THESIS

CLASSIFICATION ENSEMBLE METHODS FOR MITIGATING CONCEPT DRIFT WITHIN ONLINE DATA STREAMS

Submitted by

Michael J. Barber

Department of Computer Science

In partial fulfillment of the requirements

for the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer 2012

Master's Committee:

Advisor: Adele E. Howe

Charles Anderson

Jennifer Hoeting

ABSTRACT

CLASSIFICATION ENSEMBLE METHODS FOR MITIGATING CONCEPT DRIFT WITHIN ONLINE DATA STREAMS

The task of instance classification within very large data streams is challenged by both the overwhelming amount of data, and a phenomenon known as *concept drift*. In this research we provide a comprehensive comparison of several state of the art ensemble methods that purport to handle concept drift, and we propose two additional algorithms. Our two new methods, the AMPE and AMPE2 algorithms are then used to further our understanding of concept drift and the algorithmic factors that influence the performance of ensemble based concept drift algorithms.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Data streams consist of a sequence of data that can arrive over very long periods of time; possibly unbounded, most likely transient, and possibly sporadic. Example data streams can be thought of as complex as statistical process control within manufacturing processes, sensor network measurements, and social networking streams, or as simple as one's incoming email stream to their inbox. Data stream processing must handle an infinite amount of incoming samples, and do so efficiently so as to not lag behind the availability of incoming data. As one can see, data stream processing is then restricted by space because of its voluminous data, and it is restricted in performance because of the continuous nature of an ever increasing supply of data.

Today's software applications are distributed, networked, and produce more data than we have ever seen, had to handle, or yet alone make sense of before. Software applications can now generate data at rates and volumes so great that we cannot store it all, or ever get a complete picture of the entire data set. Data stream classification and machine learning techniques are current popular methods used for data stream analytics and classification.

The continuous nature of data streams presents two distinct challenges that previous static machine learning classification techniques have not had to address. The first is the performance requirement to classify samples as efficiently as possible so as to not disrupt the continuous classification process. And the other challenge is dealing with an anomaly known as *concept drift*. Concept drift is a condition caused when classification methods process on-line examples that have somehow changed from the original class distribution the classifier was built with [13, 19, 14, 11]. Prior literature identifies two types of concept drift: 1.) *sudden* (abrupt, instantaneous) concept drift, and 2.) *gradual* concept drift [16]. Sudden concept drift is characterized by large amounts of change between the underlying class distribution and the incoming example in a relatively short amount of time. Gradual concept drift can require a very large amount of time to see a significant change in differences between the underlying class distribution and the incoming examples. An example of concept drift that most of us face within our daily lives can be found within the classification of email messages as spam by our email clients. The individuals producing spam change the underlying representation of their emails in an effort to bypass the spam filter's current model. The underlying class distribution of spam has changed, and ideally to prevent spam, the filter should adapt to the new instances.

Tsymbal's comprehensive taxonomy[16] of concept drift, and the systems designed to lessen its effect, describe three types of approaches to handling concept drift: 1.) Instance Selection, 2.) Instance Weighting, and 3.) Ensemble Learning. The focus of our research is ensemble learning techniques, and their different approaches to concept drift abatement within very large data streams. The goals of our research effort are two-fold. We first want to provide a more comprehensive comparison of several state of the art ensemble methods that have been proposed in the literature. And our secondary research goal is to leverage what we learn of concept drift and the existing ensemble techniques we examine to design two new ensemble method approaches.

The existing ensemble techniques we assess are Wang et al.'s Accuracy Weighted Ensemble (AWE)[18], Masud et al.'s Multi-Partition Multi-Chunk Ensemble (MPC)[12], and Brzeziński and Stefanowski's Accuracy Updated Ensemble (AUE) [5]. These methods were of interest to our re-

search because of their lineage, as well as possible augmentation. All three methods are ensemble methods that operate on the data stream in portions, or chunks. Both the MPC and AUE methods are derivatives of AWE: the MPC method augmented the AWE technique by introducing multiple partitions of chunks, and the AUE method modified the AWE weighting function such that the AUE ensemble maintained more classifiers than AWE. The MPC and AUE methods were both shown to outperform its AWE predecessor. Though both methods exhibited performance gains over the AWE technique, their comparisons were performed independently of one another, and with very different experimental approaches. Our primary research contribution within this thesis is then to provide a more comprehensive comparison of the three ensemble techniques, and to do so within an experimental environment that borrows experimental procedures and elements from the prior studies.

Our second research contribution is the proposal of two new ensemble methods designed to further our understanding of concept drift and the algorithmic factors that influence performance when concept drift is determined to be present. We propose that if we can detect concept drift, we can adapt to it, and we can reduce its effects upon classifier accuracy. Our first contributed method, our Adaptive Multi-Partition Ensemble (AMPE), augments the MPC ensemble method with a concept drift detection mechanism and additional logic to dynamically change the size of partitions used for training the ensemble classifiers. The AMPE method detects concept drift within the data stream and increases the size of partitions used to train; the intention is that additional and more rigorous training will compensate for the accuracy penalty associated with the underlying drifting class distribution.

Our second contributed method, our AMPE2 method, was designed to help us appraise the importance of when to increase partition sizes. Whereas our AMPE method increases the partition used to train the classifiers when concept drift is detected, our AMPE2 method adjusts partitions when concept drift is not present.

1.1 Thesis Structure

This thesis is arranged by firstly describing the existing classification ensemble algorithms, and our contributed methods in Chapter 2. We describe the existing methods of Wang et al.'s AWE method, Masud et al.'s MPC technique, and Brzeziński and Stefanowski's AUE algorithm. We then provide descriptions of our AMPE and AMPE2 contributed methods, and what we hope to learn from them. In Chapter 3 we provide the motivations for and design of our experiments. We provide several hypotheses, and describe the performance metrics we used to examine our research questions. Our experimental design describes how we leveraged portions of empirical techniques from Wang et al.'s AWE examinations, and data sets/experimental framework from Brzeziński and Stefanowski's previous comparative studies. Chapter 4 then provides the results of our experiments, and we begin to see evidence to either support or discount our hypotheses. Chapter 5 concludes by analyzing our results in more detail to show whether our hypotheses are supported, or discounted as a result of our experiments. After addressing the research questions we pose in this thesis, we then present additional research questions and potential experiments that may address them.

1.2 Acknowledgments

The author would like to thank his advisor Dr. Adele Howe for all of her support, and motivation. I have childhood comic book heroes whose feats now seem trite in comparison to what I have witnessed Adele overcome this past year. I would like to also extend gratitude to Dr. Chuck Anderson for his guidance and direction. I would also like to thank the previous authors whose shoulders I now stand atop. Both Mr. Brzeziński and Dr. Masud provided us with their ensemble implementations, and answered numerous questions. I thank them kindly for their knowledge and their willingness to share it so that we all may improve the research field. Finally, and most importantly, I would like to thank my wife Barbara for all of her support, encouragement, and her unshakable belief in me.

Chapter 2

Methods

Our experimental approach evaluates several state of the art ensemble methods for handling concept drift, and within this chapter we examine their implementations in detail. We first describe Wang et al.'s Accuracy Weighted Ensemble (AWE); as it was the basis for defining chunk-based ensemble techniques for concept drift mitigation. Brzeziński and Stefanowski's Accuracy Updated Ensemble (AUE), and Masud et al.'s Multi-Partition Multi-Chunk Ensemble (MPC) are both derivatives of AWE, and we examine them next. We then describe a drift detection mechanism, and how it can be employed to dynamically adjust partition sizes within our Adaptive Multi-Partition Ensemble (AMPE). Finally, we describe an alternative chunk adaptation algorithm (AMPE2) that uses data chunks stored while the system is stable to train with when concept drift is encountered.

2.1 Existing State of the Art Ensemble Methods

For us to carry out a fair comparison of the AWE, MPC, and AUE methods we needed their implementations, and we are ever grateful to Dr. Masud and Mr. Brzezinski for providing them to us. The MPC implementation we use within our studies was modified to work within the MOA framework[18], and both the AWE and AUE methods are algorithms that are currently implemented by MOA. As we too try to advance the research field, like that of the prior works of Dr. Masud, Mr. Brzezinski, and Mr. Stefanowski, we are reminded that we can only do so with the cooperation, and contributions of our peers before us.

2.1.1 Accuracy Weighted Ensemble (AWE)

The Accuracy Weighted Ensemble (AWE) was developed by Wang et al. in 2003 for handling concept drift within very large data streams. Prior to Wang et al.'s AWE contribution, several ensemble methods averaged outputs of classifiers within the ensemble [15]. Wang et al. state that the implication of averaging classifier output within the ensemble leads to significant variance when applying models learned earlier to the current classification context. This variance increases classification error, and they formally prove Theorem 1 using Tumer's bias-variance decomposition method [17]; that a weighted ensemble approach E_k built from k most recent data chunks is capable of less error than a single classifier G_k that is also comprised of the k most recent data chunks. The expected error reduction can be summarized by their following theorem[18]:

Theorem 1. *E_k produces a smaller classification error than G_k, if classifiers in E_k are weighted by their expected classification accuracy on the test data.*

A significant challenge to achieving the error reduction that Theorem 1 proposes is in how to calculate the weights. Theorem 1 is dependent upon being able to give each classifier C_i a weight that is inversely proportional to the expected error in classifying examples, which is dependent upon the function being learned, which is unavailable. Instead, Wang et al. propose deriving the weights of the classifiers by estimation of the expected prediction error on test examples. Their assumption is that the class distribution of the most recent data chunk, S_n, is closest to the class distribution of the current test data, and they compute a weighting from the classification error of C_i on the most recent data chunk, S_n. The mean square error of classifier C_i can then be estimated using Equation 2.1 where $f_c^i(x)$ is the probability given by C_i that x is a given class of c. The mean square error of classifier C_i is then:

$$MSE_i = \frac{1}{|S_n|} \sum_{(x,c) \in S} (1 - f_c^i(x))^2 \tag{2.1}$$

Wang et al. also calculate the error of a random classifier, MSE_r, in an effort to discard classifiers that do not contribute to the ensemble. This error rate of a random classifier is then used to discard classifiers whose errors are equal to or larger than MSE_r The mean square error of a

random classifier is then:

$$MSE_r = \sum_c p(c)(1 - p(c))^2 \tag{2.2}$$

The weighting function for classifier C_i then becomes:

$$w_i = MSE_r - MSE_i \tag{2.3}$$

Because data stream mining consists of an infinite length of incoming instances to classify, it is impossible to maintain all of the classifiers over time. Instead, the AWE algorithm maintains the top k classifiers within its ensemble. The full AWE psuedocode can be found within Algorithm 1. Wang et al. evaluated their AWE approach by comparing their ensemble method to a single

Algorithm 1 Accuracy Weighted Ensemble (AWE) Algorithm [18]

Input:
S : data stream of examples
K : number of ensemble members
C : a set of K previously trained classifiers
Output:
C : a set of K classifiers with updated weights

train classifier C' from S
compute error rate/benefits of C' via cross validation on S
derive weight w' for C' using 2.3
for each classifier $C_i \in C$ **do**
  apply C_i on S to derive MSE_i
  compute w_i based on 2.3
end for
$C \leftarrow K$ of the top weighted classifiers in $C \cup C'$
return C

classifier approach. Their results showed their AWE ensemble method to have an average 20% decrease in classifier error over the single classifier approach for the two data sets used within their experiments. The AWE approach was also shown to be more efficient than a single classifier approach while training. The AWE experimental results show that AWE appears to require about a quarter of the time a single classifier approach requires for the training phase. These improvements over a single classifier approach appear to have influenced the research community, and several chunk based methods that are AWE derivatives have recently emerged as state of the art. And we examine them a bit closer next.

2.1.2 Multi-Partition Multi-Chunk Ensemble (MPC)

The Multi-Partition Multi-Chunk Ensemble (MPC) was developed by Masud et al. in [12]. Masud et al. attribute the inspiration for their design to Wang et al.'s AWE implementation, but distinguish key differences between AWE and MPC. Those key differences are: 1.) The MPC implementation divides the training data into multiple partitions for multiple classifiers(v), 2.) Each MPC classifier is trained with a user configurable amount (r) of data chunks instead of AWE's one, 3.) The MPC implementation represents knowledge spread across multiple partitions in the chunk space versus the AWE implementation that stores all knowledge within individual chunks, and 4.) The MPC implementation relies on simple voting between classifiers, in contrast to weighted voting that the AWE implementation uses.

Similar to AWE, the MPC ensemble approach maintains an ensemble $A = \{A_1, A_2, \cdots, A_{K*v}\}$ of the best performing classifiers. This ensemble is then updated from classifiers that have been trained using the most recent data chunks to arrive on the data stream. Unlike AWE, the MPC method divides the data chunks into v equal partitions that are then used to train a new set of potential classifiers for ensemble membership. MPC partitions the data stream by the following: Let D_n represent the last data chunk to arrive on the data stream. Compute the error of each classifier $A_i \in A$ on D_n. Let $D = \{D_{n-r+1}, \cdots, D_n\}$, or r of the most recent data chunks, including D_n. Divide D randomly into v equal partitions: $\{d_1, \cdots, d_v\}$ and train a new set of potential classifiers by iterating over the partitions and computing the relative error of the classifier. Lastly, update the ensemble membership with the best k performing classifiers. The MPC psuedocode can be found within Algorithm 2.

Algorithm 2 Multi-Partition Multi-Chunk Ensemble (MPC) Algorithm [12]

Input:
$\{D_{n-r+1}, \ldots, D_n\}$: most recent r data chunks
K: number of classifiers in ensemble
A: set of previously trained K classifiers
v : number of partitions to maintain
Output:
A': set of updated K classifiers

for each classifier $A_i \in A$ **do**
  test A_i on D_n & compute its expected error
end for
Let S = $\{D_{n-r+1} \cup D_n\}$
Divide S into v equal disjoint partitions $\{d_1, d_2, \ldots, d_v\}$
for j = 1 to v **do**
  $A_j^n \leftarrow$ train classifier with training data $D - d_j$
  test A_j^n on test data d_j & compute expected error
end for
$A' \leftarrow$ best K * v classifiers from $A^n \cup A$
return A'

Masud et al. theoretically demonstrate that an ensemble method consisting of multiple partitions can achieve an error reduction over single partition methods relative to the number of partitions v, and data chunksize r; while maintaining an efficiency that is at most $r * v$ times the running time of the single partition approach; or the AWE technique. The MPC method was compared to Wang et al.'s AWE technique in [18]. This experiment showed the MPC algorithm to have up to a 10% error reduction over the AWE technique while maintaining total running times that were within $r * v$ times that of the AWE running times. This error reduction is dependent upon satisfying inequality 2.4 which is:

$$\frac{(v-1)(1+\rho_d)^{r-1}}{rv} \leq 1 \text{ or } E_R \leq 1 \tag{2.4}$$

where E_R is then the ratio of Multi-partition Multi-chunk ensemble errors to that of a Single-partition Multi-chunk ensemble error rate, and ρ_d is defined as the magnitude of concept drift, or the maximum error introduced to a classifier as a result of concept drift.

Masud et al. also empirically corroborate their theoretical findings that a multiple partition

approach is more accurate than a single partition approach, or the AWE approach. Masud et al. demonstrated the performance gain a multiple partition approach has over AWE by using an experimental methodology that was similar to what Wang et al. utilized within their AWE experiments. Admiring this consistency in experimental evaluations, we too try to maintain an experimental methodology that is as consistent with both the MPC experiments and the AWE experiments. Our experimental approach which is detailed in Chapter 3 will show how we constructed our experimental trials using a similar experimental approach to that of the AWE and MPC experiments. We also demonstrate how we used the data sets and experimental framework used within the experiments of another chunk based ensemble method: the Accuracy Updated Ensemble (AUE)[5], and we examine it next.

2.1.3 Accuracy Updated Ensemble (AUE)

Brzeziński and Stefanowski's Accuracy Updated Ensemble (AUE)[5], like that of MPC, is also derived from the Accuracy Weighted Ensemble (AWE). Distinctions that Brzeziński and Stefanowski make of AUE to that of AWE are that the AWE implementation was designed for batch based learners versus incremental or online learners, and the weighting function of AWE, Equation 2.2, inadvertently removes classifiers they term too "risky" in rapidly changing data streams. The implications Brzeziński and Stefanowski draw from this are: 1.) AWE performance is dependent upon the size of the data chunks, and 2.) AWE's weighting mechanism hinders classification performance while data streams are in high flux, or where concept drift is more prevalent.

It is not apparent how AUE's implementation addresses 1.) above, but Brzeziński and Stefanowski propose the use of a new weighting function, Equation 2.5, that will prevent the algorithm from removing too many classifiers from the ensemble membership. AUE's updated weighting function is then:

$$w_i = \frac{1}{MSE_i + \epsilon} \tag{2.5}$$

where MSE_i is calculated similarly to Equation 2.1, and ϵ is said to be "a very small constant value" for those rare situations when MSE_i is zero. Brzeziński and Stefanowski state that "AUE not only selects classifiers, but also updates them according to the current distribution." This

suggests that AUE will maintain a larger set of classifiers that are updated versus removed. The full psuedocode for the AUE method is presented within Algorithm 3.

Brzeziński and Stefanowski empirically show how AUE performs as well, if not better than AWE,

Algorithm 3 Accuracy Updated Ensemble (AUE) Algorithm [5]

Input:
S : data stream of examples
k : number of ensemble members
Output:
ε : ensemble of k online classifiers with updated weights

$C \leftarrow 0$ *C: set of stored classifiers*
for all data chunks $x_i \in S$ **do**
  train classifier C' on x_i
  compute MSE of C' via cross validation on x_i
  derive weight w' for C' using 2.5
  for all classifiers $C_i \in C$ **do**
    apply C_i on x_i to derive MSE_i
    compute weight w_i based on 2.5
  end for
  $\varepsilon \leftarrow k$ of the top weighted classifiers in $C \cup C'$
  $C \leftarrow C \cup C'$
  for all classifiers $C_e \in \varepsilon$ **do**
    if $w_e > \frac{1}{MSE_r}$ and $C_e \notin C'$ **then**
      update classifier C_e with x_i
    end if
  end for
end for

on several data sets within their experiments. The AUE method was shown to be more accurate than the AWE method on all but one of the data sets Brzeziński and Stefanowski used within their studies. The results of examining the AUE method also show it to be significantly less efficient than the AWE technique for all of the data sets used within their experiments.

Brzeziński and Stefanowski show that both the training and testing times of the AUE method require significantly more time than the AWE method. For some data sets the training times AUE requires was shown to be as high as 5 to 10 times that of the AWE method. Where the prior examinations of the AWE and MPC techniques measure only training times, the AUE study examines both training and testing times, as well as the impact the methods have on system memory consumption. The AUE experimental results show it to also require significantly more memory than the AWE technique, and at times the AUE method was shown to require 3 to 6 times the amount of resources the AWE method required.

The experimental methodology used by Brzeziński and Stefanowski in their comparisons was significantly different than what was used by Wang et al. and Masud et al. in their previous comparative studies. An example difference in experimental methodologies between Brzeziński and Stefanowski's AUE experiments compared to that of the Wang et al. and Masud et al. is that while Wang et al. and Masud et al. manipulate both the size of chunks and the number of classifiers within the ensemble population, Brzeziński and Stefanowski do not. In addition, where Wang et al. and Masud et al. appear to measure ensemble efficiency in terms of only training times, Brzeziński and Stefanowski measure both the training and testing times, and also the memory consumption of the methods. The differences within experimental methodologies not only makes it difficult to determine which of the three ensemble methods performs best, but as we will discuss later in Chapter 3, it also shows how a more complete comparison of AUE to that of AWE complements the lineage of ensemble methods that address concept drift. But before we delve further into empirical methods and results, we introduce our contributed ensemble methods: the Adaptive Multi-Partition Ensemble (AMPE), and the AMPE2 algorithm.

2.2 Contributed Methods

All of these algorithms (AWE, MPC, and AUE) operate on static hyperparameters (chunksize, partitions, etc.) that were determined in prior trials versus the actual characteristics of the data stream when concept drift occurs. This observation lead us to examine the MPC method further and question if the number of partitions used within training a classifier could be dynamically adjusted as classification error increased, or concept drift became more prevalent. Both our AMPE and AMPE2 algorithms then adjust partition sizes dependent upon the detection of concept drift in the data stream.

2.2.1 Adaptive Multi-Partition Ensemble (AMPE)

Our Adaptive Multi-Partition Ensemble (AMPE) method is inspired by the want to make an ensemble approach to concept drifting data streams more robust. But a significant challenge to mitigating concept drift is detecting when, or if, concept drift has even occurred. If concept drift can be detected, one might be able to make changes to reduce its influence. In our case, we would like to make training the classifiers within the ensemble more robust. We would like to exploit the knowledge from a drift detection technique that concept drift is imminent, and we should attempt to prepare for it, hoping to accommodate the changes ahead within the class distribution.

One such drift detection method is the Early Drift Detection Method (EDDM) of Baena-García et al.[1]. EDDM model uses a binomial distribution to calculate the average distance between two consecutive classification errors (p'_i) and its standard deviation (s'_i) to determine two thresholds:

- **Warning level α:**

  Past this level, begin accumulating examples in anticipation of concept drift.

- **Drift level β:**

  Past this level, concept drift is determined to be true. Reset model and learn new model from accumulated examples stored.

The EDDM technique then uses these warning levels to accumulate examples in anticipation of concept drift, and once the concept drift is determined to be true, it adjusts its underlying model representation to accommodate the drift. Our interest within EDDM is to use its ability to detect concept drift levels and use those levels to then adjust the number of partitions needed to train a new set of classifiers. In [1], Baena-García et al. set both α and β to be 0.95 or $\approx 2\sigma$ and 0.90 or $\approx 1.645\sigma$ within their empirical studies and their source implementation. They state they determined these values to be optimal from earlier experiments they performed, but no additional detail about how they were derived is provided. Our study leverages their previous findings for these parameters, but exactly *how* they were determined by Baena-García et al. is unknown at this time.

Our proposed approach, the Adaptive Multi-Partion Ensemble (AMPE), builds upon the successes of Masud et al.'s MPC technique by adapting the number of partitions to use within the ensemble by using EDDM to detect concept drift. Similar to MPC, we maintain an ensemble of the best $K * v$ classifiers, and as each data chunk arrives, we test it, and determine its expected error. Unlike MPC, we then determine the drift threshold using Garcia et al.'s EDDM technique on the incoming data chunk. If the data chunk's EDDM threshold is above α, but yet under β, the chunk is considered to be drifting, and we increase the number of partitions to train with. Once concept drift stabilizes within the data stream, we reset the partition size to be the default(2) to mimic MPC while not in concept drift. Figure 2.2.1 is a visual representation of how our AMPE method adjusts partition memberships based upon the EDDM thresholds. Algorithm 4 provides

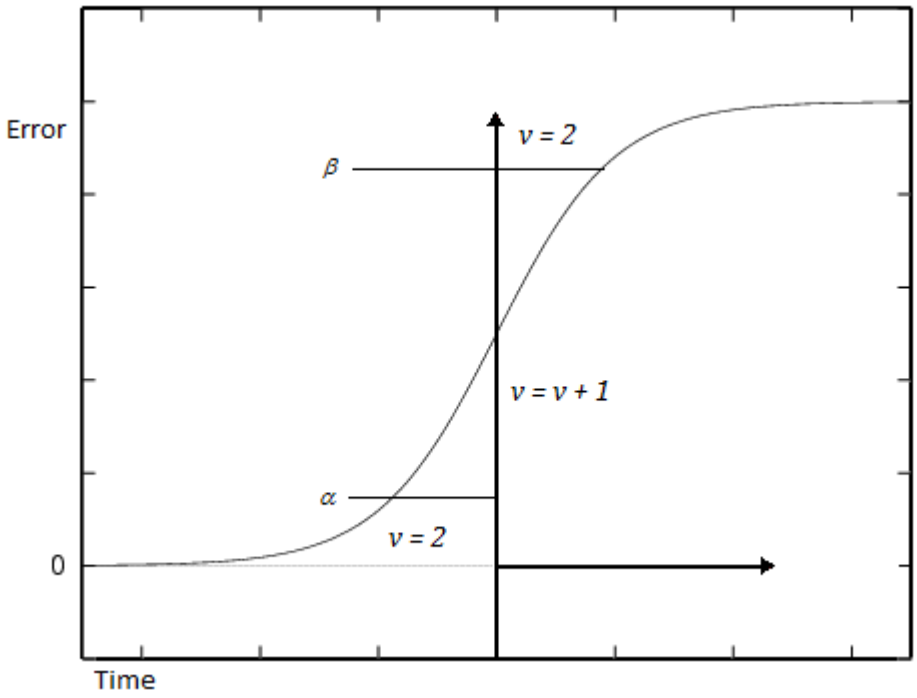

Figure 2.1: AMPE Partition Adaptation and EDDM thresholds

an overview of the adaptive partitioning approach.

