a validation set and prunes instances based on the AUC performance. ABEL is computationally more intensive than EBEL and hence more suitable in an offline EL framework like the popular DROP2 scheme. However, ABEL returns high quality exemplar sets as compared to DROP2 with a significantly lower memory overhead. Using a comprehensive experimental analysis we have shown that our proposed methods consistently outperform the popular DROP algorithms and the exemplar sets obtained using our methods generalize well to other classifiers such as SVM. In the future, we plan to extend our work in the following directions:

1. Investigate on approximate AUC computation methods to reduce its runtime complexity further.

2. Avoid recomputation of the ν-matrix by developing efficient update schemes. This would also remove the user parameter α from the system and improve the quality of pruning results.

3. Develop efficient methods to incorporate class imbalance awareness in EBEL.

References

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

[2] A. Asuncion and D. Newman. UCI machine learning repository. (http://www.ics.uci.edu/~mlearn/mlrepository.html), 2007.

[3] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *Neural Networks, IEEE Transactions on*, 5(4):537–550, Jul 1994.

[4] H. Brighton and C. Mellish. Advances in instance selection for instance-based learning algorithms. *Data Mining Knowledge Discovery*, 6(2):153–172, 2002.

[5] J. R. Cano, F. Herrera, and M. Lozano. Using evolutionary algorithms as instance selection for data reduction in kdd: An experimental study. *Evolutionary Computation, IEEE Transactions on*, 7(6):561–575, Dec. 2003.

[6] C. Cortes and M. Mohri. AUC optimization vs. error rate minimization. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2004.

[7] T. Fawcett. ROC graphs: Notes and practical considerations for researchers. Technical Report HPL-2003-4, HP Labs, Palo Alto, CA, 2004.

[8] K. Fukunaga and R. R. Hayes. The reduced Parzen classifier. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 11(4):423–425, 1989.

[9] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, April 1982.

[10] T. Hastie, R. Tibshirani, and J. Friedman, editors. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Series in Statistics, 2001.

[11] N. Jankowski and M. Grochowski. Instances selection algorithms in the conjunction with lvq. In *Artificial Intelligence and Applications*, pages 703–708, 2005.

[12] E. Keogh and T. Folias. The UCR time series data mining archive, 2002. Riverside CA. University of California - Computer Science & Engineering Department.

[13] N. Kwak and C.-H. Choi. Input feature selection by mutual information based on Parzen window. *Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1667–1671, Dec. 2002.

[14] E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[15] E. Pkalska, R. P. W. Duin, and P. Paclík. Prototype selection for dissimilarity-based classifiers. *Pattern Recogn.*, 39(2):189–208, 2006.

[16] B. Smyth and E. McKenna. Building compact competent case-bases. In *ICCBR'99: Proceedings of the Third International Conference on Case-Based Reasoning and Development*, pages 329–342, London, UK, 1999. Springer-Verlag.

[17] A. Thomasian. Review of 'transmission of information, a statistical theory of communications' (Fano, R. M.; 1961). *Information Theory, IEEE Tran. on*, 8(1):68–69, Jan. 1962.

[18] D. R. Wilson and T. R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38(3):257–286, 2000.

[19] W. Wolf. What is embedded computing? *Computer*, 35(1):136–137, Jan. 2002.

[20] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana. Fast time series classification using numerosity reduction. In *ICML'06: Proceedings of the 23rd International Conference on Machine learning*, pages 1033–1040, New York, NY, USA, 2006. ACM.

[21] L. Yan, R. Dodier, M. C. Mozer, and R. Wolniewicz. Optimizing classifier performance via approximation to the Wilcoxon-Mann-Witney statistic. In *Proceedings of the 20th International Conference on Machine Learning*, pages 848–855, Menlo Park, CA, USA, 2003. AAAI Press.

[22] J. Zhu and Q. Yang. Remembering to add: Competence-preserving case-addition policies for case base maintenance. In *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 234–241, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.