AMPE sets the size of the partition relative to the amount of drift seen within the data chunk segments. Similar to MPC, we then randomly divide the data stream $\{D_{n-r+1} \cup D_n\}$ and train v classifiers. Each A_j^n classifier within the ensemble is trained using training data $D - d_j$ and tested independently. The expected error is determined, and the best $K * v$ classifiers are used to build the new ensemble elements. Masud et al. show as v is increased, so will the running time. A key difference to note here is our AMPE implementation dynamically adjusts v during concept drift, and remains constant similarly to how MPC works. This allows AMPE to remain more or less constant as v is only increased during instability within the underlying class distribution.

Data stream processing is conceptually unbounded in size, and it is quite plausible concept drift can happen over a very long strata. We recognize that the number of partitions (v) is dependent upon the size of concept drift, and Masud et al. show how both the number of chunks r and v are dependent upon the characteristics of the drift[12]. Deciding not to leave v unbounded for our

Algorithm 4 Adaptive Multi-Partition Ensemble (AMPE) Algorithm

Input:
$\{D_{n-r+1}, \ldots, D_n\}$: most recent r data chunks
K: number of classifiers in ensemble
A: set of previously trained K classifiers
v : minimum number of partitions to maintain
Output:
A': set of updated K classifiers
v': number of partitions to maintain

for each classifier $A_i \in A$ **do**
  test A_i on D_n & compute its expected error
  calculate threshold τ using EDDM(D_n)
  if $\tau \geq \alpha$ **and** $\tau \leq \beta$ **then**
    $v \leftarrow Max((v+1), 100)$
  else
    $v \leftarrow 2$
  end if
end for
Let S = $\{D_{n-r+1} \cup D_n\}$
Divide S into v equal disjoint partitions $\{d_1, d_2, \ldots, d_v\}$
for j = 1 to v **do**
  $A_j^n \leftarrow$ train classifier with training data $D - d_j$
  test A_j^n on test data d_j & compute expected error
end for
$A' \leftarrow$ best K * v classifiers from $A^n \cup A$
$v' \leftarrow v$
return A', v'

studies, we placed a ceiling condition for v not to exceed 100 for our experiments.

2.2.2 Adaptive Multi-Partition Ensemble2 (AMPE2)

Our motivation for AMPE was in questioning whether modifying the partition sizes of the MPC algorithm, in association with the error rate, would increase ensemble performance. By increasing the partition membership of AMPE while the error rate was in flux, or concept drift was prevalent, we favored training the classifiers with data nearest to where concept drift occurs in the data stream. Our AMPE2 algorithm was motivated by questioning the importance of whether training with more instances nearest concept drift is better than training the ensemble classifiers with data stream instances that are further from where concept drift is found, instead of where the system is fairly quiescent, or stable. Whereas our AMPE method is intended to be a viable improvement over the existing state of the art ensemble techniques, our AMPE2 method is expected to perform worse and is designed to study the influences of increasing training partitions with differing data instance selections.

The AMPE2 algorithm is identical to AMPE with the exception of how it operates on the EDDM α and β thresholds. AMPE2 places more emphasis on training with data instances collected when the system is not in concept drift, in contrast to how the AMPE method increases training partition size when concept drift has been detected. AMPE2's psuedocode is outlined within Algorithm 5. AMPE2 places more importance on classifiers within the ensemble during times in the system that

Algorithm 5 Adaptive Multi-Partition Ensemble2 (AMPE2) Algorithm

Input:
$\{Dn - r + 1, \ldots, D_n\}$: most recent r data chunks
K: number of classifiers in ensemble
A: set of previously trained K classifiers
v : minimum number of partitions to maintain
Output:
A': set of updated K classifiers
v': number of partitions to maintain

for each classifier $A_i \in A$ **do**
  test A_i on D_n & compute its expected error
  calculate threshold τ using EDDM(D_n)
  if $\tau \leq \alpha$ **and** $\tau \geq \beta$ **then**
    $v \leftarrow Max((v + 1), 100)$
  else
    $v \leftarrow 2$
  end if
end for
Let S = $\{Dn - r + 1 \cup D_n\}$
Divide S into v equal disjoint partitions $\{d_1, d_2, \ldots, d_v\}$
for j = 1 to v **do**
  $A_j^n \leftarrow$ train classifier with training data $D - d_j$
  test A_j^n on test data d_j & compute expected error
end for
$A' \leftarrow$ best K * v classifiers from $A^n \cup A$
$v' \leftarrow v$
return A', v'

are not experiencing concept drift. It does this by increasing the partitions used to train and test the classifiers with while the error rates of the ensemble are outside of the EDDM thresholds. AMPE2 is an example algorithm modification that aims to answer questions about concept drift and how we might accommodate changes within the underlying class distribution better.

Chapter 3

Evaluation

The experimental methodologies used within the comparative experiments of MPC to AWE and that of AUE to AWE are significantly different, yet both MPC and AUE were independently shown to outperform AWE. The study between the MPC and AWE methods was performed using one real data set, and one synthetically generated data set. The real data set Masud et al. used was botnet traffic collected over a week's network traffic and consisted of $30,000$ records. Similar to the original AWE study, the MPC comparison used the Rotating Hyperplane[10] generator for generating $250,000$ synthetic examples within a data stream. Also similarly to the AWE experiments, the MPC trials utilized Weka [20] for its baseline algorithms - J48 decision tree, Ripper, and a Bayesian Network. Both the original AWE and MPC studies identified the size of chunk (chunksize) and number of classifiers within the ensemble (ensemblesize) as independent variables that influence classifier performance. Both studies account for their influence by manipulating chunksizes and ensemblesizes while studying the influence of the other on classifier performance. For example, both AWE and MPC trials manipulate chunksize selections that range from 250 to 2000 while examining the influence ensemblesize has on performance. The AWE and MPC studies also examined chunksize selection influence on classifier performance by manipulating ensemblesizes ranging from 2 to 16 classifiers within the ensemble membership.

The comparison study between the AUE method and AWE technique did not identify chunksize and ensemblesize as independent variables within their experimental procedure. Instead, the

AUE study selected a chunksize, $d_r = 500$ for the real data set trials, and a chunksize, $d_a = 1000$ for the artificial data set trials [5]. The AUE comparison also fixed the ensemblesize selection to 15. Brzeziński and Stefanowski appear to anchor the chunksize and ensemblesize selections within their experiments, but their experimental trials also appear to examine the ensemble methods more thoroughly by using much larger and many more data sets than the prior AWE and MPC studies.

The AUE and AWE comparative study utilized the Massive Online Analysis (MOA)[2] framework, and its associated data set generators. Similar to the Weka framework, the MOA framework provides a standard platform for examining machine learning algorithms. But unlike the Weka toolset, the MOA framework is specifically designed to study concept drift in very large, or massive data sets. The MOA framework afforded Brzeziński and Stefanowski to not only use more data sets than the prior AWE and MPC studies, but it also allowed studying the algorithms with much larger streams of data in the data sets.

The AUE and AWE comparison used three real data sets, and four synthetically generated data sets. The real data sets used within the AUE comparisons were the Australian Electricity data set [9], the Ozone detection data set [21], and the 2^{nd} annual KDD Cup challenge Donation data set [7]. Similarly to the AWE and MPC experiments, the AUE and AWE comparison also used the Rotating Hyperplane generator for one of its data sets. The other three synthetic data sets Brzeziński and Stefanowski used within their experiments were the LED data set [4], the Waveform generator data set [8], and the SEA data set[15]. Whereas the population sizes of the real data sets used within the AUE and AWE comparisons was relatively small (tens of thousands each), the synthetic population sizes used within the AUE and AWE comparisons were massive in comparison. The synthetically generated population sizes Brzeziński and Stefanowski used within their studies ranged from $1,000,000$ to $20,000,000$ examples within the data stream.

Another distinction to be made between the AWE/MPC study and the AWE/AUE comparison is that where the AWE and MPC studies examined their ensemble techniques using several baseline classifier methods (J48 decision tree, Ripper, Bayesian Network), the AUE and AWE comparative

study constructed ensembles using Hoeffding Trees as the only classifier method. Examining the experimental procedure of Brzeziński and Stefanowski's AUE and AWE comparison a bit further shows that while they examined the algorithms with larger and more data sets, they only appear to measure each ensemble method from a single trial per data set. The AUE and AWE study may have examined the methods with larger data streams, and convincingly with more diversity in the additional data sets, but the experiment ignored the influence chunksize and ensemblesize have on algorithm performance.

As we examined the differences between experimental procedures of the prior studies we also found differences in which metrics were measured within each. For example, the AWE and MPC studies measured only classification accuracy and algorithm efficiency, whereas the AWE and AUE study also examined the amount of resources the methods consumed. The AWE and MPC studies may have observed classification accuracies and the times required to train and test the ensembles, but their results do not reflect how system resources can be impacted by the different approaches.

The inconsistencies between the experimental methods of the prior studies motivated us to examine the three ensemble methods together, and to borrow experimental procedure elements from each. We have already identified the importance of the chunksize and ensemblesize independent variables both Wang et al. and Masud et al. used within their studies, and we too will engineer our experimental trials around chunksize and ensemblesize. We also see the advantages the MOA framework provided Brzeziński and Stefanowski, and our study leverages it as well. Our experimental procedure then hopes to bridge the differences between the two prior comparisons by using the data sets and framework Brzeziński and Stefanowski used to examine the independent variables (chunksize & ensemblesize) that Wang et al. and Masud et al. identified as having an influence on algorithm performance.

Our performance criteria are the same that were used within the evaluations of the AUE and AWE methods, but accounting for the influence both chunksize and ensemblesize selections have on performance. We examine the methods' performance as characterized by classification accuracy,

ensemble training and testing times, and resource consumption in terms of ensemble memory use. Similarly to the previous studies of the methods, we too are most interested in classification accuracy, but not to the extent that a method continually consumes computational resources, both in terms of training/testing times and memory consumption.

Our first hypothesis is that of the three ensemble algorithms (AWE, MPC, and AUE), one will outperform the rest across the data sets we examine them with. We define outperform as the algorithm that has the highest average accuracy rate, while not being overly inefficient. That is to say a method is considered overly inefficient if it continually consumes computational resources whether it's the time to train/test or the amount of memory required by the algorithm.

Our second hypothesis is that a multiple partition ensemble that dynamically adjusted the number of partitions relative to the amount of error seen within the classification stream would perform better than a partition ensemble that maintained a static partition size, or MPC. That is to say that our contributed algorithm, AMPE, will classify more instances correctly while maintaining reasonable efficiency. We say reasonably efficient because the performance of AMPE is relative to the magnitude of concept drift within the data stream. We measure the performance of our contributed method to that of MPC using the same metrics we use while comparing the AWE, MPC, and AUE algorithms: classification accuracy, training/testing times, and resource consumption.

And finally, our third hypothesis is that the training data accumulated while the system is less stable, or while concept drift is more prevalent, is more important for an accurate system than training data accumulated while the system is quiescent, or while not experiencing concept drift. Our AMPE2 algorithm was motivated by this hypothesis and the hope to gain a better understanding of the characteristics of concept drift through its examination. We measure AMPE2's classification accuracy and efficiency similar to our previous trials, and we compare its performance to our AMPE technique.

3.1 Experimental Design and Data Sets

Our experimental design utilized the Massive Online Analysis (MOA) framework and its associated data set generators to evaluate the performance of AWE, AUE, MPC, and AMPE(2). The data sets we chose for our examinations were most of the data sets used within the comparison Brzeziński and Stefanowski performed between the AUE and AWE techniques. But unlike the AUE and AWE experiments that appear to only evaluate the methods once per data set, our experimental approach accounts for the influence both chunksize and ensemblesize have on algorithm performance. Because of this, we reduce the data sets used within our comparisons to one real data set, and three synthetically generated data streams. They are the Rotating Hyperplane Data Set, the LED Data Set, the SEA Data Set, and the Australian Electricity Data Set. Again, trying to borrow the best approaches of both prior experimental approaches, we select the majority of the very large data sets used within the AWE and AUE experiments. Although our data set selection is larger than the data sets used while comparing the AWE and MPC methods, due to project scope and time we were unable to use all of the data sets used by Brzeziński and Stefanowski in their study. All of the data set parameters and sizes that we used within our experiments were the same that Brzeziński and Stefanowski used, and we examine them shortly.

3.1.1 Massive Online Analysis (MOA) Framework

Machine Learning (ML) frameworks, like that of Weka[20], provide a standardized platform for studying classification algorithms. The Massive Online Analysis (MOA)[2] framework developed by Bifet et al. is a ML framework similar to Weka, but with the specific purpose of providing an environment for implementing and examining data stream classification algorithms. MOA's primary components are data stream generation and management, baseline classification methods, and evaluation procedures. The data stream generation and management provide the capabilities to both generate data stream examples as well as create a data stream from a static file. MOA's data stream generation can also add concept drift to a static data stream through the use of parameters that define the fraction of attributes to add noise to. The baseline classifiers implemented within MOA include Naive Bayes, Hoeffding Tree, Hoeffding Option Tree, Oza and Russel online bagging/boosting, AWE, and AUE. The evaluation methods implemented within MOA include

periodically testing on a holdout set to evaluate a classifier, testing and then training with each example in sequence to evaluate a classifier, or testing and then training with chunks of data in sequence. This latter evaluation technique, the Data Chunk Evaluation method, was introduced by Brzeziński in his 2010 Master's thesis[6].

Prior to Brzeziński's Data Chunk Evaluation method, MOA did not have the ability to evaluate chunk based algorithms like AWE, MPC, and AUE. The Data Chunk Evaluation method evaluates incoming examples within the stream by forming a collection of examples, or "chunks". These chunks are tested on the current model, the model is updated, and then the chunk is disposed of to conserve resources. Once the model is updated, the Data Chunk Evaluation method collects and records classifier performance metrics. Similar to MOA's other evaluation methods, the Data Chunk Evaluation method records classifier metrics in windows of limited purview due to the possible infinite size of incoming data stream examples. Brzeziński recently contributed his chunk evaluation method to MOA, and it has been in subsequent releases since the moa-20110505 release. Brzeziński's Data Chunk Evaluation method contribution allowed us to compare several chunk based ensemble methods together that, to our knowledge, have not all been empirically compared together before.

MOA includes data set generators and evaluation procedures specifically for studying the classification of data streams with concept drift. A significant advantage that MOA provides over using batch methods, like that of Weka and its use within the AWE and MPC studies, is MOA can generate massive amounts of data within a data stream. For example, our studies utilized data stream populations of 1 million, 5 million, and 20 million data stream instances.

MOA provides a controlled environment for evaluating machine learning methods in very large data streams. For instance, MOA's data stream generators create their stream elements randomly, but their "randomness" can be controlled by the use of a seed to the framework. This feature allows the framework to generate very large and diverse populations of elements in a data stream, as well as a means of reproducing the stream for uniform comparison between algorithms. The MOA framework then allows us to compare our methods of interest with incredibly large data streams populated with diverse examples, and to do so uniformly between algorithms.

3.1.2 Data Sets and Experimental Parameters

An advantage of using a data set generator to study concept drift is the size of data streams that can be generated synthetically can be significantly larger than the size of most real data sets. A disadvantage of using a data set generator can be reproducibility, but MOA addresses this through the use of seed parameters. MOA's data stream generators can be directed to reproduce stream elements through these seed parameters that are used internal to MOA's random element generation process. The real data set we use within our studies is the Australian Electricity price data set [9], and it consists of roughly $45,000$ instances. The synthetic data sets we use are from the MOA data set generators. Those data sets are the Rotating Hyperplane data set [8], the LED data set [4], and the SEA data set [15], and we enumerate the characteristics of each:

- **Synthetic Data Sets:**

  Synthetic data sets are data streams generated by MOA concept drift data stream generators. Each data set has attributes and associated parameters that control how concept drift is manifested within the data streams. The three data set generators we used within our studies are:

  - **Rotating Hyperplane:**

    The Rotating Hyperplane data set appears to originate from [10], and was used in a comparison Gamma performed in [8]. The hyperplane representation is of a d-dimensional space consisting of the points x such that:
    $\sum_{i=1}^{d} w_i x_i = w_0 = \sum_{i=1}^{d} w_i$ is satisfied.
    Instances are then labeled:

    * *true* where $\sum_{i=1}^{d} w_i x_i \geq w_0$ and
    * *false* where $\sum_{i=1}^{d} w_i x_i < w_0$

    Perturbation is introduced into the dataset by adding a weight to each attribute with a small probability calculated thusly: $w_i = w_i + d\sigma$, where σ represents the probability of the hyperplane changing direction, and d is the amount of change applied to each example[3]. All of our experiments used a hyperplane data set generated with 5% noise and 5 million instances. The Rotating Hyperplane data set is known to have gradual

concept drift that is proportional to the calculation of w_i above. The parameters we used with the MOA Hyperplane generator were the same Brzeziński and Stefanowski used within their study and are:

* -i (*seed for random generation of instances*) = 1000000
* -c (*number of classes to generate*) = 10
* -a (*number of attributes to generate*) = 10
* -k (*number of attributes with drift*) = 2
* -t (*magnitude of the change for every example*) = 0.01
* -n (*percentage of noise to add to the data*) = 5
* -s (*percentage of probability that the direction of change is reversed*) = 10

– **LED Data Set:**

The LED data set represents a digit within a seven digit LED display. The LED data set originated within the CART book[4]. The LED generator produces 24 attributes, of which only 7 are relevant, and is known to have very sudden concept drift between class instances. For our studies we generate 1 million LED instances. The parameters we used with the MOA LED generator were the same Brzeziński and Stefanowski used within their study and are then:

* -i (*seed for random generation of instances*) = 1000000
* -n (*percentage of noise to add to the data*) = 5
* -d (*number of attributes with drift*) = 2

– **SEA Data Set:**

The SEA concepts generator was used first in [15], and similarly to the LED data set, it generates sudden, or abrupt concept drift. The data set consists of two decision classes, and three attributes. Only the first two attributes are relevant, and all attributes have values between 0 and 10. The SEA generator then divides those values between 4 blocks of differing concepts, and assigns classes according to a linear function. All of our experiments generated 20 million instances with 5% noise. The data set population size generated by the SEA generator is then as big as the Rotating Hyperplane data stream, but unlike the Rotating Hyperplane data stream, the SEA generator creates sudden concept drift many times within its massive population. The parameters we used with the MOA SEA generator were the same Brzeziński and Stefanowski used within their study and are:

∗ -i (*seed for random generation of instances*) = 1000000

∗ -n (*percentage of noise to add to the data*) = 5

- **Real Data Set:**

Our experiments used one real data set that is known to have concept drift within it. MOA streams this data set to our algorithms under study similarly to how it generates data streams artificially, but adds concept drift to the data stream using a sigmoid function. The characteristics specific to the Australian Electricity data set are then:

– **Australian Electricity Data Set:**

The Electricity data set was described by M. Harries[9] and used by Gama in [8]. The data is from the New South Wales Electricity market and consists of ≈45k instances, collected within 30 minute intervals of electricity prices affected by market demand, season, weather, etc. The decision class consists of "up" or "down" values, and concept drift is known to be somewhat balanced between the two decision classes as the prices of electricity fluctuated moderately back and forth during the times the data was collected. The parameters we use within generating concept drift for our real data stream were the same Brzeziński and Stefanowski used within their study and are as follows:

* -r (*seed for random generation of instances*) = 1000000
* -a (*fraction of attribute values to disturb*) = 0.1
* -d (*number of attributes with drift*) = 2

3.1.3 Base Classifier

All of the methods we examine within this thesis are ensemble techniques that maintain a collection of classifiers. In an effort to reduce experimental scope and time, we fixed the underlying classifier in our examinations. The base classifier chosen for our comparative study between ensemble methods was a Hoeffding Tree[3, 2] with Naive Bayes leaves. We selected this classifier to remain consistent with the prior study Brzeziński and Stefanowski performed in their comparison of the AWE and AUE techniques.

A Hoeffding Tree is an incremental decision tree that exploits the mathematical concept known as a Hoeffding bound. The Hoeffding bound guarantees that the true mean of a random variable with range R will not vary from the estimated mean after n independent observations by more than ϵ, where $\epsilon = \sqrt{\frac{R^2 ln(\frac{1}{\delta})}{2n}}$, within a probability 1-δ[2].
The Hoeffding Tree parameters used within the experiments were as follows: split confidence $\delta = 0.01$. These choices were selected simply because that is what Brzeziński and Stefanowski used within their studies, and are fixed for our studies.

3.1.4 Algorithm Parameters and Experimental Procedure

All of our experiments were conducted on systems with 8 Intel Xeon 2.5GHz CPUs, 16GB of RAM, and Fedora Core 15 (X64) operating systems. All of the algorithms under test were written in Java and run from within the MOA framework. We used the MOA Data Chunk Evaluation method to measure classifier accuracy, CPU times required for training and testing, and overall memory the ensemble method has allocated.

In their examinations of AWE, Wang et al. show that not only does data set selection affect

performance and classification error, but so do the size of data chunk, and number of classifiers within the ensemble[18]. Both experiments that examined AWE and MPC take into account the influence of chunksize, ensemblesize, and data set selection. They do so by manipulating the size of data chunks and ensemble memberships within their experimental trials across several data sets. Our experimental procedure follows this approach, and all of our experiments manipulate data sets (Hyperplane, SEA, LED, & Elec), chunksize (250, 500, 1000, 1500), and ensemblesize (2, 4, 8, 16). In all of our trials the following additional parameters were used for each of our algorithms under test:

- AUE: *none.*
- AWE: *none.*
- MPC: *partition number* $v = 2$
  number of chunks $r = 5$
- AMPE: *number of chunks* $r = 5$
- AMPE2: *number of chunks* $r = 5$

Chapter 4

Results and Analysis

One of the goals of our experiments was to gain a better understanding of the trade-offs between classification accuracy and efficiency between several state of the art ensemble methods that all purport to cope with the effects of concept drift. We hypothesized that of these three methods (AWE, MPC, & AUE), one would show itself to be more accurate and as efficient as the others for the data sets we examined.

Our contributed methods, the AMPE, and AMPE2 algorithms, were motivated by the desire to gain a better understanding of concept drift adaptation and concept drift characteristics. We hypothesized that adding a concept drift detection mechanism and adjusting partition sizes accordingly would yield an ensemble technique with better accuracy than the MPC method. We also hypothesized that an ensemble that adjusted partition sizes to train and test with while concept drift was present, as in the case of our AMPE technique, would be more accurate than a method that adjusted partition sizes when the underlying class distribution was stable, as is the case with our AMPE2 method.

The results and associated analysis are organized in this chapter first by our experiments that examine the existing state of the art techniques, and we then present our results and analysis of our contributed methods compared to the MPC technique.

4.1 Existing Ensemble Methods

Our experimental procedure followed what had been done in the prior examinations of AWE, MPC, and AUE. We identified two independent variables: chunksize & ensemblesize. And our results are then presented by data set, and then according to ensemblesizes and chunksizes. For example, the evaluation of classification accuracy was measured for each of the chunksizes (250, 500, 1000, 1500) by averaging results for each of the ensemblesizes (2, 4, 8, 16) at that chunksize. Both the figures and tables within the remainder of this thesis are then organized by chunksize and ensemblesize trials. We now examine the classification accuracy rates, and efficiency of the AWE, MPC, and AUE methods while processing the Australian Electric data set, the LED data set, the Rotating Hyperplane data set, and the SEA data set.

4.1.1 Electricity Data Set Accuracy Results

Recall that the Australian Electricity data set is relatively small, consisting of roughly 45k instances, and the smallest of the data sets we used within our experiments. The Electricity data set is also the only *real* data set used within our trials. Figure 4.1 graphically presents the accuracy results of running the three ensemble methods on the Australian Electricity data set. The first observation that we make of the results in Figure 4.1 is that the selection of chunksize appears to influence some algorithms more than others. Though all of the accuracy rates appear to be declining as chunksize increases, the AUE and AWE methods appear to have chunksize selections that improve accuracy at times, whilst the MPC method's accuracy rate appears monotonic in decline.

The second observation we make of these results is that increasing the size of the chunk appears to inversely affect classification accuracy for all the methods, whereas adding additional classifiers into the ensemble appears to improve performance. The inverse influence data chunksize selection has on performance might be attributed to trying to use too large a chunksize within a smaller data set. The accuracy gains of adding additional classifiers to the ensemble might be attributed to the additional diversification of more classifiers within the ensemble population. Wang et al. attribute

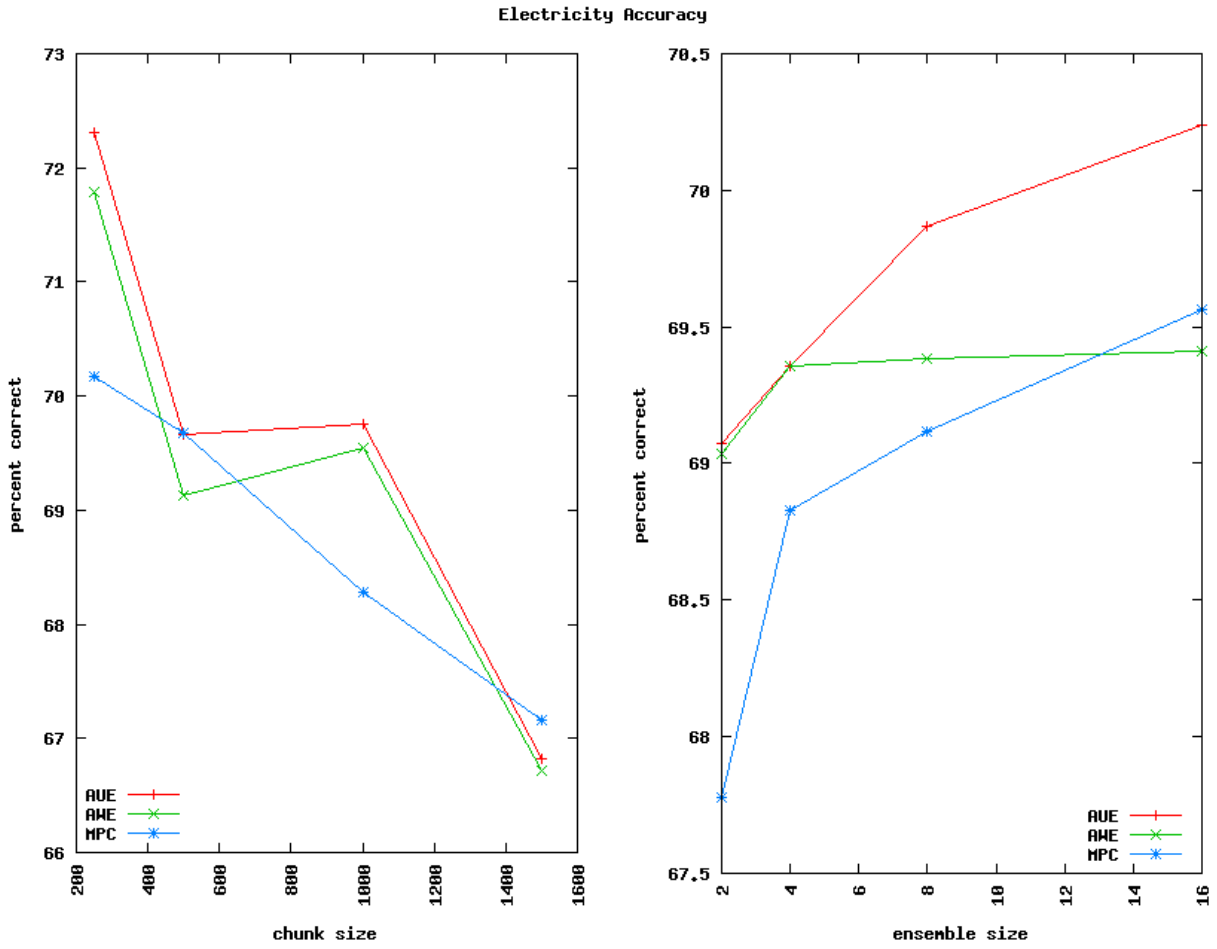

Figure 4.1: Accuracy for AWE, MPC, & AUE methods on Electricity Data Set.

the error reductions they witnessed within their experiments to the increased diversification of the ensemble when more classifiers were added.

And finally the third observation that we make of these results is that it is difficult to distinguish one algorithm outperforming another. The results for AUE, AWE, and MPC presented within Figure 4.1 do not clearly show one technique outperforming the rest, and require additional analysis. Table 4.1 and Table 4.2 show more detailed results of each of the experimental trial results. Note: Table results appearing in **bold** represent the **best** average for that data set, and results appearing in *italics* highlight *worst* average values for that data set. Table results can be found at the end of each results section in this chapter.

If we average our chunksize and ensemblesize manipulation results for each data set, we see that the $AUE_{avg.}$ has the highest mean classification accuracies for both the chunksize and ensemblesize manipulation trials. All prior comparisons performed by Wang et al., Masud et al., and Brzeziński and Stefanowski only report the mean measurements, and not the accompanied errors. If we decided not to proceed with additional analysis, and followed the experimental protocols of the prior comparisons, we would report that the AUE method performed better than both the AWE and MPC methods. Examining the results of the different ensemble methods a bit closer shows that the variances between measurements cannot be discounted. The measured $AUE_{avg.}$ error of ± 2.24 for the chunksize trials also encompasses the mean measurements for all of the other methods.

Our experimental design consisted of two independent variables: chunksize and ensemblesize. Due to this, further analysis of our experimental results require the use of a two way ANOVA approach per independent variable versus method. We ran two way ANOVA statistical tests at the $\alpha = 0.05$ for our methods versus chunksize, and method versus ensemblesize. For chunksize, the two way ANOVA did not show main effects for either method ($F = 6.71, p > 0.086$) or chunksize ($F = 4.34, p > 0.204$). These p-values tell us that the test fails to reject the null hypothesis that the average accuracies are equal, thus there is not a statistically significant difference between choice of method or chunksize for this data set. The interaction effect between chunksize and method was determined to be significant ($F = 2.97, p < 0.0185$). Whereas we do not see an influence either the method or chunksize selections make independently, rather we see that we fail to reject the null hypothesis that there is not an interaction between the method and chunksize factors.
The ANOVA test for our second independent variable, ensemblesize, also fails to show a significant statistical significance between the three methods($F = 0.657, p > 0.524$) or the ensemble size($F = 0.6627, p > 0.580$) for this data set. A test of independence between method and ensemblesize fails to find a significant interaction between method and ensemblesize selection ($F = 0.102, p > 0.996$). The results of our ANOVA analysis confirm that it is difficult to establish a clear performance leader for this data set. The results also show that both the selection of chunksize and number of classifiers within the ensemble population may not be as important for

classifier accuracy with this particular data set.

4.1.2 Electricity Data Set Efficiency Results

Figure 4.2 shows the different memory requirements each ensemble method requires while processing the Australian Electricity data set. Similar to our earlier accuracy results, the size of chunk appears to influence the efficiency in all methods. Unlike our previous results, we see that the selection of ensemblesize only appears to influence the AUE method. Unlike AUE, both the AWE and MPC methods do not appear to grow in size as the ensemblesize is increased. Examining Table 4.3 and Table 4.4 shows the AUE method had the highest averaged mean memory requirement, and we see that the AWE method had the least average memory requirement of the methods for the Electricity data set. We examine both the Electricity data set training times in Figure 4.3, and the testing times within Figure 4.4 together. These graphs show that both classifier training and testing times appear to be influenced by the selection of data chunksize. Similar to the memory resource consumption AUE showed within 4.2, AUE's training and testing times also appear to be influenced by the selection of ensemblesize. This might be attributed to a resource leak within the AUE algorithm, or it could be an anomaly of this particular data set. Larger data sets may exacerbate this growth in resource requirements that AUE shows to have had on the Electricity data set.

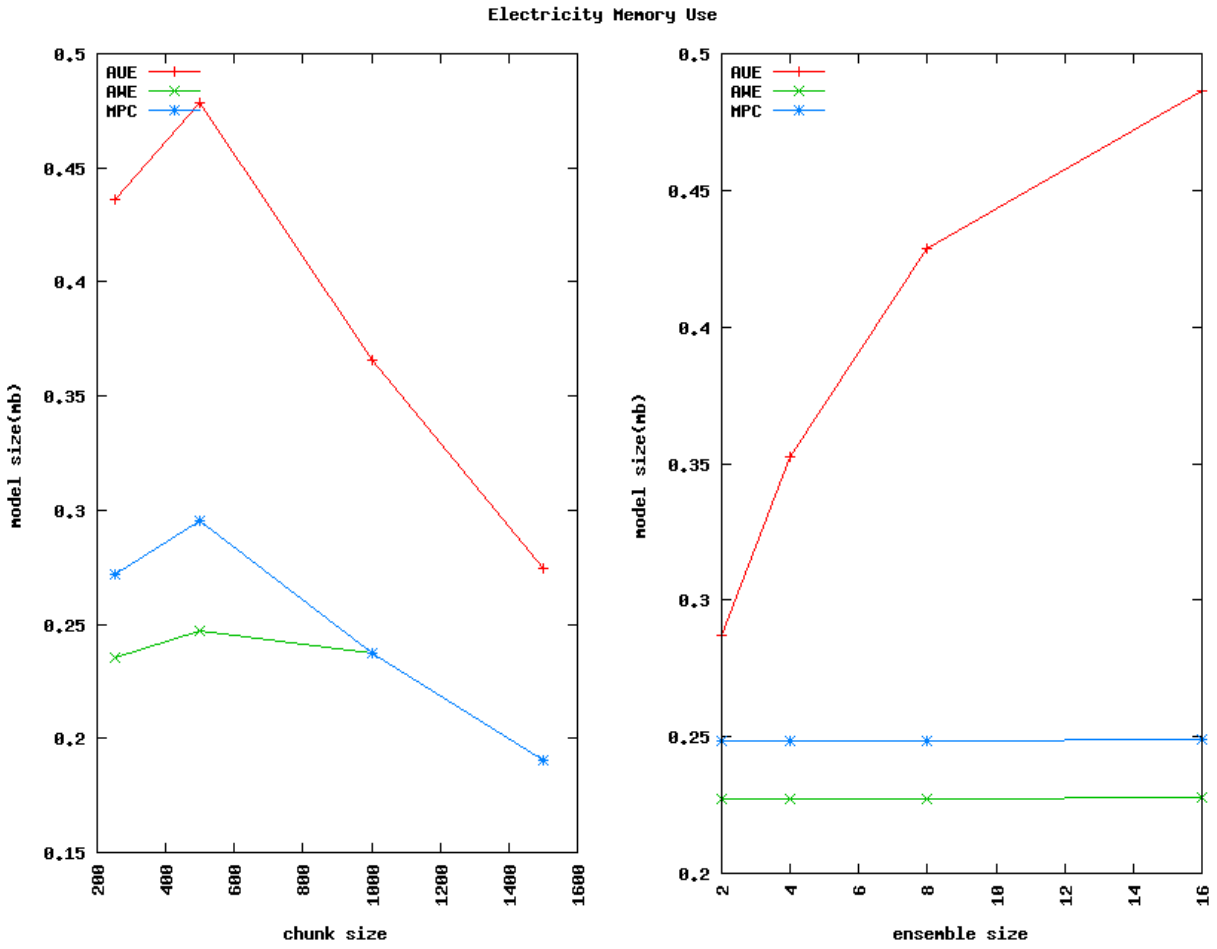

Figure 4.2: Ensemble Model Size for AWE, MPC, & AUE methods on Electricity Data Set.

4.1.3 LED Data Set Accuracy Results

The LED data set is the smallest of our artificial data sets, but still quite large with 1 million examples generated within the data stream. Similar to our Electricity results, we present our findings by examining classification accuracy and we then look at efficiency of the ensemble techniques. Figure 4.5 shows the results of measuring classification accuracy for chunksize and ensemblesize while processing the LED data set. The first attribute we notice about the LED data set results is how chunksize selection does not appear to have the influence it did within the Electricity data set. Where we saw certain chunksizes influence accuracy more than others in the Electricity data set results, the LED accuracy results for chunksize appear monotonic in growth. Our second observation is that both chunksize and ensemblesize selections appear to cause the accuracy rates to

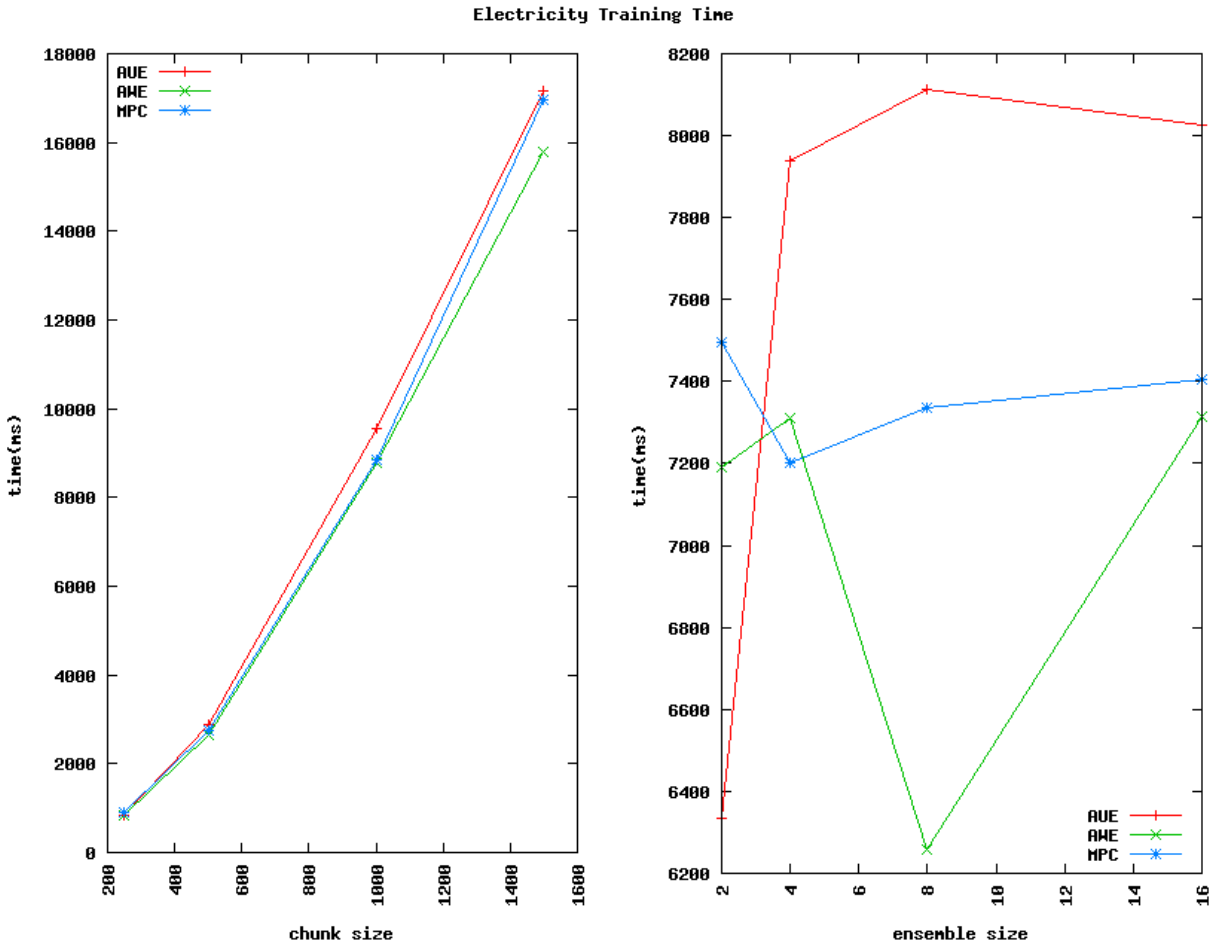

Figure 4.3: Training Times for AWE, MPC, & AUE methods on Electricity Data Set.

converge across the three methods at about 1500 for chunksize, and 16 for ensemblesize. Wang et al. also describe seeing trade-offs between chunksize selection and classification error [18] within their experiments. Their research shows how increasing chunksize is beneficial to classification accuracy, but only up to a point, and then any additional increase within chunksize would only harm performance. Their reasoning was that concept drift could be spread between chunks if the chunksize was too little, or concept drift could go undetected within very large chunks if they occur multiple times in the same chunk. Masud et al. also describe "sweet spots" for ensemblesize selection within their examinations of AWE and MPC [12]. Their studies show error reduction associated with increases in ensemblesize, but again only to a point. Their trials show that performance did not improve much after ensemblesize was increased above 8 classifiers in the ensemble. Both the AWE and MPC

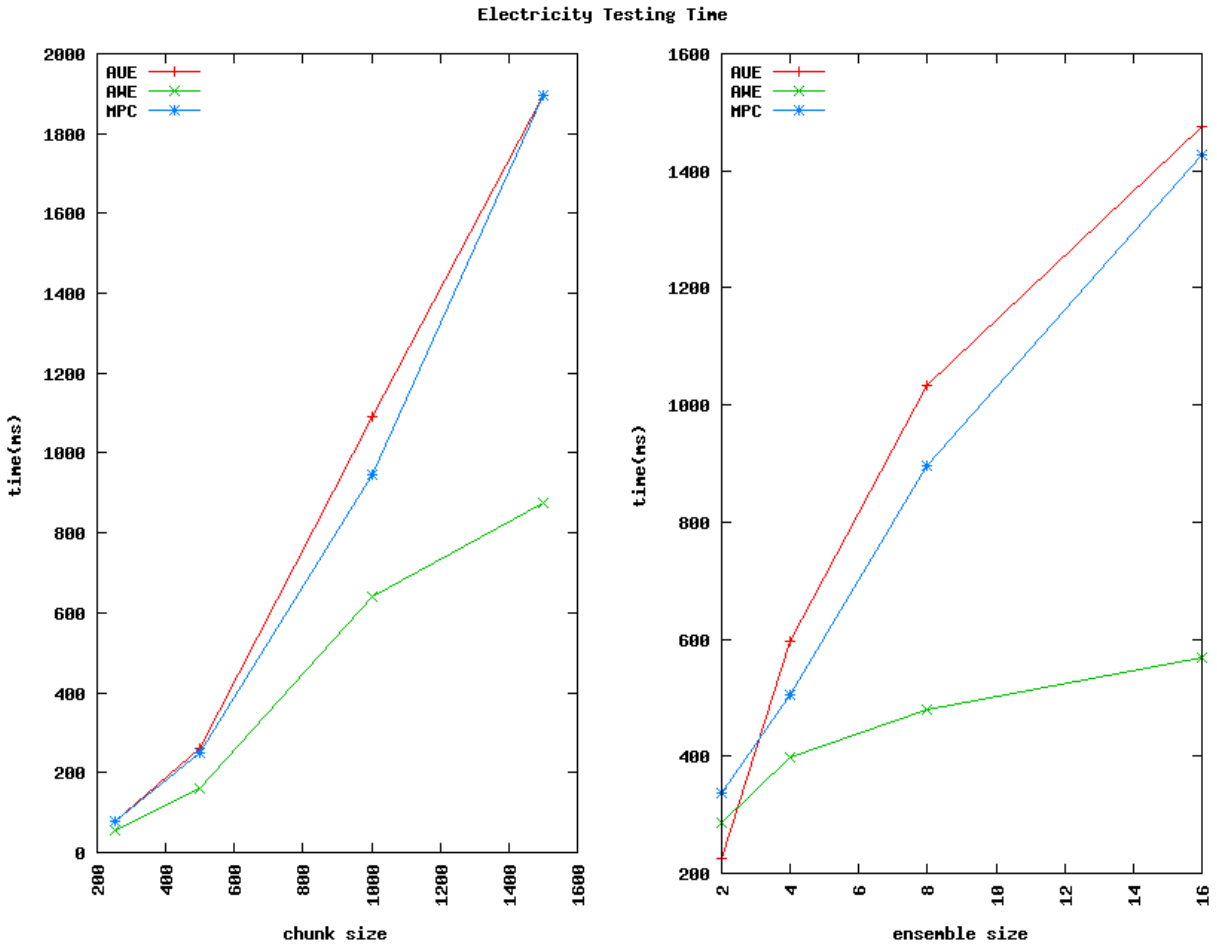

Figure 4.4: Testing Times for AWE, MPC, & AUE methods on Electricity Data Set.

studies suggest the importance of identifying the relationship between classification performance and selection of chunksize and ensemblesize. The results we present here also appear to reaffirm the importance of chunksize and ensemblesize.

Finally our third observation we make of this data is that while the LED data set results appear to be more independent of one another than what we observed within the Electricity data set, it is still too difficult to determine a clear leader of the methods. Additional analysis of Table 4.1 and Table 4.2 shows the MPC method to have the highest mean classification rate, but again like the Electricity results, the errors of the method results overlap and cannot be discounted. And again similar to the Electricity data set analysis, we perform two way ANOVA statistical tests using

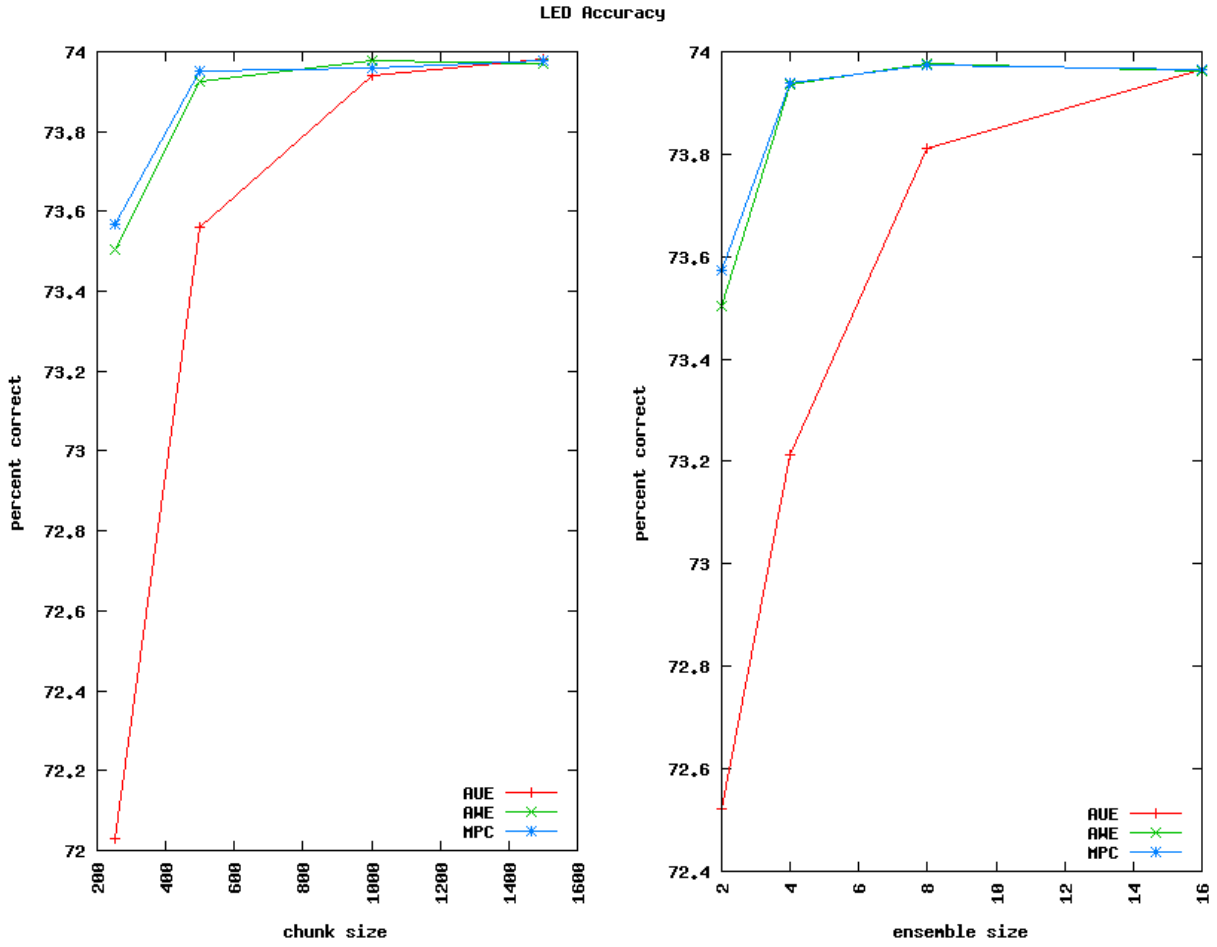

Figure 4.5: Accuracy for AWE, MPC, & AUE method on LED Data Set.

chunksize and ensemblesize for independent variable selection.

Our ANOVA results for our first factor, chunksize, fail to reject that the average classification rates of the three methods are the same ($F = 246, p > 0.099$). We cannot establish a significant statistical difference between the AWE, MPC, and AUE methods' accuracy rates for this data set. This first ANOVA test also reported that chunksize selection did not make a statistical contribution to performance ($F = 4.88, p > 0.059$), and we do not see an interaction between chunksize and method ($F = 1.37, p > 0.255$).

Our second two way ANOVA for ensemblesize also fails to reject that method average classification rates were equal ($F = 1.92, p > 0.161$), and we also do not see an interaction between ensemblesize and method ($F = 0.48, p > 0.818$). But unlike our earlier Electricity data set result, this two way ANOVA test rejects the hypothesis that ensemblesize influences classification accuracy ($F = 2.34, p < 0.008$). We examine this statistical significance further with a multiple comparison test. After identifying a statistical difference exists between ensemblesize selections, we use a Tukey HSD test for comparing the multiple ensemblesizes together. This test identified a significant statistical difference between average ensemble classification rates in trials that used 16 classifiers versus trials that only used 2. Our Tukey HSD test showed that ensemble trials that used 16 classifiers showed an average classification accuracy increase of 0.278 percent over those trials that were comprised of ensembles that only had two classifiers. We did not see a statistical difference between ensemble sizes in our prior examination of the methods using the Electricity data set, but this might be attributed to the sizes of two data sets. The Electricity data set may not be large enough to distinguish an optimal ensemblesize parameter selection, whereas the LED data set is comprised of a million data instances.

Our ANOVA results fail to show a statistical difference between average accuracy rates of the three methods, but we are able to gain a better understanding of how chunksize and ensemblesize selections affect classifier performance. Even though we were unable to distinguish which method is more accurate, we still look at the efficiency characteristics of the techniques.

4.1.4 LED Data Set Efficiency Results

The first resource we examine within our ensemble efficiency analysis is the average required memory for the ensemble while processing the LED data set. Figure 4.6 shows the average model size for each of the ensemble methods for the LED data set. This plot shows us that the AUE method appears to be affected by both the size of data chunks and the size of ensemble. The AUE algorithm appears to require more memory when either the number of chunks are increased, or the ensemblesize grows in classifiers for this data set. This plot does suggest the AUE algorithm to have the highest average mean model size of all the algorithms under test. And as we examine the results within Table 4.3

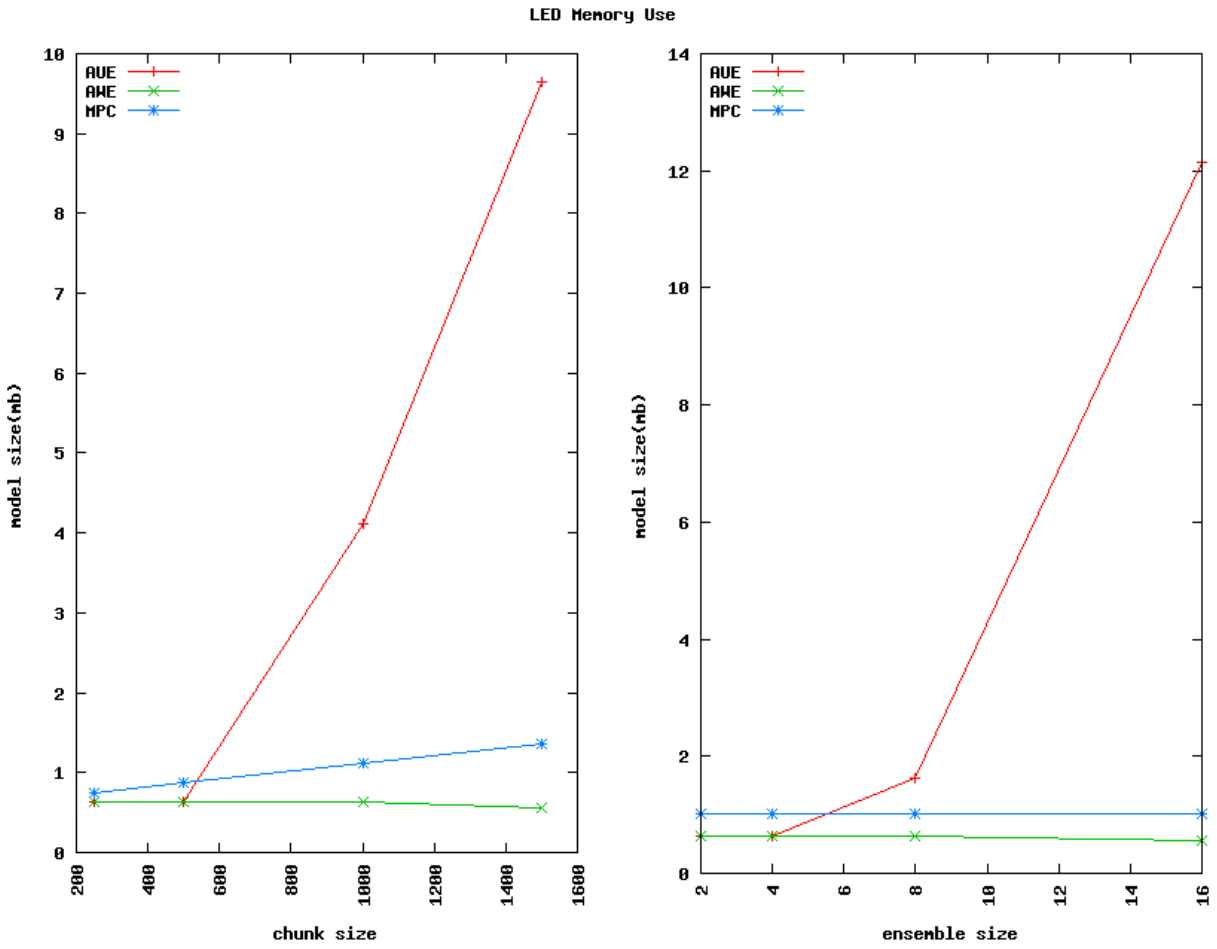

Figure 4.6: Ensemble Model Size for AWE, MPC, & AUE methods on LED Data Set.

and Table 4.4, we observe the AWE method has the lowest average memory requirement, and it appears that the AUE ensemble requires the most memory. One would think that if an ensemble method's memory requirements grow linearly, so would the time it takes to train and test. We then examine the LED training and testing time measurements plotted within Figure 4.7, Figure 4.8, Figure 4.7, and Figure 4.8.

The LED data set is our first really large data set that we examine testing and training time measurements with. Similarly to the Electricity data set results, we see that chunksize and ensemblesize selection affects the time the AUE method requires for training and testing phases. But unlike the results we had seen earlier, the LED data set results appear to show that the AUE method is affected more by ensemblesize selection than chunksize selection. Examining Table 4.5,

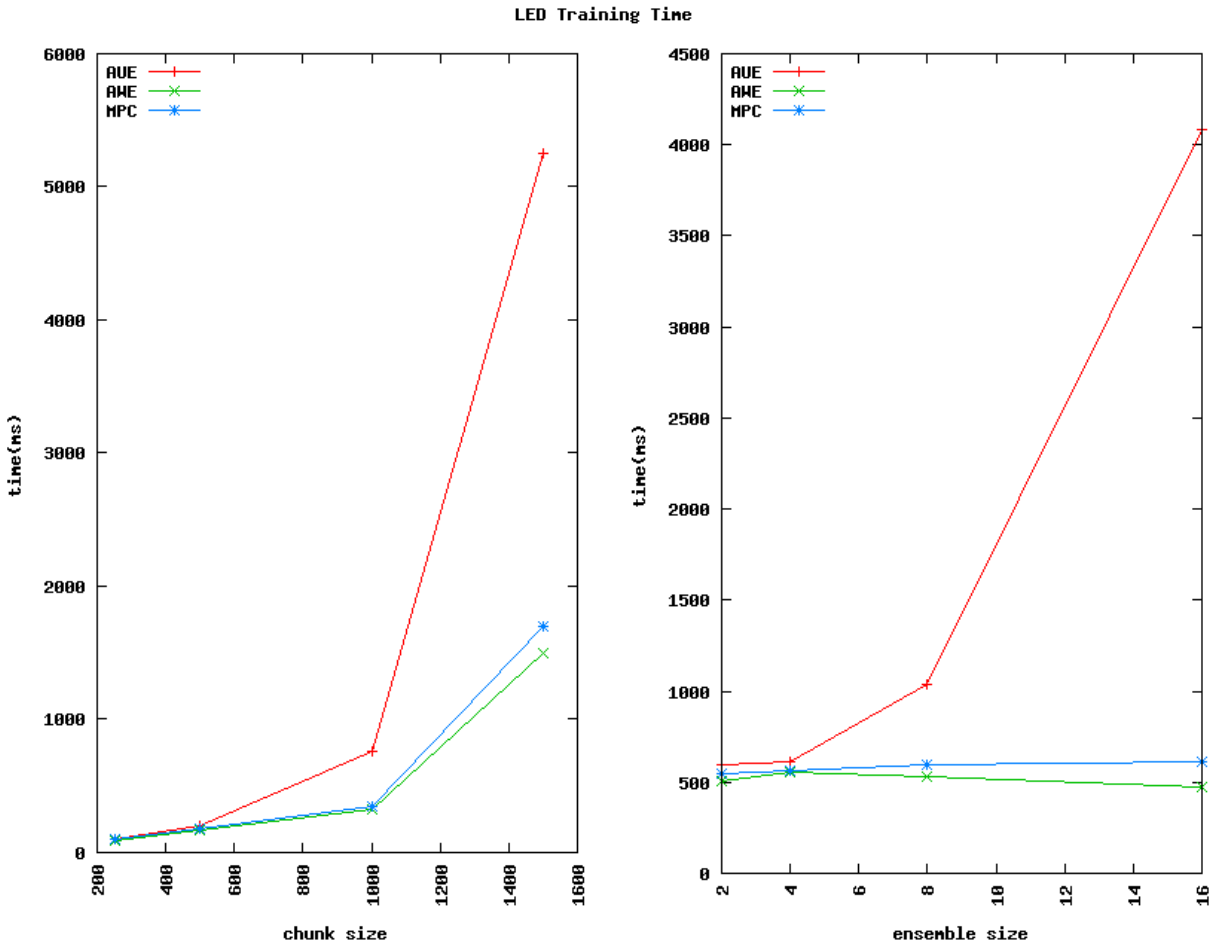

Figure 4.7: Training Times for the AWE, MPC, & AUE methods on LED Data Set.

Table 4.6, Table 4.7, and Table 4.8 shows that the AUE method requires the most amount of average time for training and testing with the LED instances. They also show that the AUE method's training times to be more affected by ensemblesize selection versus chunksize selection. The AUE testing times are approximately one third higher than both AWE and MPC, but the AUE training times are nearly three times that of the AWE and MPC requirements. Upon seeing the influence ensemblesize selection has on the AUE method's resource consumption within the LED data set results, we begin to look for it within the additional larger data set results.

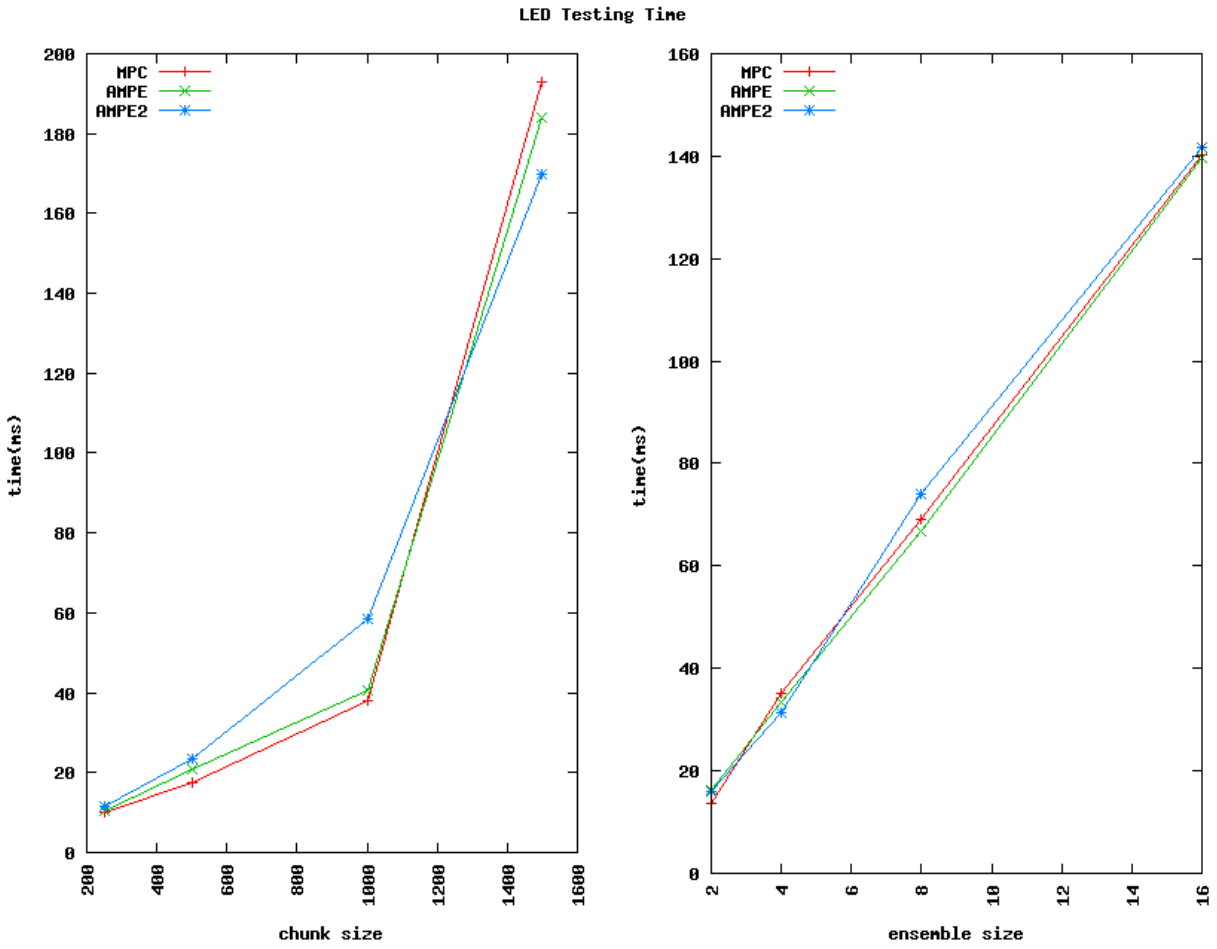

Figure 4.8: Testing Times for the AWE, MPC, & AUE methods on LED Data Set

4.1.5 Rotating Hyperplane Data Set Accuracy Results

The Hyperplane data set is massive in comparison to the size of the Australian Electricity data set, and it is 5 times the size of the LED data set used within our trials. The Hyperplane data set has a very gradual representation of concept drift and we used the MOA Hyperplane data generator to create 5 million instances for our experiments. The Hyperplane results are from the second largest of the artificially generated data sets we tested the three methods with, and as we can see within Figure 4.9, it is the first data set to show more variance between method accuracy rates.

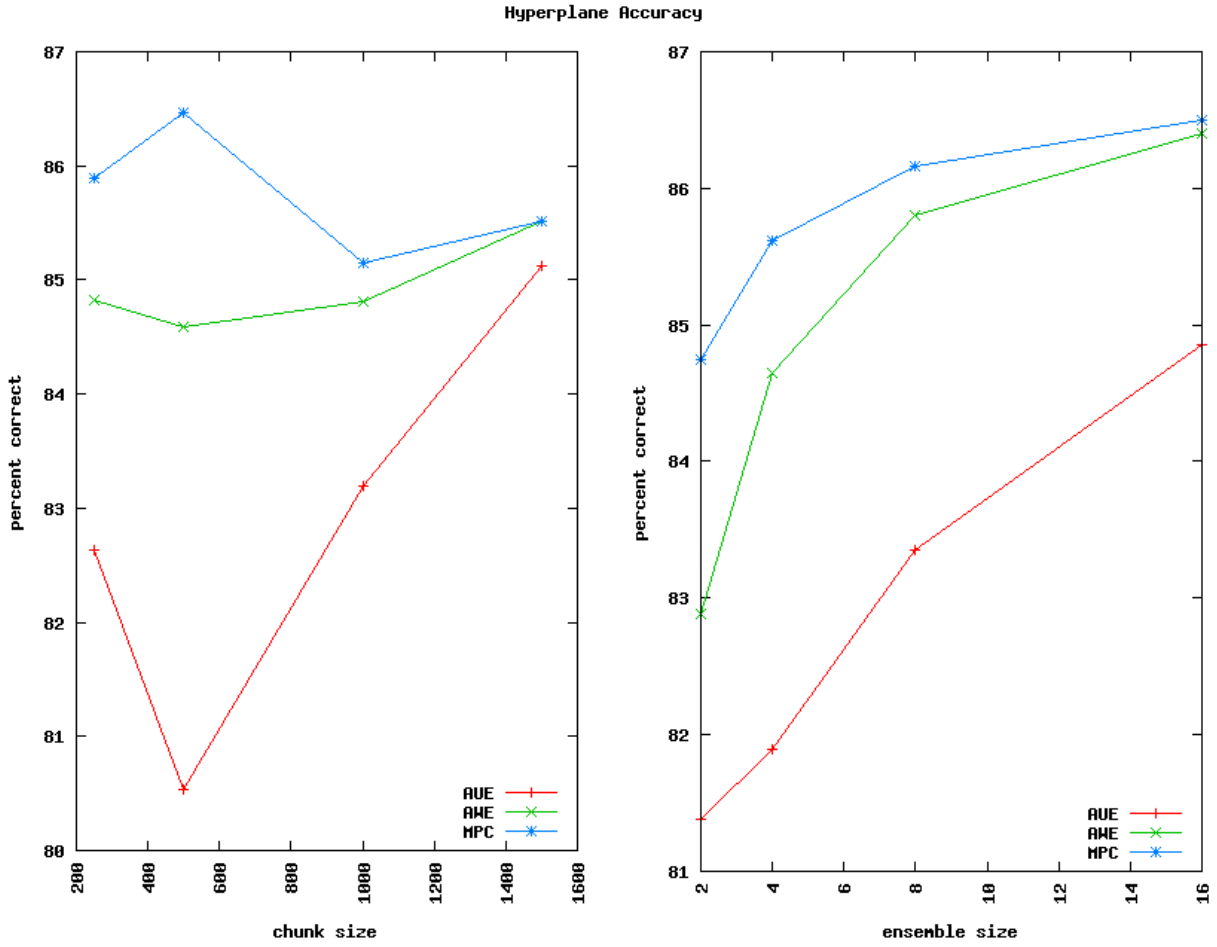

Figure 4.9: Accuracy for AWE, MPC, & AUE methods on Hyperplane Data Set.

Our first observation about the method accuracy results is that similarly to what we had seen with the Electricity and LED data sets, some methods appear to have chunksize selections that influence performance more than other chunksize selections. Figure 4.9 shows that for chunksize = 500 the MPC method's average accuracy rate to be the highest of the different chunksize selections. The plot also shows the chunksize selection = 500 to have an inverse influence on the AUE method's average accuracy. The accuracy gain of the MPC method, and the accuracy decrease of the AUE method can also be seen within Table 4.1.

Our second observation about the method accuracy rates is that it appears the AUE method significantly underperforms both AWE and MPC in both the chunksize and ensemblesize experi-

ments. Not resting on just our "eyeball heuristics", we performed another set of two way ANOVA tests. The first test examines the effects method and chunksize have on classifier accuracy rates, and the second test examines the effect our second factor, ensemblesize, has on accuracy. Our ANOVA results for our first factor, chunksize, reject that the average classification rates of the three methods are the same ($F = 14.63, p < 2.23E-05$). This ANOVA result soundly rejects the null hypothesis that the average method accuracies equal. Our two way ANOVA test identifies that a difference exists between at least one of the methods' average accuracy rates, but it does not identify which method, only that a difference exists. We continue our analysis by performing a Tukey HSD multiple comparison test to determine which methods exhibited a statistical performance difference. Our Tukey tests yielded statistically significant differences between the MPC and AUE methods, and between the AWE and AUE methods. These results show that the MPC method's average accuracy rate was approximately 2.88 percent better than the AUE technique, and the AWE method's average accuracy rate was approximately 2.06 percent better than the AUE method. Our Tukey tests show a significant difference between the AUE method and the AWE and MPC techniques, but we do not see a significant difference between the AWE and MPC average accuracy rates.

Unlike our previous results thus far, the Hyperplane data set results appear to distinguish differences between method accuracy rates. Our first two way ANOVA results for chunksize and method also show that chunksize selection was not statistically important to average accuracy of the methods ($F = 1.99, p > 0.132$), and there does not appear to be an interaction between method and chunksize for this data set either ($F = 2.35, p > 0.0515$). Although the observations we make of chunksize influence in Figure 4.9 suggested that the MPC and AUE methods' accuracy rates were influenced by chunksize, our statistical tests of all three methods said otherwise.

Our second two way ANOVA test examines the interaction of ensemblesize and method. This second ANOVA test also rejected the hypothesis that the average accuracy rates of the three ensemble methods were equal ($F = 18.17, p < 3.51E-06$). We examine this significance further with a Tukey test. That Tukey test showed differences between the MPC and AUE methods, and

differences between the AWE and AUE techniques. The ANOVA tests that examine chunksize and ensemblesize both show the MPC and AWE methods to perform similarly, and the AUE method to underperform both. Our second ANOVA test also examined the effect ensemblesize selection has on accuracy performance. Unlike our first ANOVA test that examined chunksize and found it to not affect accuracy, our second test shows that ensemblesize selection is important to classifier accuracy ($F = 9.93, p < 6.57E-05$). Our second ANOVA results also fail to show an interaction between ensemblesize and method ($F = 0.63, p > 0.703$). We examine the significance of ensemblesize selection to accuracy with a Tukey test, and we find a significant difference in average accuracies of the trials that used 2 classifiers within the ensemble versus 16 classifiers within the ensemble. The average classifier accuracy rate was found to be approximately 2.59 percent higher in the Hyperplane trials that used 16 classifiers versus only 2. Similar to the accuracy increases we had seen within the LED data set experiments, the ensembles also appear to benefit from additional classifiers while processing the Hyperplane data set. Of interest to us now is the trade-off between efficiency and additional classifiers within the ensemble, and we examine it next.

4.1.6 Rotating Hyperplane Data Set Efficiency Results

The average model size required for ensembles processing the Hyperplane data stream can be seen within Figure 4.10. The first observation we make of these results is that the plot appears to show the AUE average model size to be significantly larger, by 20-30 times that of the other ensemble methods. The AUE ensemble method appears to be very sensitive to both the size of data chunk and ensemblesize. When we examine the tabular results a bit closer within Table 4.3 and Table 4.4, we see that the AWE method to have the lowest average memory requirement, and again similarly to the LED results, the AUE method's memory requirement is substantially higher than the other ensemble methods. The AUE method appears to be performing worse, requiring more and more memory as both chunksize and ensemblesize are increased.

Figure 4.11 and Figure 4.12 depict the training and testing times each of the ensemble techniques required while processing the Hyperplane data set. Similarly to the LED results, the AUE algorithm appears to require more time for training and testing the Hyperplane instances. All of the methods

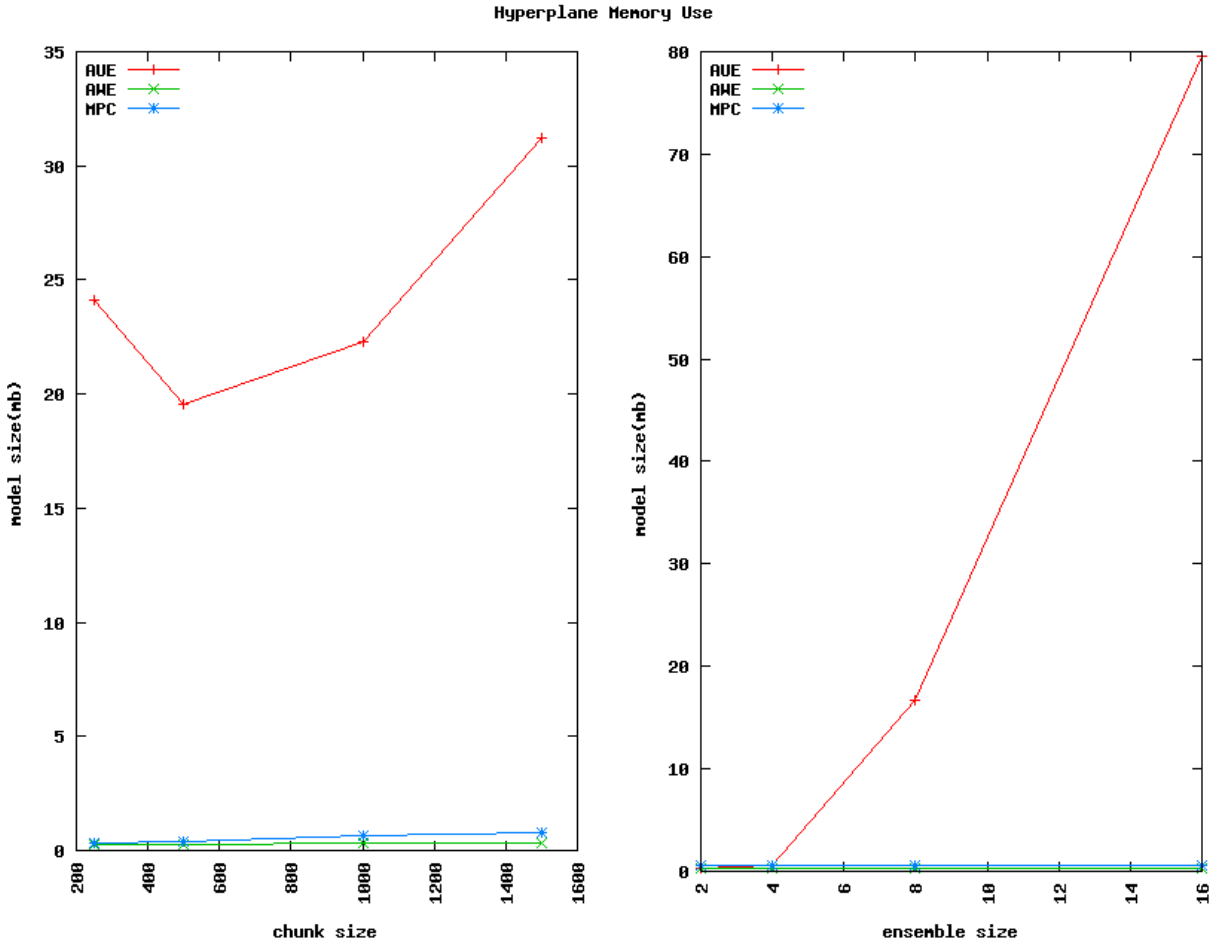

Figure 4.10: Ensemble Model Size for AWE, MPC, & AUE methods on Hyperplane Data Set

appear to be influenced by both chunksize and ensemble membership size, but not to the degree the AUE method is. Table 4.5, Table 4.6, Table 4.7, and Table 4.8 show the MPC method to have the least amount of time required for training and testing Hyperplane examples. And again, similarly to the LED results, the AUE method consistently required more time for testing and training as both chunksize and ensemblesize were increased. The increased size of the Hyperplane data set appears to exacerbate the AUE's resource consumption we witnessed while processing the previous data sets. We begin to see that AUE's memory consumption and time required for training is influenced by the size of the data set. The next data set results we review, the SEA data set, is the largest of the data sets we performed our trials with.

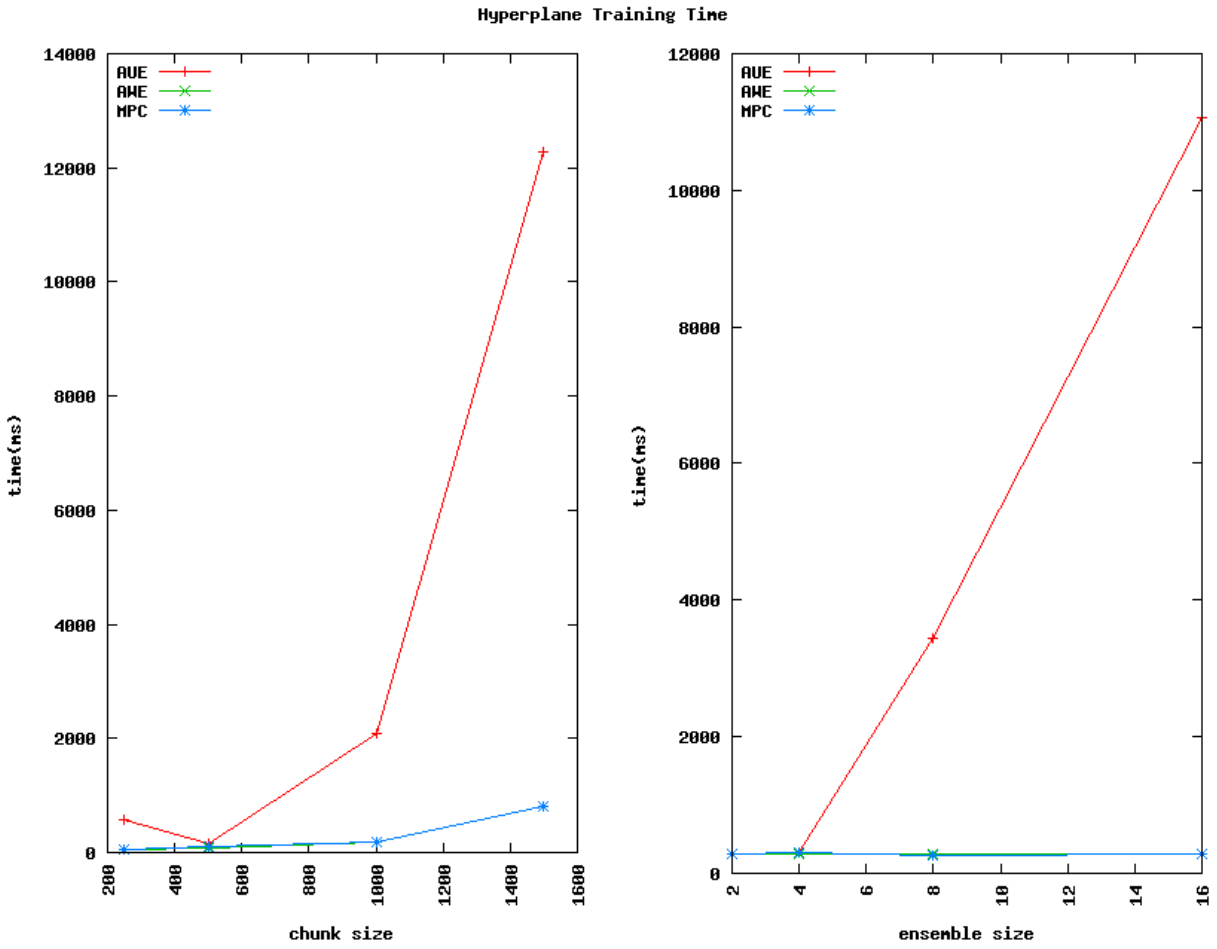

Figure 4.11: Training Times for AWE, MPC, & AUE methods on Hyperplane Data Set.

4.1.7 SEA Data Set Accuracy Results

Figure 4.13 shows the accuracy results from running our experimental trials using the SEA data set. The SEA data set is even larger than the Hyperplane data set, but has concept drift characteristics similar to that of the LED data set. Instead of the slow concept drift that manifests itself over a very large sample of classification instances as in the Hyperplane data set, the SEA data set consists of a set of underlying class distributions that change rapidly, and the relative length of the concept drift is short. Our trials use 20 million SEA instances.

Similarly to both the Electricity and Hyperplane results we analyzed earlier, the SEA data set results appear to have chunksize selections that benefit, or hinder some of the ensemble methods

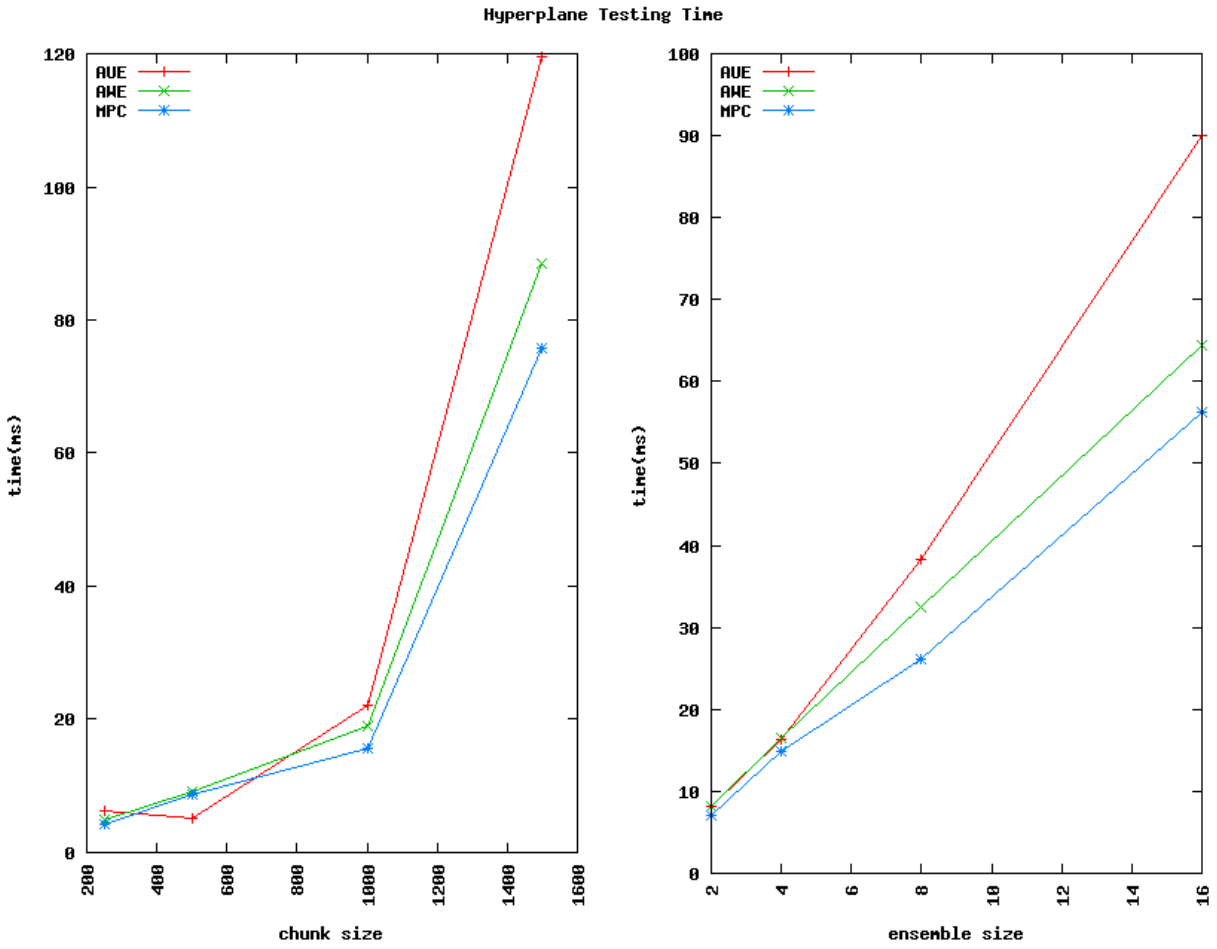

Figure 4.12: Testing Times for AWE, MPC, & AUE methods on Hyperplane Data Set.

more than others. Figure 4.13 shows that when chunksize selection equals 500 the MPC and AUE methods' average accuracy declines. As we perform additional statistical analysis below, we will examine these methods to see if there are particular chunksize selections that are beneficial, or even a hindrance to accuracy, as is the case of the SEA data set and the MPC and AUE methods here. Also similarly to our prior examinations, we see ensemblesize selection to benefit accuracy rates, but at somewhat monotonic growth rates.

Also likewise to our Hyperplane experiments, it appears that the MPC method has the higher of the mean classification rates for the SEA data set. Figure 4.13 shows the classification rate of MPC to be quite a bit higher than the rest in both the chunksize and ensemblesize trials. As we

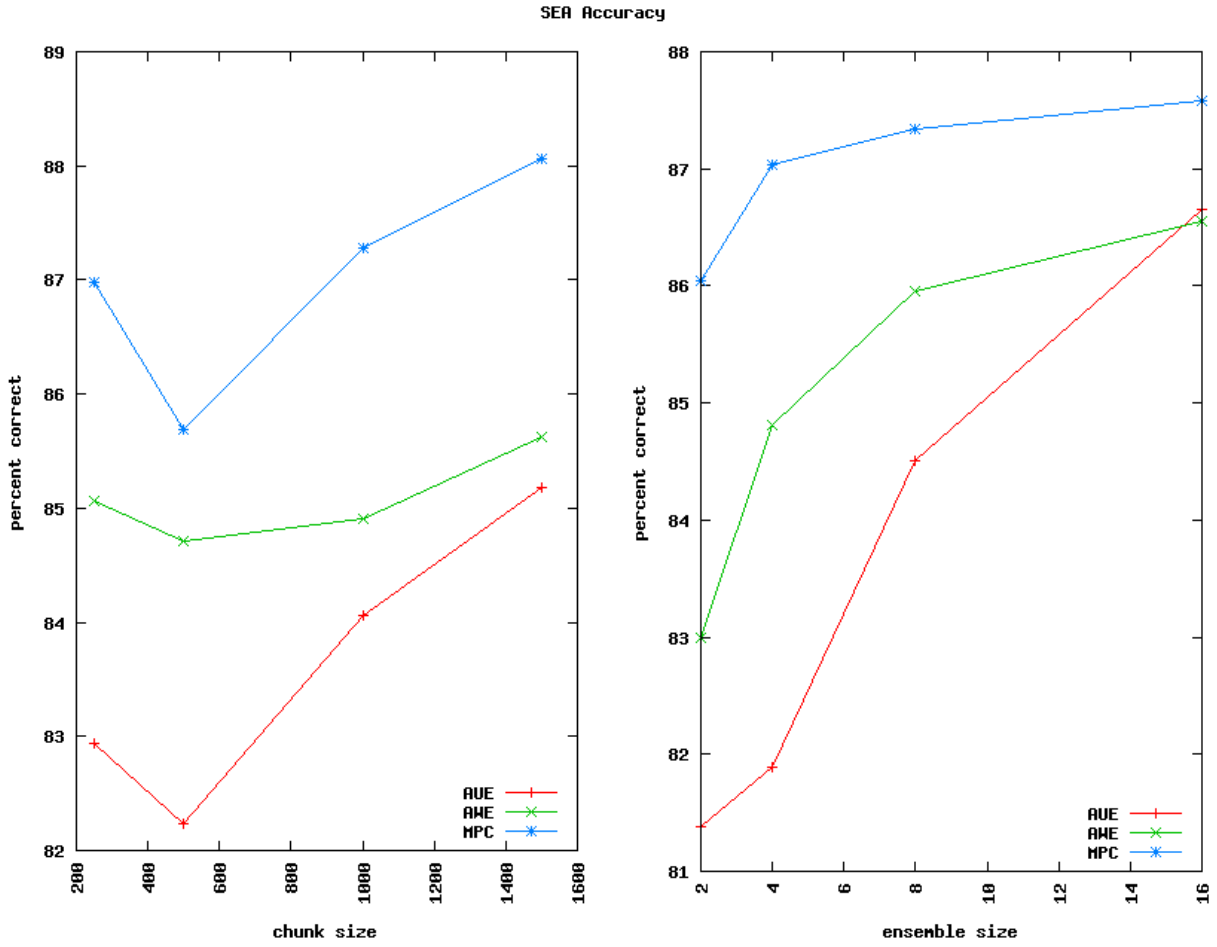

Figure 4.13: Accuracy for AWE, MPC, & AUE methods on SEA Data Set.

have seen with the previous results, examining the tabular results shows a more complete picture, but still requires additional statistical testing. Examining Table 4.1 and Table 4.2 shows us that the MPC ensemble had the highest average mean classification rate of the algorithms tested. Examining these results further leads us to apply another set of two way ANOVA tests for our chunksize and ensemblesize independent variables. Our first ANOVA which examines method and chunksize influence on classification accuracy reject that the three methods' average accuracy rates are equal ($F = 6.68, p < 0.00299$). This result identifies a statistical difference between the methods' average classification rates, but does not say which methods were different. To determine which ensemble methods statistically differ, we use a Tukey HSD multiple comparison test. The results of our Tukey test identified a significant difference between the MPC and AWE methods, a difference between

the MPC and AUE methods, and a statistically significant difference between the AWE and AUE methods. The MPC method was found to have an average accuracy rate that was 2.6475 percent higher than the AUE technique. The MPC method also showed an average accuracy rate that was 1.28 percent higher than the AWE method, and the AWE algorithm was found to have an average accuracy rate that was 1.47 percent higher than the AUE method. These statistical tests show the MPC method to outperform the AWE and AUE methods for the Hyperplane data set. Of the three data set results we have analyzed thus far, these are the first statistically significant findings that show one ensemble method to outperform the others.

Our first ANOVA results also reject the null hypothesis that chunksize selection does not influence average classification accuracy for all three methods examined ($F = 6.11, p < 0.006$). A test of independence between factors also shows that chunksize and method do not have an interaction ($F = 1.005, p > 0.437$). Our second two way ANOVA test examined the influence ensemblesize selection has on average method accuracy rates for this data set. And again, similar to our earlier Hyperplane results, our SEA statistical analysis for our second factor, ensemblesize, also shows a statistical difference between ensemble methods' average accuracy rates ($F = 19.83, p < 1.56E - 06$). Our ANOVA results for ensemblesize selection reject the null hypothesis that ensemblesize and method are independent ($F = 3.22, p < 0.012$). And also like our Hyperplane Tukey tests, the MPC method was found to have the highest average accuracy rate of the three techniques, and the AUE method the lowest. Ensemblesize interaction analysis also showed that ensemblesize selection influenced average classification rates of the methods. The ensemble methods all appeared to benefit from additional classifiers while processing the SEA data set, and similarly to our Hyperplane resource analysis we now follow up by looking at how efficient the techniques were for this data set.

4.1.8 SEA Data Set Efficiency Results

Figure 4.14 then shows us the memory requirements each of the classifier ensembles has while processing the SEA data set. Similar to the Hyperplane results we examined earlier, the AUE ensemble appears to be affected by both the size of data chunks and the size of ensemble membership.

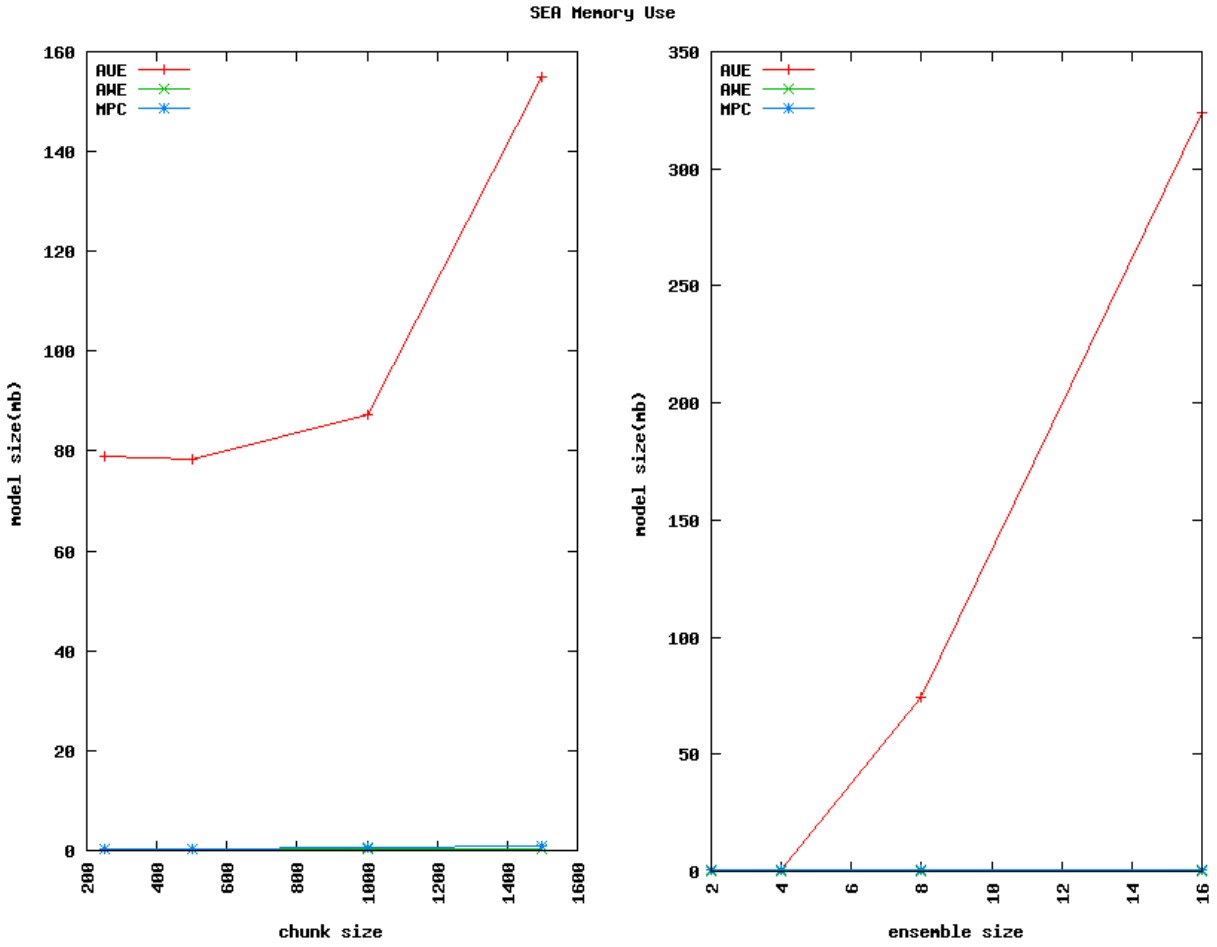

Figure 4.14: Ensemble Model Size for AWE, MPC, & AUE methods on SEA Data Set.

We examine the resource requirements of the different ensemble methods in Figure 4.14, and can see that the AUE method appears to require 80-150 times the memory of all the other ensemble methods for the data chunk manipulation trials. Table 4.3 and Table 4.4 appear to back this up. The ensemble method that had the lowest average memory requirement for the SEA data set was the AWE technique, and the ensemble method that appears to require the most amount of memory was the AUE method. For the third very large data set we tested the AUE method with, we see its memory to be affected by the size of the data stream.

The SEA training and testing times for each ensemble are plotted within Figure 4.15 and Figure 4.16. These graphs again show the training and testing time requirements for the AUE algorithm

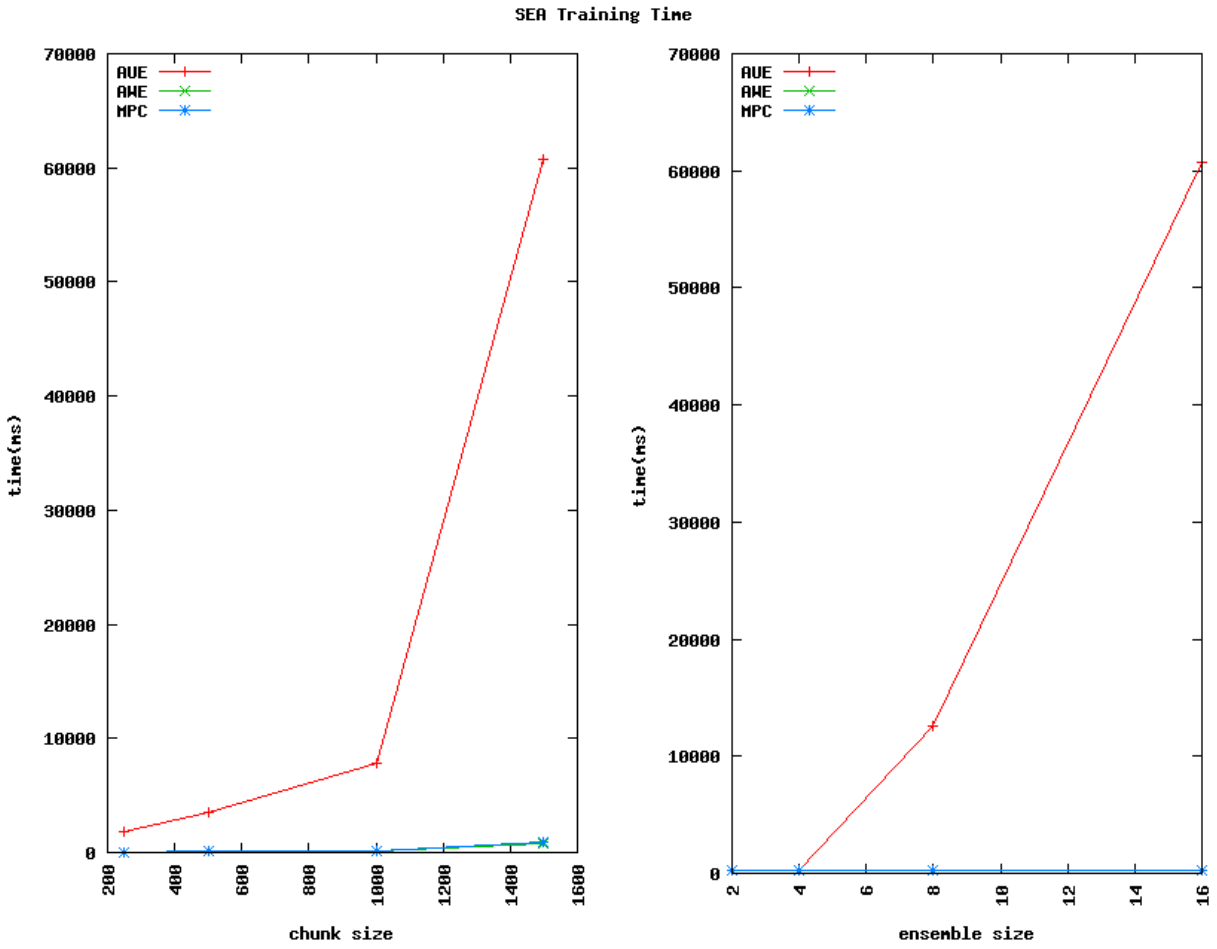

Figure 4.15: Training Times for AWE, MPC, & AUE methods on SEA Data Set.

to be larger than the other ensemble techniques. Table 4.5 and Table 4.6, show AUE to require the most time for training, and AWE the least. Table 4.7 and Table 4.8 show MPC to have the best testing times, and again AUE to require the most time for training while processing the SEA data set. Even though the AUE method had the highest average testing times of all the ensemble methods, the testing time measurements do not appear to be affected by the size of data stream being processed. The testing times actually declined some between the Hyperplane measurements and the SEA results, even though the SEA data set's size is four times that of the Hyperplane's.

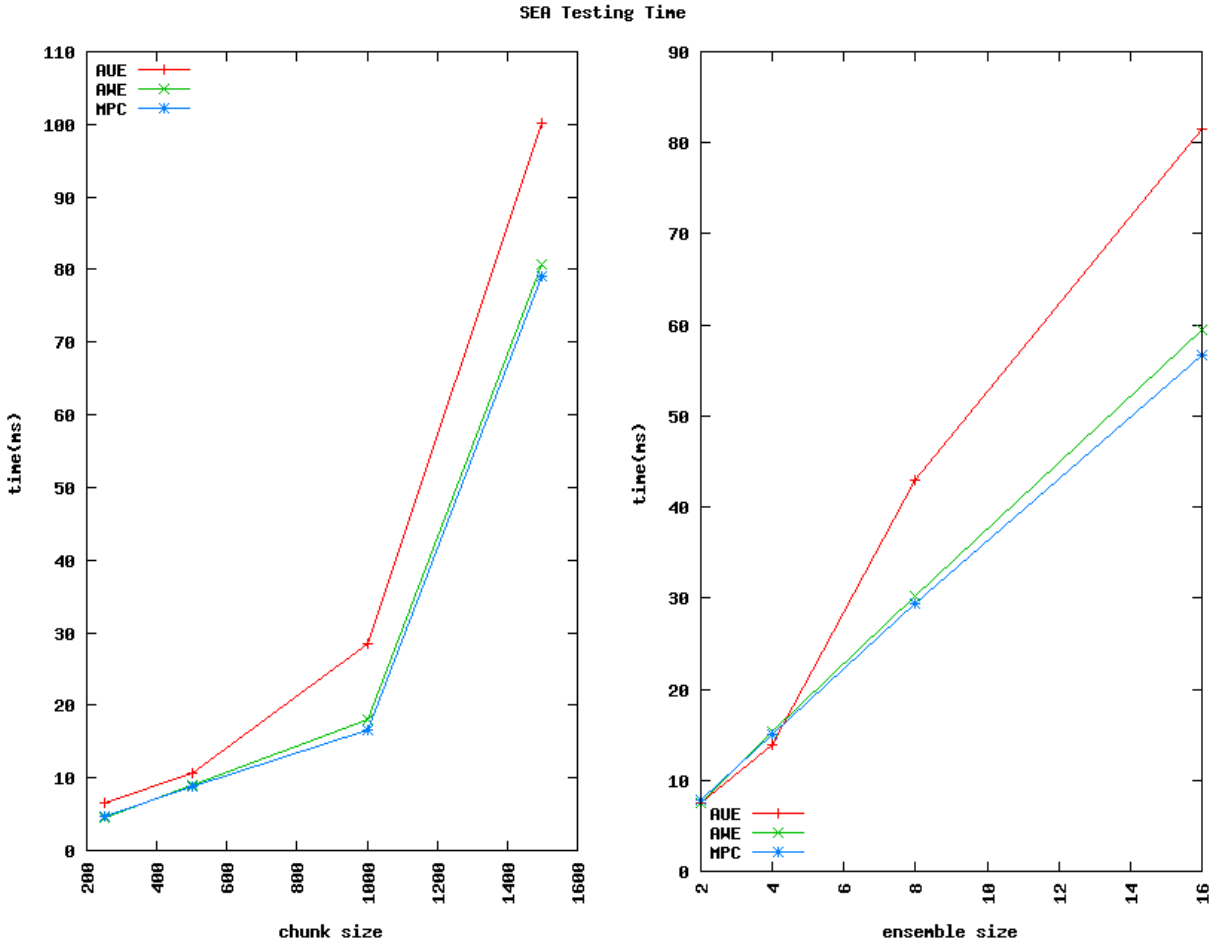

Figure 4.16: Testing Times for AWE, MPC, & AUE methods on SEA Data Set.

4.1.9 Summary of Existing Ensemble Method Results

Our first hypothesis was that of the three existing ensemble methods (AWE, MPC, & AUE), one would clearly outperform the rest. Our experimental results were not able to distinguish a statistically significant leader for all of the data sets we used within our experiments. The Australian Electricity data set, the LED data set, and the Hyperplane data set analysis all showed the MPC, AWE, and AUE methods' average accuracy rates to be statistically inseparable. The SEA data set analysis did show the MPC method to outperform both AWE, and AUE, but we cannot claim the MPC method to clearly outperform the other methods just because it was statistically significant for one data set. Because of this, we say that our experimental data does not support the claim of our first hypothesis: that one ensemble technique will clearly outperform the rest.

Though we may not be able to distinguish a clear performance leader between the methods, we can certainly provide additional characteristics of each algorithm that we learned as a result of the experiment. We observed the AUE method's accuracy and training/testing times were inversely influenced by the chunksize and ensemblesize selections for several data sets. We found chunksize selection to provide both a positive and a negative influence on classifier accuracy for some of the methods, yet only on some of the data sets. We found ensemblesize selection to only influence classifier accuracy in a positive manner when additional classifiers where added to the ensemble membership. Unlike Masud et al. that identified ensemblesize "sweet spots" in their experiments [12], we were not able to reproduce this in our studies. This might be attributed to just not using enough classifiers within our ensemble membership to begin seeing a negative impact upon performance.

Another observation we received from our ANOVA analysis is how the two independent variables interact with our methods we tested. Whereas the prior experiments with AWE and MPC looked at the influences chunksize and ensemblesize selection have on performance, they do not appraise the influence an interaction between the factors and the methods may have on performance. In our examinations of the existing methods we only observed an interaction between chunksize selection and method on the Electricity data set, and an interaction between ensemblesize and method on the SEA data set, but none the less, we cannot ignore their influence. It appears that for some data sets, and for some ensemble methods, there are combinations of method and chunksize/ensemblesize selections that influence classifier performance.

As we complete our comprehensive examination of the three state of the art ensemble techniques, we begin to raise additional research questions about our observations and analysis. These questions surround the chunksize and ensemblesize selection interactions we observed between several of the methods. We would also be interested in learning about combining mixtures of ensembles, and techniques for determining when one ensemble may be more beneficial to the system than another.

Table 4.1: Classification Accuracy vs. Chunksize for AWE, MPC, & AUE methods.

$Algorithm_{ChunkSize}$	Electricity	LED	Hyperplane	SEA
AUE_{250}	72.31 ±0.94	72.03 ±2.10	82.64 ±2.51	82.94 ±2.87
AUE_{500}	69.66 ±0.77	73.56 ±0.49	80.54 ±1.12	82.23 ±3.19
AUE_{1000}	69.75±0.25	73.94 ±0.04	83.20 ±2.12	84.06 ±2.38
AUE_{1500}	66.82 ±0.18	73.98 ± < 0.005	85.12 ±1.45	85.19 ±1.44
$AUE_{avg.}$	**69.64** ±2.24	*73.38* ±0.092	*82.88* ±1.88	*83.61* ±1.30
AWE_{250}	71.79 ±0.28	73.50 ±0.84	84.82 ±2.00	85.06 ±2.00
AWE_{500}	69.12±0.27	73.93 ±0.09	84.59 ±2.53	84.71 ±2.53
AWE_{1000}	69.55 ±0.08	73.98 ±0.01	84.80 ±1.13	84.91 ±1.15
AWE_{1500}	66.72 ±0.08	73.97 ±0.02	85.52 ±0.55	85.62 ±0.59
$AWE_{avg.}$	69.30 ±2.01	73.85 ±0.23	84.93 ±0.41	85.01 ±0.39
MPC_{250}	70.17 ±2.10	73.57 ±0.72	85.90 ±1.11	86.98 ±1.09
MPC_{500}	69.67 ±0.64	73.95 ±0.05	86.46 ±1.07	85.69 ±0.99
MPC_{1000}	68.29 ±0.83	73.96 ±0.007	85.15 ±0.53	87.28 ±0.49
MPC_{1500}	67.16 ±0.38	73.98 ±0.006	85.51 ±0.36	88.06 ±0.35
$MPC_{avg.}$	*68.83* ±1.36	**73.87** ±0.20	**85.76** ±0.56	**87.00** ±0.99

Table 4.2: Classification Accuracy vs. Ensemblesize for AWE, MPC, & AUE methods.

$Algorithm_{EnsembleSize}$	Electricity	LED	Hyperplane	SEA
AUE_2	69.07 ±2.02	72.52 ±2.19	81.39 ±1.87	81.38 ±1.89
AUE_4	69.36 ±1.96	73.21 ±1.21	81.90 ±1.73	81.89 ±1.76
AUE_8	69.87 ±2.35	73.81 ±0.28	83.35 ±1.74	84.51 ±1.46
AUE_{16}	70.24 ±2.68	73.96 ± < 0.008	84.86 ±3.00	86.65 ±0.24
$AUE_{avg.}$	**69.64** ±0.52	73.38 ±0.66	*82.88* ±1.56	*83.61* ±2.45
AWE_2	69.04 ±1.97	73.50 ±0.95	82.88 ±1.49	82.99 ±1.45
AWE_4	69.35 ±2.13	73.94 ±0.09	84.65 ±0.62	84.81 ±0.57
AWE_8	69.38 ±2.10	73.98 ± < 0.004	85.80 ±0.21	85.96 ±0.24
AWE_{16}	69.41 ±2.13	73.96 ± < 0.003	86.40 ±0.63	86.55 ±0.64
$AWE_{avg.}$	69.30 ±0.17	73.85 ±0.23	84.93 ±1.56	85.10 ±1.57
MPC_2	67.78 ±0.95	73.57 ±0.73	84.74 ±0.35	86.04 ±0.42
MPC_4	68.83 ±1.33	73.94 ±0.07	85.62 ±0.46	87.04 ±0.56
MPC_8	69.12 ±1.81	73.97 ±0.02	86.16 ±0.72	87.34 ±0.79
MPC_{16}	69.56 ±2.19	73.97 ± < 0.007	86.50 ±0.91	87.58 ±0.94
$MPC_{avg.}$	*68.82* ±0.76	**73.86** ±0.20	**85.76** ±0.77	**87.00** ±0.68

Table 4.3: Model Size vs. Chunksize for AWE, MPC, & AUE methods.

$Algorithm_{ChunkSize}$	Electricity	LED	Hyperplane	SEA
AUE_{250}	0.44	0.63	24.13	79.05
AUE_{500}	0.48	0.63	19.54	78.40
AUE_{1000}	0.37	4.12	22.32	87.35
AUE_{1500}	0.27	9.64	31.22	154.81
$AUE_{avg.}$	*0.39* ±0.09	*3.76* ±4.25	*24.30* ±4.98	*99.90* ±36.83
AWE_{250}	0.24	0.63	0.26	0.26
AWE_{500}	0.25	0.63	0.27	0.26
AWE_{1000}	0.24	0.63	0.35	0.35
AWE_{1500}	0.19	0.56	0.35	0.35
$AWE_{avg.}$	**0.23** ±0.03	**0.61** ±0.04	**0.31** ±0.05	**0.31** ±0.05
MPC_{250}	0.27	0.75	0.33	0.33
MPC_{500}	0.30	0.87	0.41	0.41
MPC_{1000}	0.24	1.12	0.62	0.62
MPC_{1500}	0.19	1.36	0.76	0.76
$MPC_{avg.}$	0.25 ±0.05	1.03 ±0.27	0.53 ±0.20	0.53 ±0.20

Table 4.4: Model Size vs. Ensemblesize for AWE, MPC, & AUE methods.

$Algorithm_{EnsembleSize}$	Electricity	LED	Hyperplane	SEA
AUE_2	0.29	0.63	0.34	0.34
AUE_4	0.35	0.63	0.51	0.52
AUE_8	0.43	1.62	16.77	74.49
AUE_{16}	0.49	12.15	79.59	324.27
$AUE_{avg.}$	*0.39* ±0.09	*3.76* ±5.61	*24.30* ±37.66	*99.91* ±153.60
AWE_2	0.23	0.63	0.31	0.31
AWE_4	0.23	0.63	0.31	0.31
AWE_8	0.23	0.63	0.31	0.31
AWE_{16}	0.23	0.56	0.31	0.31
$AWE_{avg.}$	**0.23** ±0.00	**0.61** ±0.04	**0.31** ±0.00	**0.31** ±0.00
MPC_2	0.25	1.03	0.53	0.53
MPC_4	0.25	1.03	0.53	0.53
MPC_8	0.25	1.03	0.53	0.53
MPC_{16}	0.25	1.03	0.53	0.53
$MPC_{avg.}$	0.25 ±0.00	1.03 ±0.00	0.53 ±0.00	0.53 ±0.00

Table 4.5: Training Time vs. Chunksize for AWE, MPC, & AUE methods.

$Algorithm_{ChunkSize}$	Electricity	LED	Hyperplane	SEA
AUE_{250}	830.80	99.33	577.95	1789.35
AUE_{500}	2870.45	204.73	1161.90	3544.52
AUE_{1000}	9546.30	761.61	2074.77	7867.97
AUE_{1500}	17166.67	5256.48	12266.86	60685.63
$AUE_{avg.}$	*7603.56* ±7382.41	*1580.54* ±2467.79	*4020.37* ±5532.06	*18471.87* ±28258.19
AWE_{250}	821.59	85.904	45.67	44.25
AWE_{500}	2661.36	168.97	88.35	85.24
AWE_{1000}	8796.30	320.64	180.58	165.97
AWE_{1500}	15791.67	1498.90	813.24	752.62
$AWE_{avg.}$	**7017.73** ±6770.41	**518.60** ±660.718	281.96 ±358.63	**262.02** ±330.95
MPC_{250}	898.86	95.20	47.87	52.88
MPC_{500}	2740.91	175.18	91.98	93.01
MPC_{1000}	8842.59	346.72	175.67	175.30
MPC_{1500}	16958.33	1702.38	799.00	861.37
$MPC_{avg.}$	7360.17 ±7243.58	579.87 ±755.66	**278.63** ±350.94	295.64 ±380.58

Table 4.6: Training Time vs. Ensemblesize for AWE, MPC, & AUE methods.

$Algorithm_{EnsembleSize}$	Electricity	LED	Hyperplane	SEA
AUE_2	6333.54	594.70	286.09	265.15
AUE_4	7940.07	611.48	296.69	282.38
AUE_8	8114.29	1036.04	3444.05	12588.77
AUE_{16}	8026.31	4079.92	11054.66	60751.17
$AUE_{avg.}$	*7603.55* ±849.66	*1580.54* ±1678.72	*3770.37* ±5078.52	*18471.87* ±28777.84
AWE_2	7190.43	509.78	278.05	260.21
AWE_4	7310.23	552.55	285.35	267.99
AWE_8	6256.54	534.90	286.31	262.30
AWE_{16}	7313.72	477.18	278.14	257.58
$AWE_{avg.}$	**7017.73** ±510.69	**518.60** ±32.72	281.96 ±4.48	**262.02** ±4.42
MPC_2	7496.63	551.18	274.31	306.52
MPC_4	7201.96	560.13	294.20	300.39
MPC_8	7336.68	597.05	265.85	294.71
MPC_{16}	7405.43	611.11	280.16	280.94
$MPC_{avg.}$	7360.18 ±124.17	579.87 ±28.77	**278.63** ±11.93	295.64 ±10.92

Table 4.7: Testing Time vs. Chunksize for AWE, MPC, & AUE methods.

$Algorithm_{ChunkSize}$	Electricity	LED	Hyperplane	SEA
AUE_{250}	78.98	11.49	6.26	6.47
AUE_{500}	261.36	25.90	5.08	10.73
AUE_{1000}	1092.59	55.87	22.03	28.41
AUE_{1500}	1895.83	288.48	119.62	100.11
$AUE_{avg.}$	*1895.83* ±835.13	*95.44* ±130.02	*38.25* ±54.80	*36.43* ±43.50
AWE_{250}	56.59	10.0	4.85	4.59
AWE_{500}	161.36	22.50	9.20	9.04
AWE_{1000}	638.89	40.35	19.09	18.11
AWE_{1500}	875.00	180.99	88.45	80.81
$AWE_{avg.}$	**432.96** ±388.69	**63.46** ±79.34	30.40 ±39.16	28.14 ±35.56
MPC_{250}	76.59	10.0	4.23	4.65
MPC_{500}	247.73	17.50	8.63	8.72
MPC_{1000}	944.44	37.84	15.63	16.60
MPC_{1500}	1895.83	192.77	75.72	79.10
$MPC_{avg.}$	791.148 ±826.58	64.53 ±86.30	**26.05** ±33.44	**27.27** ±34.91

Table 4.8: Testing Time vs. Ensemblesize for AWE, MPC, & AUE methods.

$Algorithm_{EnsembleSize}$	Electricity	LED	Hyperplane	SEA
AUE_2	225.74	20.70	8.22	7.50
AUE_4	595.16	40.99	16.36	13.81
AUE_8	1033.35	85.95	38.29	42.91
AUE_{16}	1474.52	234.11	90.10	81.50
$AUE_{avg.}$	*832.19* ±540.68	*95.44* ±96.39	*38.24* ±36.83	*36.43* ±33.77
AWE_2	285.40	17.50	8.13	7.55
AWE_4	398.26	37.39	16.55	15.29
AWE_8	479.08	70.35	32.53	30.26
AWE_{16}	569.11	128.60	64.38	59.44
$AWE_{avg.}$	**432.96** ±120.61	**63.46** ±48.59	30.4 ±24.81	28.14 ±22.90
MPC_2	336.32	13.71	7.01	7.87
MPC_4	505.01	35.00	14.85	15.11
MPC_8	896.55	69.15	26.05	29.48
MPC_{16}	1426.72	140.24	56.30	56.61
$MPC_{avg.}$	791.15 ±484.36	64.53 ±55.40	**26.05** ±21.63	**27.27** ±21.52

4.2 Contributed Methods

Our two contributed methods, the AMPE and AMPE2 algorithms, were inspired by the desire to learn how we may adapt to concept drift. Our AMPE method introduced a drift detection mechanism, and our AMPE2 algorithm is used to examine the importance of when to increase the training and testing phases according to concept drift presence. Our second hypothesis claimed the AMPE method would outperform the MPC method, and our third hypothesis said that the AMPE method would outperform the AMPE2 algorithm.

The results of our contributed methods compared to the MPC method are then organized similarly to our prior results and analysis of the three contributed ensemble methods. We examine the methods' performance per data set, and then by accuracy and efficiency of the algorithms.

4.2.1 Electricity Data Set Accuracy Results

Figure 4.17 graphically presents the accuracy results of running our two contributed methods and the MPC method while processing the Australian Electricity data set. The first observation we make of these results is that the AMPE2 algorithm appears to have performed significantly worse than all of the others. Its classification rate was seen to be roughly 30% less than the rest. Another

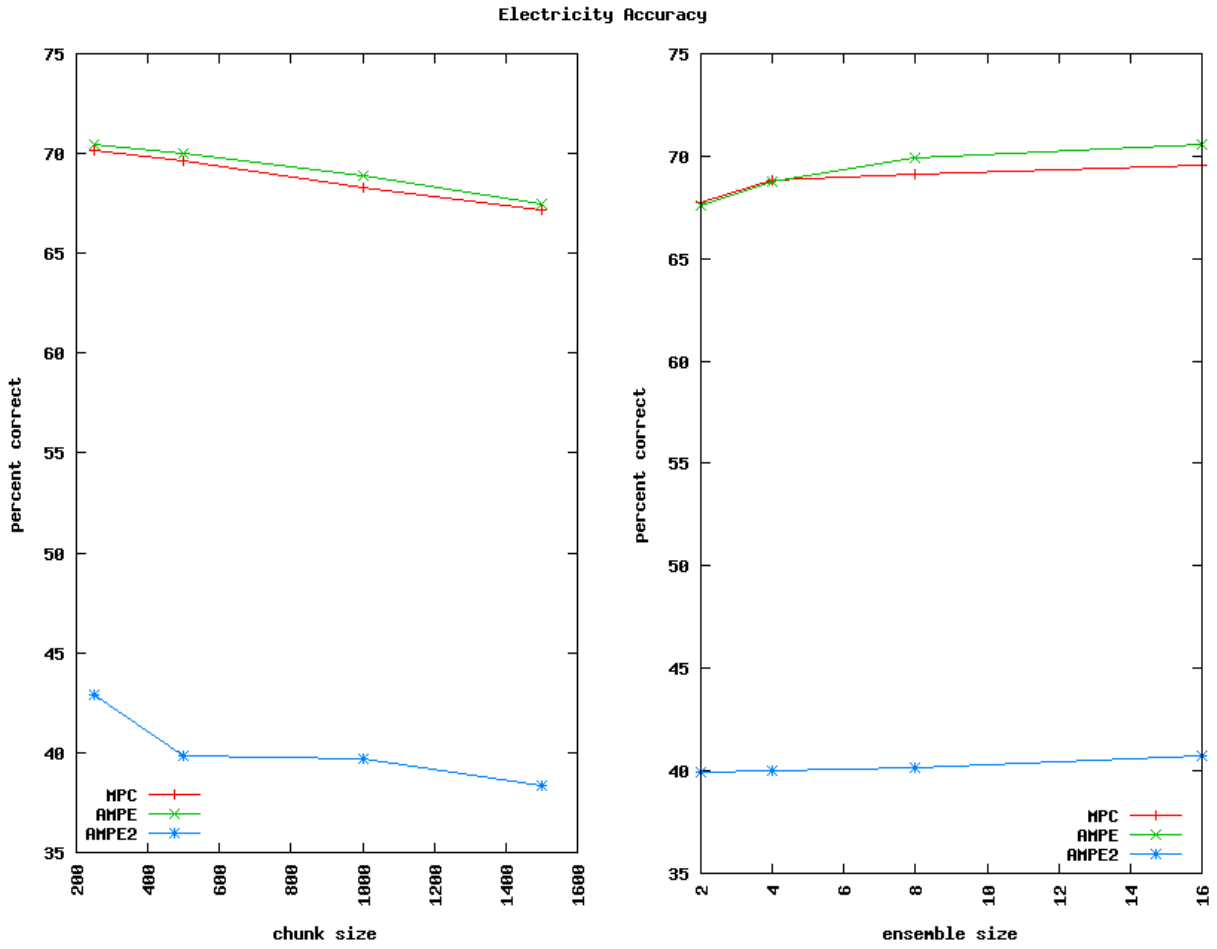

Figure 4.17: Accuracy for MPC, AMPE, & AMPE2 methods on Electricity Data Set.

observation that we make of these results is that it is difficult to distinguish a clearly superior ensemble method between the MPC and AMPE techniques. Table 4.9 and Table 4.10 show more detailed results of each of the experimental trial results, but the MPC and AMPE results are just too close. So we proceed with a set of two way ANOVA tests like we did within the examinations of the existing methods previously. And similar to our ANOVA tests earlier, we first examine the effects chunksize selection and method have on accuracy, and we then look at ensemblesize and method interaction. Our first two way ANOVA test soundly rejected that the average classification rates of the MPC, AMPE, and AMPE2 methods are equal($F = 3473.7, p < 6.61E-42$). Following this test up with a Tukey HSD multiple comparison, we determine that there is a significant statistical difference between the MPC and the AMPE2 methods, a difference between the AMPE

and AMPE2 techniques, but no identifiable difference between MPC and AMPE. These results show that the AMPE2 method did underperform the other two, but we cannot distinguish a true difference between the MPC and AMPE methods. Our first two way ANOVA test also showed chunksize selection to not make a difference in ensemble average accuracy rates for this data set ($F = 2.05, p > 0.609$). A test of factor independence failed to show a significant interaction between chunksize and method ($F = 1.23, p > 0.313$). The second two way ANOVA test examined the interaction that ensemblesize and method have on accuracy. This second ANOVA test also rejected that the methods' average classification rates are equal ($F = 1513.91, p < 1.82E - 35$), and there does not appear to be an interaction between ensemblesize and method ($F = 0.377, p > 0.889$). We follow this test with a Tukey test to examine which methods were statistically significant. The Tukey test for ensemblesize interaction found the MPC and AMPE methods to again outperform the AMPE2 technique, and it failed to show that ensemblesize selection affected average accuracy of the methods.

Our first observation was that our contributed AMPE2 method performed significantly worse than the others for this data set: to the tune of nearly a 30% difference. Our Tukey HSD tests did show significant differences between the MPC and AMPE2 methods, and the AMPE and AMPE2 algorithms. MPC was found to be approximately 28.625% better than the AMPE2 method, and our AMPE technique was determined to have an average classification rate that was 29.025% better than AMPE2. Our second observation was that due to the error rates, we could not clearly distinguish between MPC's performance and our AMPE method. We might attribute this to the small size of the data set, or there just may not be a performance distinction between these methods for this data set. We now take a look at the efficiency trade-offs of the three methods.

4.2.2 Electricity Data Set Efficiency Results

Our first observation we make of the memory use results we see in Figure 4.18 is that our AMPE method appears to require significantly more memory than both the MPC and the AMPE2 techniques. Examining Table 4.11 and Table 4.12 show the AMPE method to have the highest average model size of the three techniques. At times it can be seen that the AMPE method requires nearly

twice the memory of MPC and AMPE2. The AMPE method appears to be adapting and creating larger partitions, but the performance gains are not readily visible for this data set. We examine

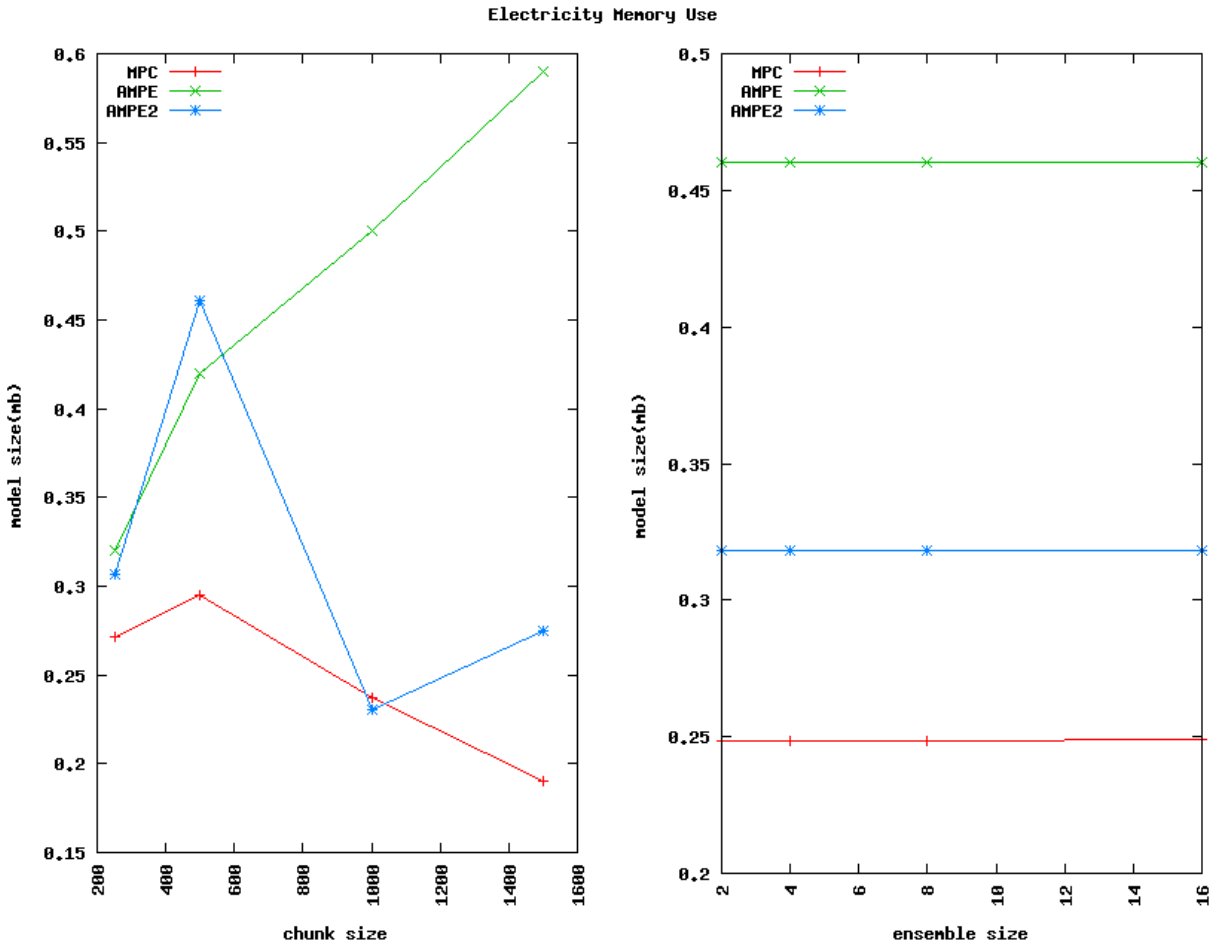

Figure 4.18: Ensemble Model Size for MPC, AMPE, & AMPE2 methods on Electricity Data Set.

both the Electricity data set training times in Figure 4.19, and the testing times within Figure 4.20 together. We observe that the AMPE algorithm appears to be influenced by chunksize, and we see that it is requiring up to two times the amount of time the other ensemble methods require while processing the Australian Electricity data set. Table 4.13, Table 4.14, Table 4.15, and Table 4.16 show the AMPE2 algorithm requires the least amount of time for testing and training while processing the Electricity data set.

The AMPE algorithm appears to take the longest of all the methods for training and testing

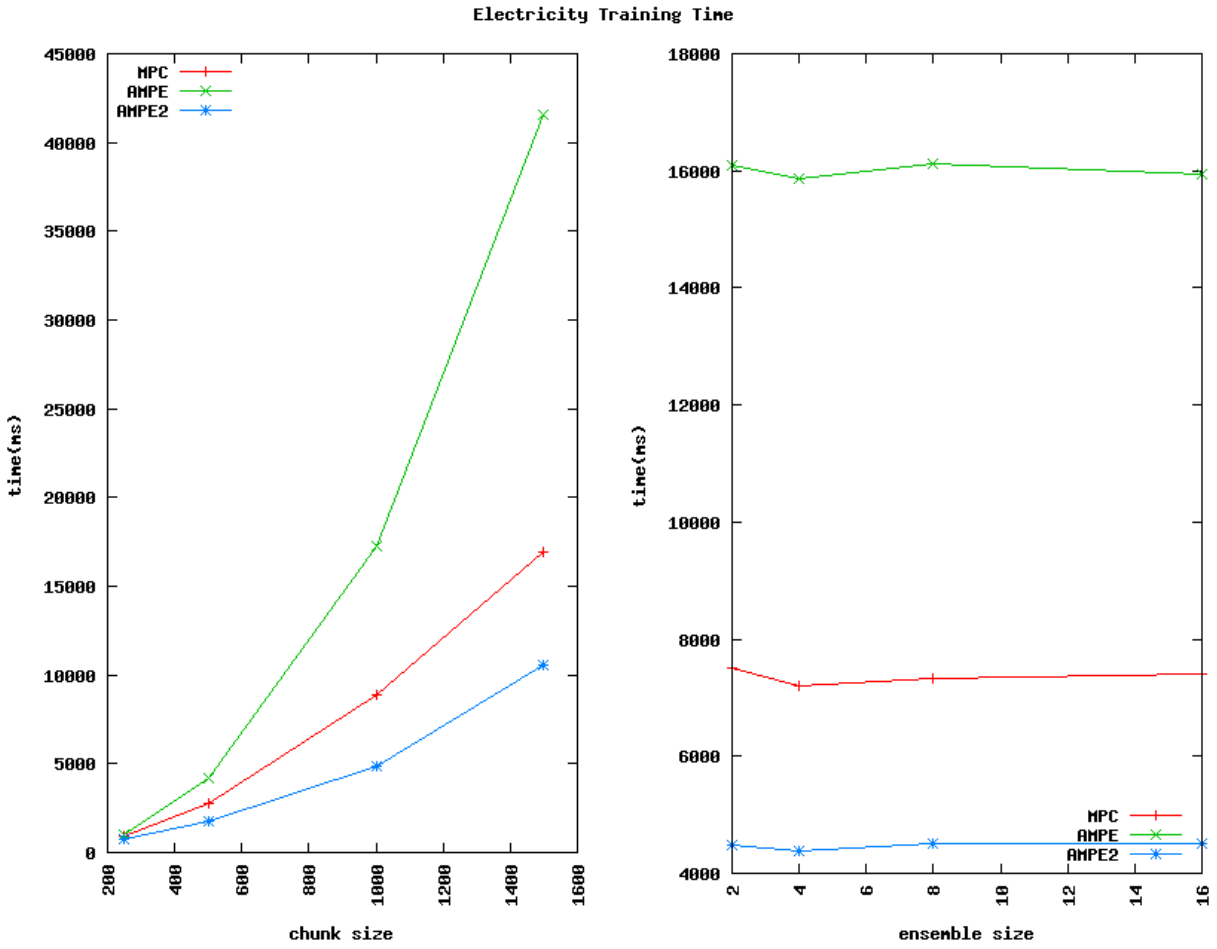

Figure 4.19: Training Times for MPC, AMPE, & AMPE2 methods on Electricity Data Set.

instances within the Electricity data set. Our AMPE method appears to take roughly two to seven times the MPC method to test and train with. From these measurements we conclude that our AMPE algorithm is not overly inefficient, but we still cannot justify the additional resource utilization given the performance measurements we observed within our analysis of ensemble accuracy for this particular data set.

4.2.3 LED Data Set Accuracy Results

Similarly to the Electricity Data Set results, Figure 4.21 appears to show the AMPE2 method significantly underperforming the other two techniques. Figure 4.21 also makes it difficult to dis-

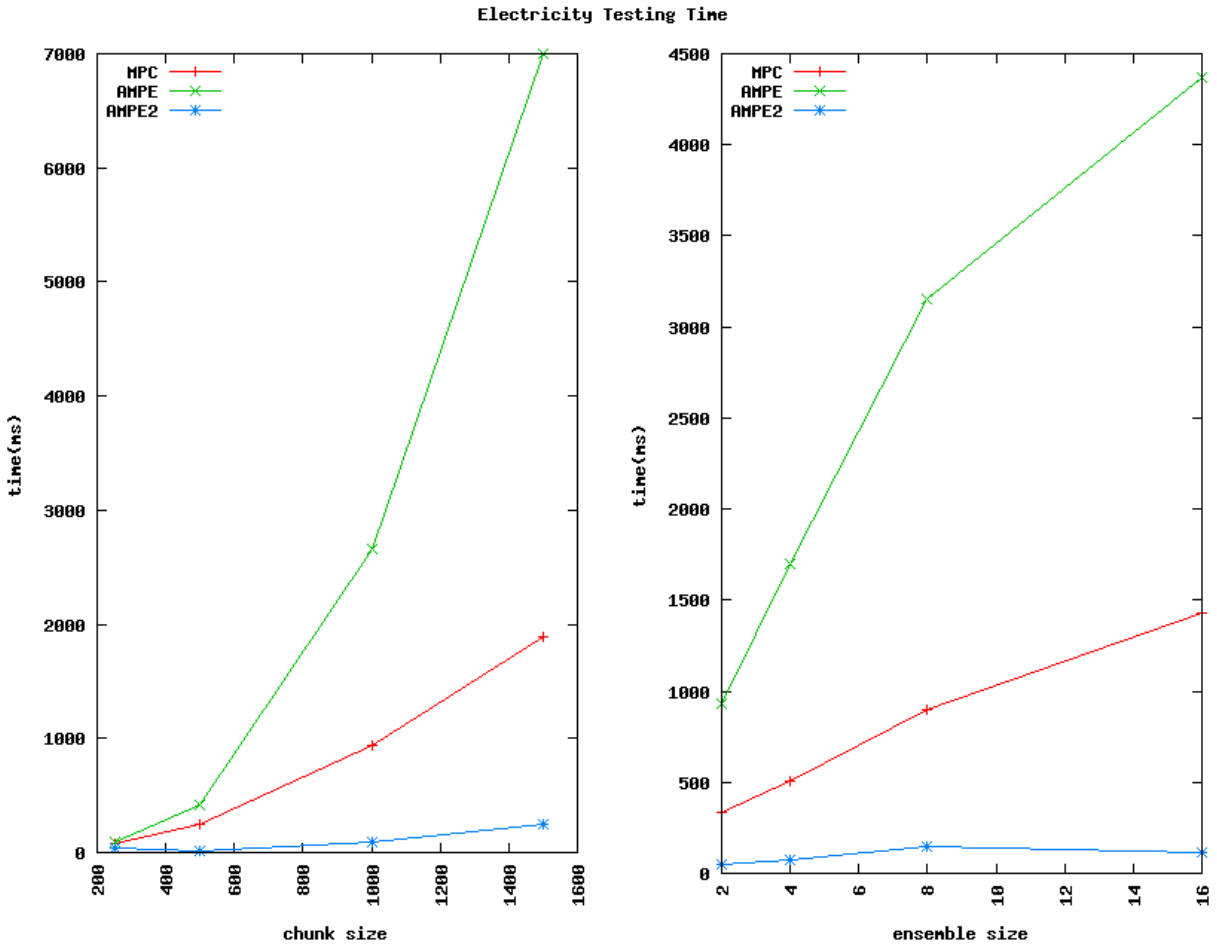

Figure 4.20: Testing Times for MPC, AMPE, & AMPE2 methods on Electricity Data Set.

tinguish between the AMPE and MPC techniques. The ensemblesize trials appear to make the performance of MPC and AMPE look identical. We examine Table 4.9 and Table 4.10 and see that the accuracy measurements are too close to distinguish between the methods. A set of follow up two way ANOVA tests reveal that there is a significant statistical difference between at least one of the methods' average accuracy rates. Both sets of ANOVA tests that examined chunksize versus method($F = 40.24, p < 6.62E-10$), and ensemblesize versus method($F = 29.35, p < 2.75E-08$) showed statistical differences between at least one of the methods. These two ANOVA tests also show chunksize($F = 2.33, p > 0.089$) and ensemblesize selection($F = 2.57, p > 0.0686$) to not influence average accuracy rates on this data set. A test of independence between chunksize and method shows that there is an interaction between them for these methods and data

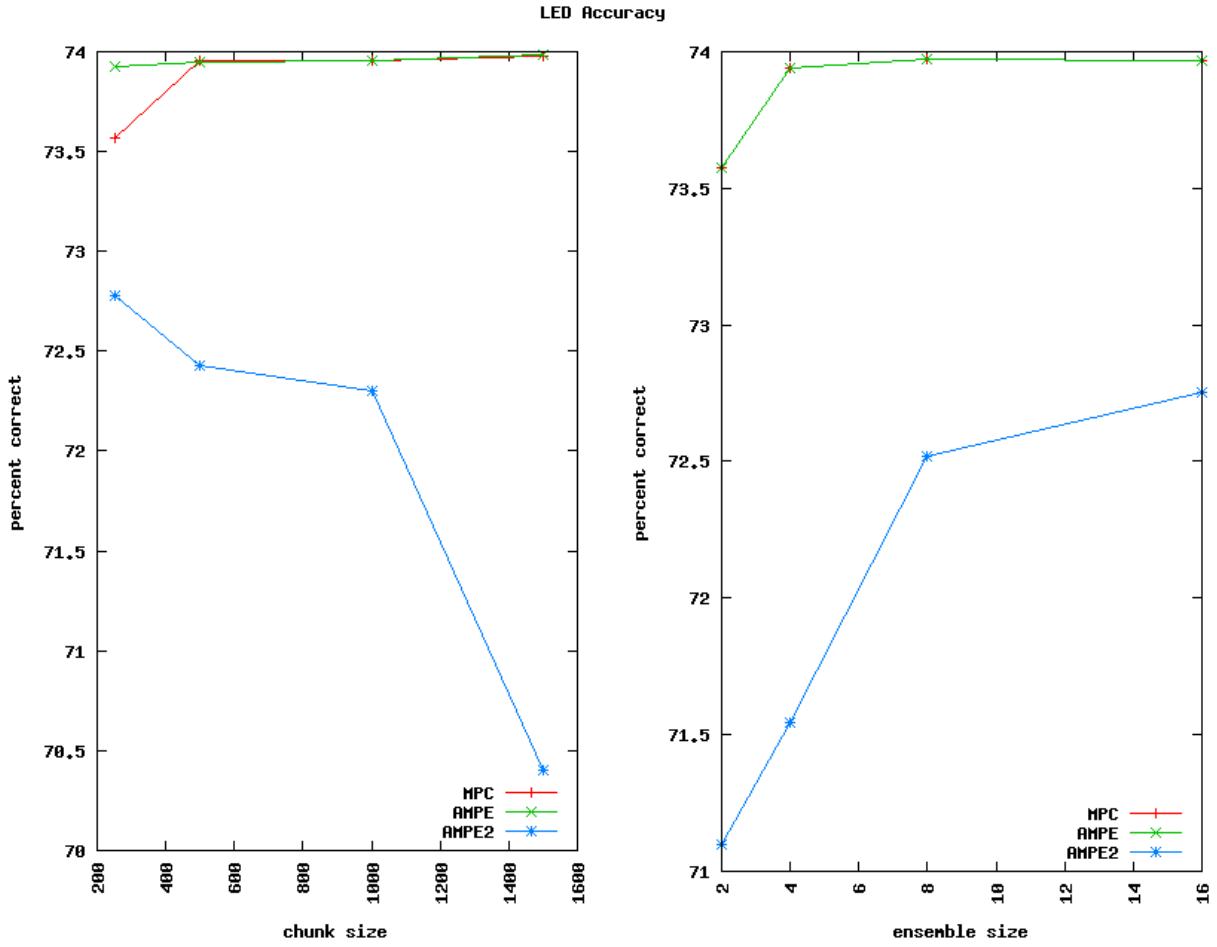

Figure 4.21: Accuracy for MPC, AMPE, & AMPE2 methods on LED Data Set.

set($F = 4.004, p < 0.0035$).

We follow up the statistically significant findings our ANOVA analysis yield with a set of Tukey HSD multiple comparison tests. Both sets of Tukey tests, for chunksize and ensemblesize factors, show a significant difference between the MPC and AMPE2 methods, as well as a difference between the AMPE and AMPE2 techniques. The MPC method was reported as performing approximately 1.885 percent better than the AMPE2 algorithm, and the AMPE technique performed similarly with a 1.8825 percent improvement over the AMPE2 method. These tests provide additional support to our initial suspicion that the AMPE2 method underperformed the others for the LED data set. What is surprising is to see that unlike the nearly 30% performance difference we saw between

these methods for the Electricity Data set, the differences we see between the MPC/AMPE2 and AMPE/AMPE2 methods for the LED data set are less than 2%. We now examine the performance implications of the different methods to see if our performance gain is worth the performance penalty.

4.2.4 LED Data Set Efficiency Results

Figure 4.22 shows the average model size for each of the ensemble methods for the LED data set. This plot shows us that our AMPE method appears to be influenced by chunksize selection the most. The ensemblesize manipulation trials show a constant difference between methods, whereas the chunksize trials depict our AMPE method to be influenced by the size of chunk. We examine

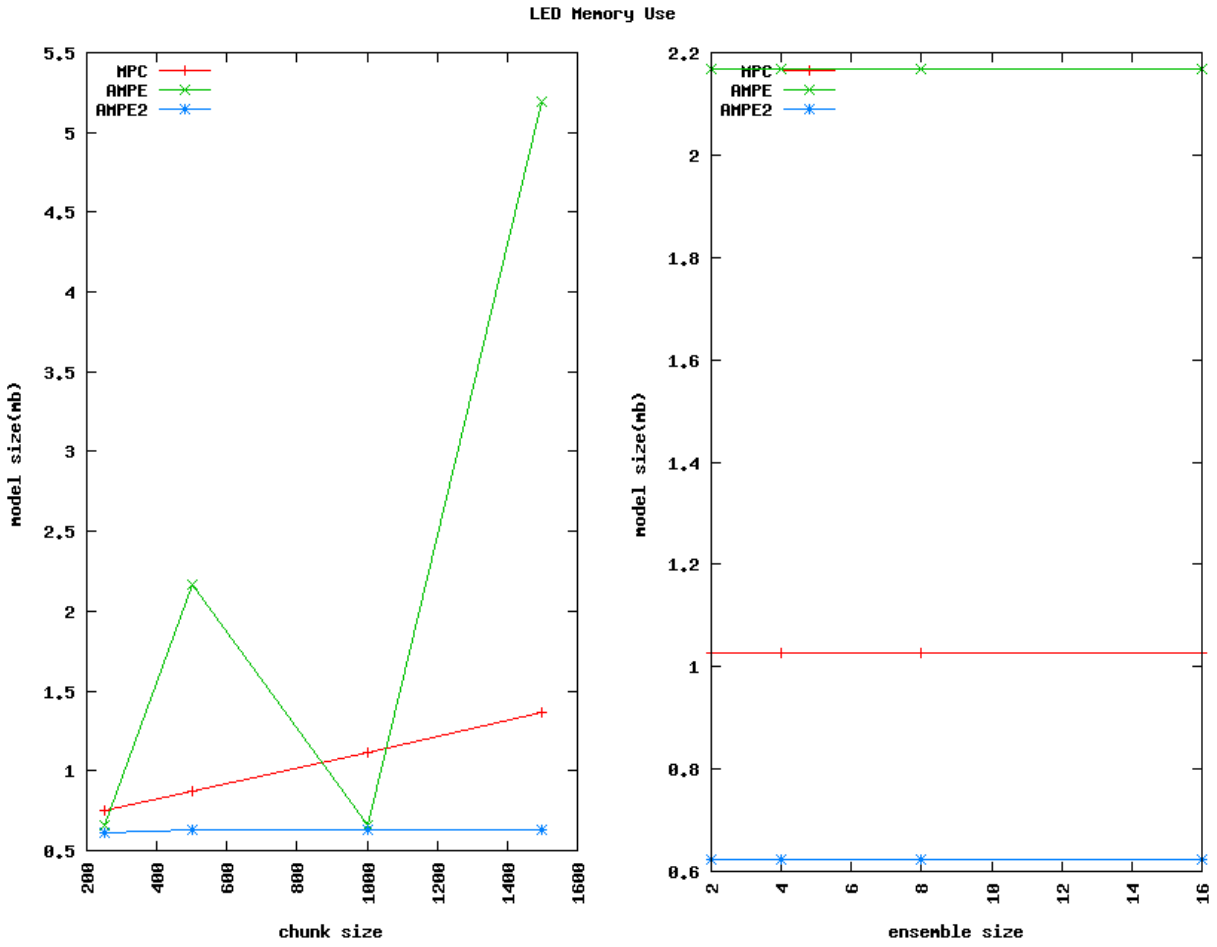

Figure 4.22: Ensemble Model Size for MPC, AMPE, & AMPE2 methods on LED Data Set.

the results within Table 4.11 and Table 4.12 and we observe that our AMPE method appears to require the most memory, nearly twice that of the MPC technique for both the chunksize and ensemblesize trials. The AMPE2 algorithm appears to have required the least amount of memory, but as we had seen within our prior accuracy results, it was also the least accurate of the three. And similarly to what we had seen within the Electricity Data set results, our AMPE algorithm

appears to require significantly more resources than the MPC method, but without an associated classification accuracy gain, just additional resource consumption.

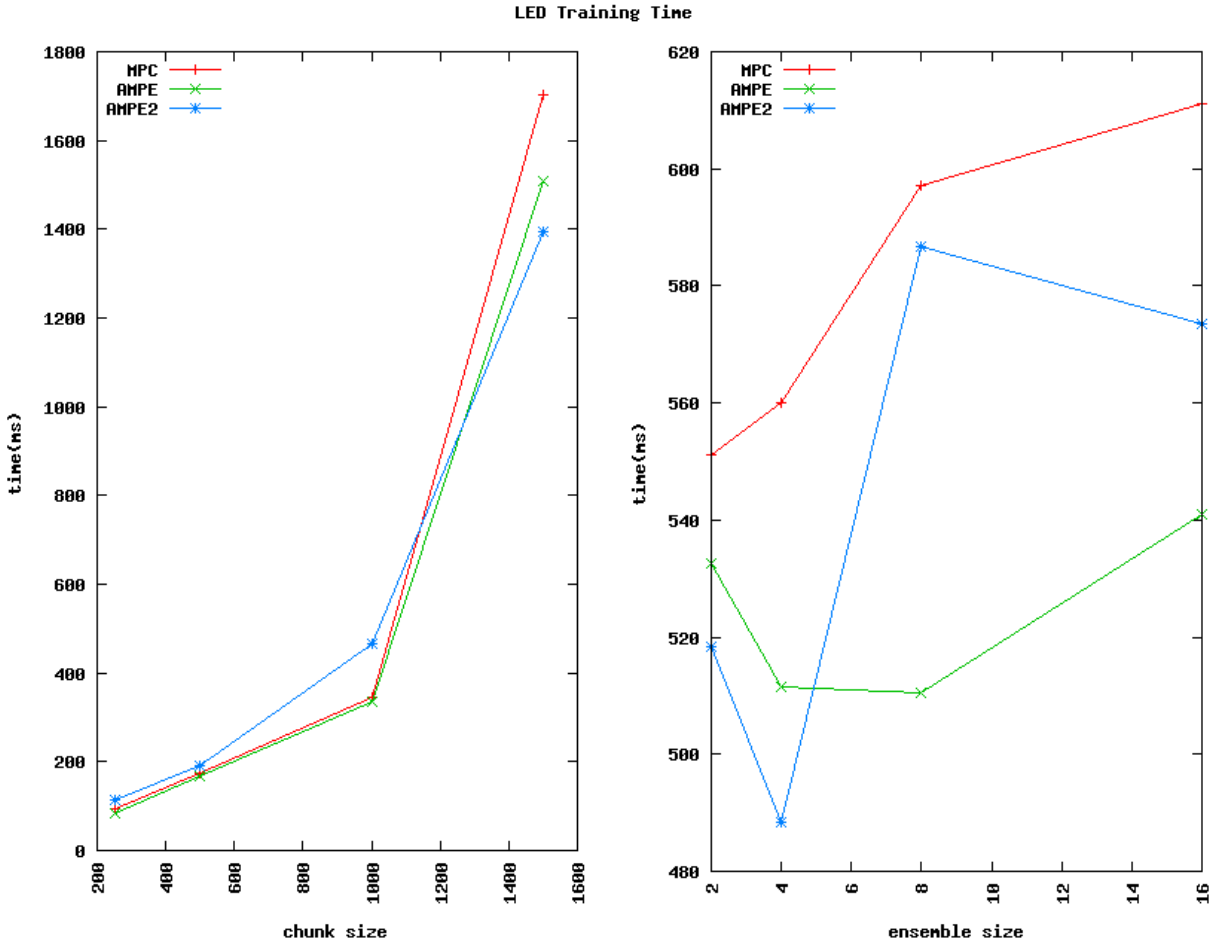

Figure 4.23: Training Times for MPC, AMPE, & AMPE2 methods on LED Data Set.

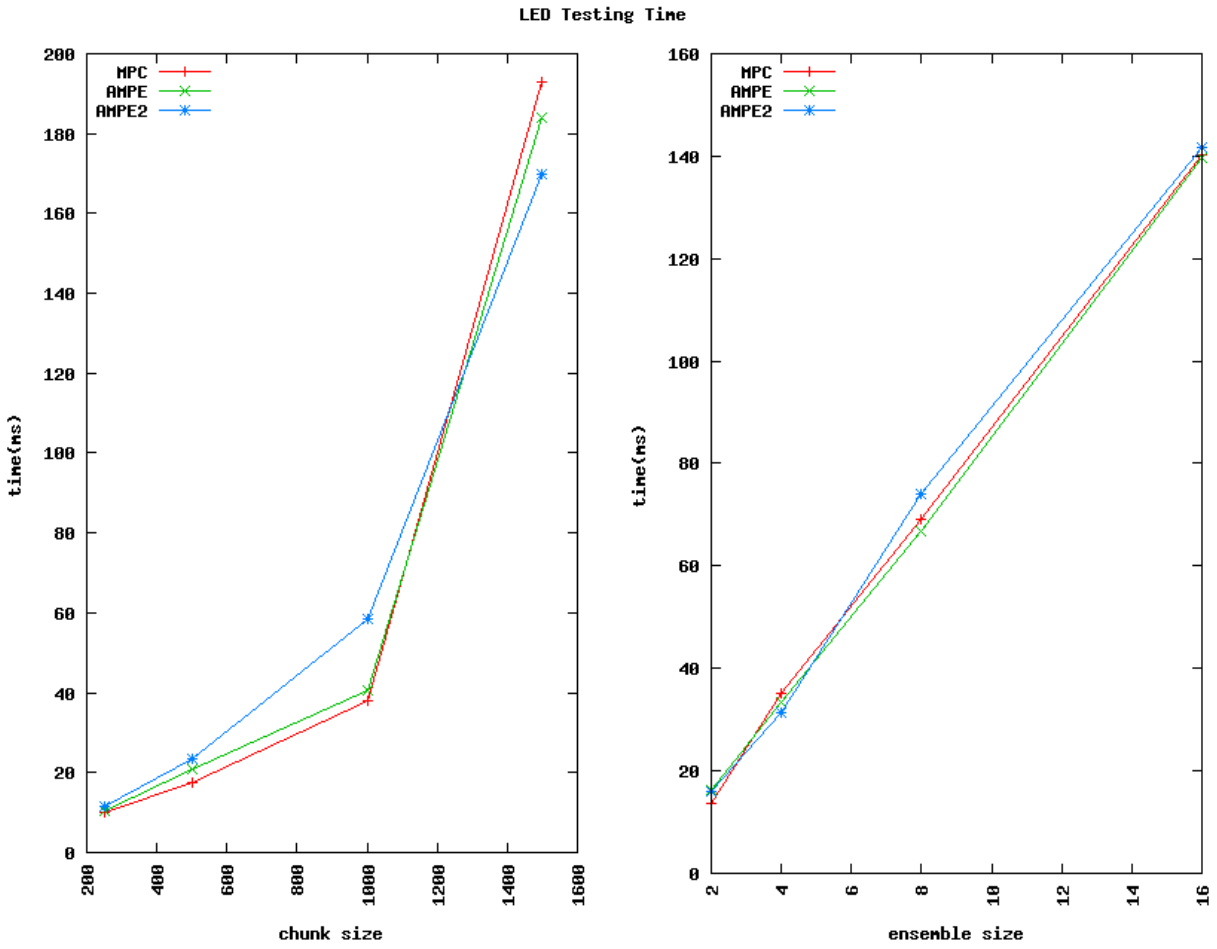

Figure 4.24: Testing Times for MPC, AMPE, & AMPE2 methods on LED Data Set.

The LED training and testing time measurements are plotted within Figure 4.23 and Figure 4.24. Unlike our earlier Electricity data set results that appeared to show significant difference in training and testing times of the three methods, the LED data set results for training and testing times appear to have little variation between the methods. Table 4.13, and Table 4.14 actually show our AMPE method to have the lowest average training times. They also show the MPC method to have the highest average training times, but like we have seen within our prior analysis, the errors of the measurements overlap, and we cannot distinguish a true statistical difference between the methods' average training times. The testing time results are similar, and can be seen within Table 4.15 and Table 4.16. We decide not to pursue further algorithm efficiency analysis at this point, and rely more on what we have learned within our accuracy results. We next examine our

contributed methods and the MPC method while processing the Rotating Hyperplane data set.

4.2.5 Rotating Hyperplane Data Set Accuracy Results

Figure 4.25 shows the average accuracy results of the two contributed methods and the MPC technique. Again, the MPC and the AMPE methods' average accuracy rates appear to be statistically insignificant. And again, the AMPE2 method appears to have underperformed the other two. We

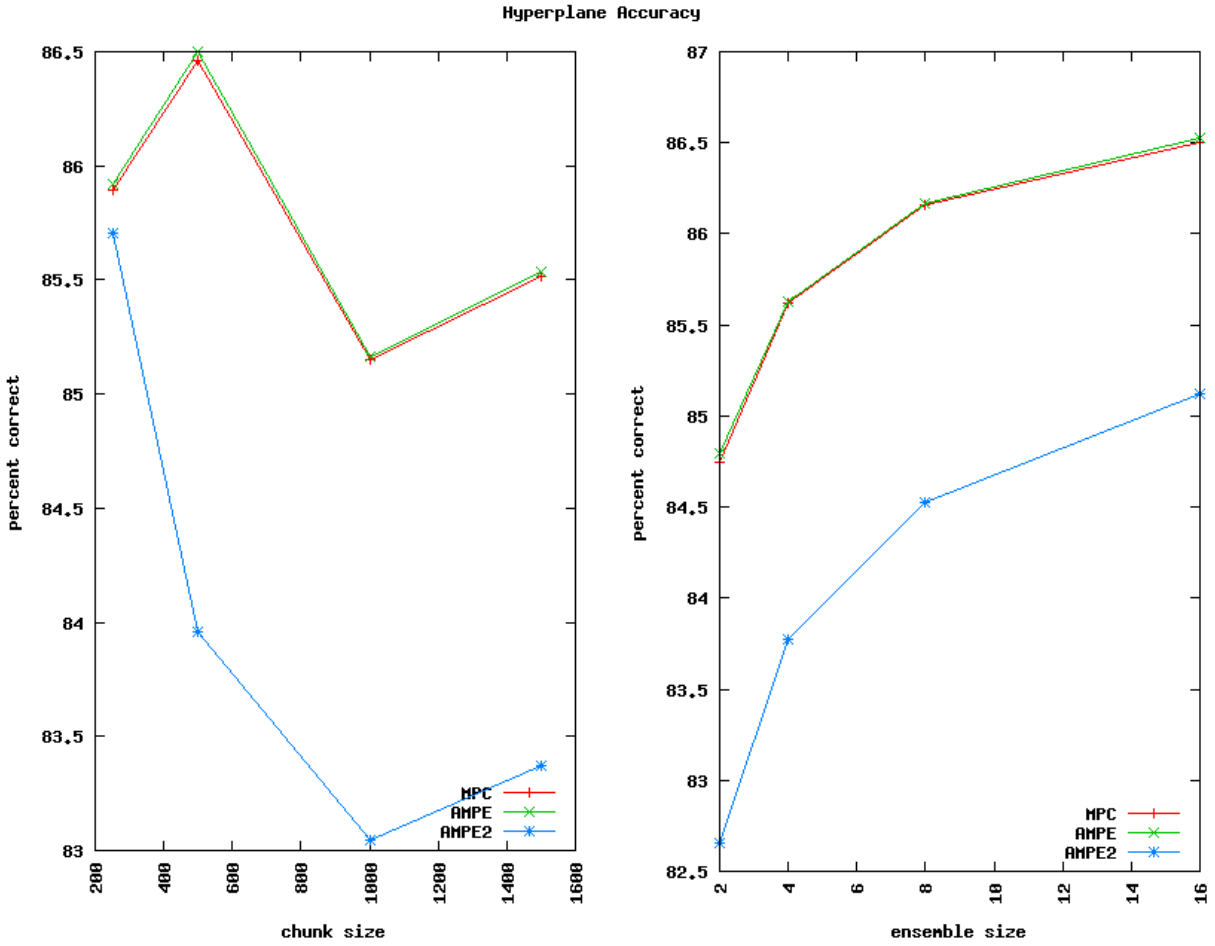

Figure 4.25: Accuracy for MPC, AMPE, & AMPE2 methods on Hyperplane Data Set.

also observe in Figure 4.25 that chunksize selection appears to have an influence on both the MPC and AMPE techniques when chunksize = 500. Recall in our examination of the AWE, MPC, and AUE methods, found in Figure 4.9, we observed the MPC method to show an accuracy improvement when chunksize selection was equal to 500, and we saw the AWE method's accuracy decline

at the chunksize selection of 500. And after seeing these possible interactions we began asking questions about how we may combine ensemble memberships to increase accuracy by using the optimal chunksize of the differing methods. And again we see possible interactions of chunksize and method within the Hyperplane results for the MPC, AMPE, and AMPE2 techniques. The results within Figure 4.25 show both the MPC and AMPE methods improve performance when chunksize selection equals 500. It also shows a diminished accuracy when chunksize selection is set to 1000. This tells us that accuracy increases/decreases associated with chunksize selection is not just monotonic in performance gains, but there appears to be "sweet spots" for chunksize parameter selection in some data sets.

If we examine Table 4.9 and Table 4.10 we see the AMPE method to have the highest average accuracy rate in both the chunksize and ensemblesize trials. The results within the tables also show that due to the error there may not be a statistical difference between the AMPE and MPC methods. We follow this up with a couple of two way ANOVA analyses and we find, while examining the interactions of both chunksize and ensemblesize, that there is a statistical difference between one of the ensemble techniques($F = 18.92, p < 2.42E-06$). And in fashion, we follow this test up with a Tukey HSD multiple comparison test to determine which methods were statistically separable. Our Tukey tests showed a significant difference in accuracy rates between the MPC and AMPE2 methods, as well as a difference between AMPE and AMPE2 methods, but not between the MPC and AMPE techniques. These results indicate that the AMPE2 method did underperform the other two, but we cannot statistically distinguish between the MPC and AMPE methods' average classification accuracy rates. Our first two way ANOVA analysis that examined chunksize and method found chunksize selection not to influence accuracy rate($F = 6.11, p > 0.182$), and there does not appear to be an interaction between chunksize and method ($F = 1.69, p > 0.151$). Although our examinations of Figure 4.25 show chunksize selection to both improve and hinder accuracy, our statistical tests do not. This might be attributed to the significant differences seen between our AMPE2 methods results and the other methods we used within our ANOVA analysis.

Similar to our prior examinations of ensemblesize selection influence, we examine our results of

our second ANOVA test and see that there is a statistical difference between some of the ensemblesizes' average accuracy rates. We follow up this finding with another Tukey HSD analysis, and determine a significant difference between accuracy rates of ensembles that had only 2 classifiers to those that have 16. The ensembles that had 16 classifiers within the ensemble population performed approximately 2.59% better than the ensembles with only 2 classifiers. These results are similar to what we had seen while comparing the existing methods (AWE, MPC, and AUE). And yet again we see what appears to be a monotonic increase in performance gains associated with ensemblesize increases. Yet chunksize selection appears to require more honing to establish. We next begin to compare the efficiency of our contributed methods to the MPC technique.

4.2.6 Rotating Hyperplane Data Set Efficiency Results

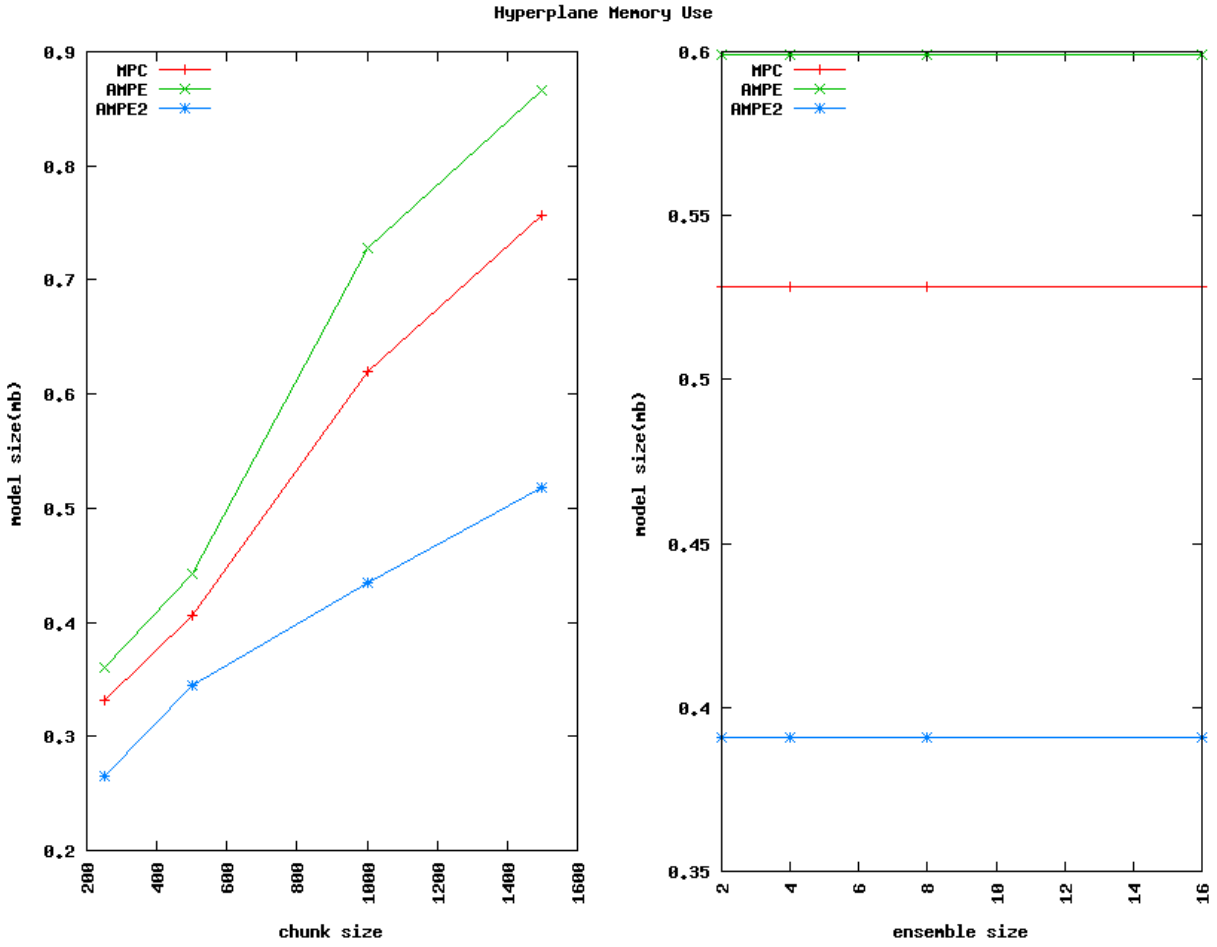

Figure 4.26: Ensemble Model Size for MPC, AMPE, & AMPE2 methods on Hyperplane Data Set.

The amount of memory consumed by the three methods can be seen within Figure 4.26. Our AMPE method appears to require the most memory in both the chunksize and ensemblesize experiments. Tables 4.11 and 4.12 show the AMPE method to have the highest average accuracy rate, and the AMPE2 method again appears to have the lowest accuracy of the methods. Figures

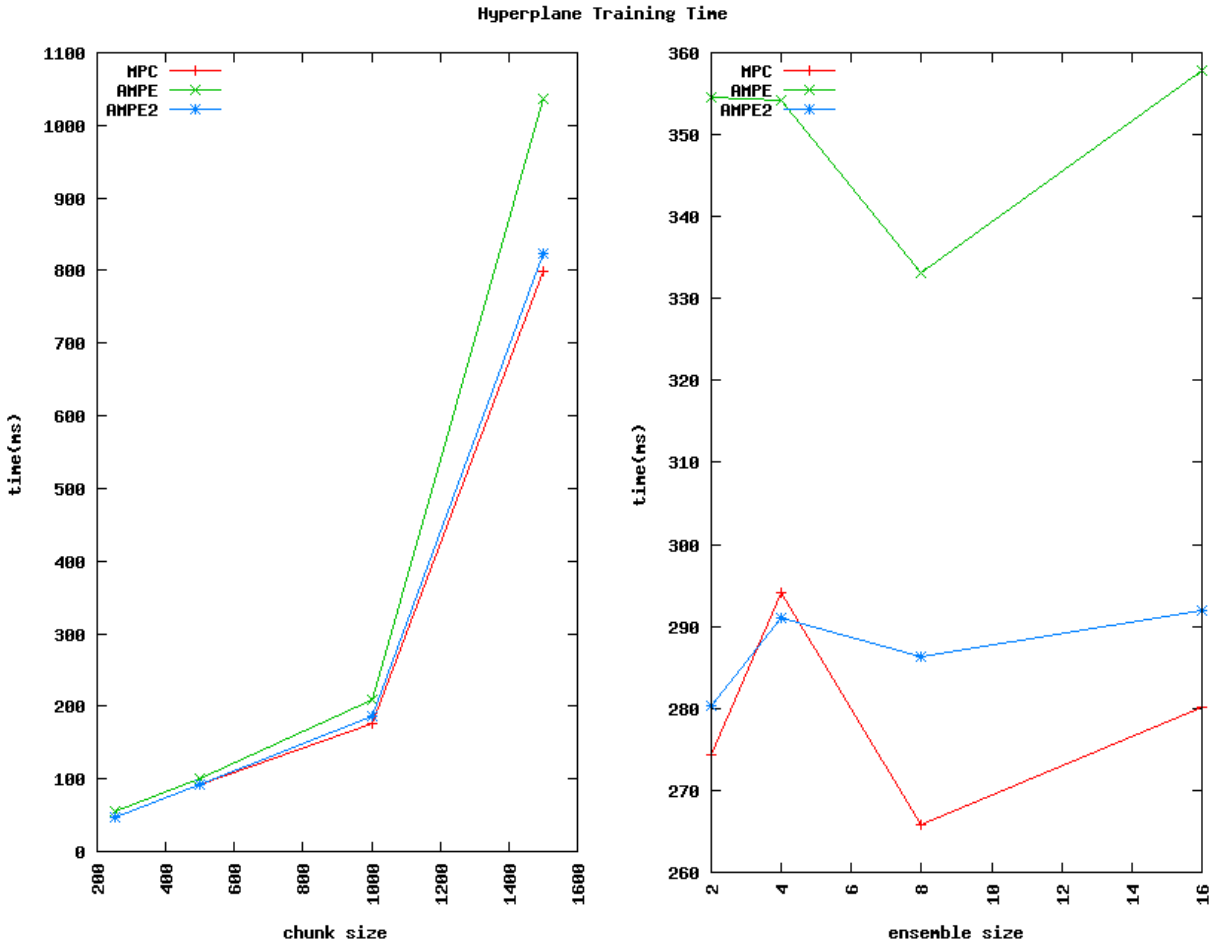

Figure 4.27: Training Times for MPC, AMPE, & AMPE2 methods on Hyperplane Data Set.

4.27 and 4.28 show the AMPE methods' average training and testing times to be as high, if not higher than the other methods. The results are very close, and Tables 4.13, 4.14, 4.15, and 4.16 show the MPC method to have the lowest average training and testing times, while the AMPE2 method appears to have the worst times measured for testing and training phases. We decide not to pursue any additional analysis at this point; mostly due to the disappointing accuracy results found above. We next examine the accuracy and efficiency of the methods while processing the

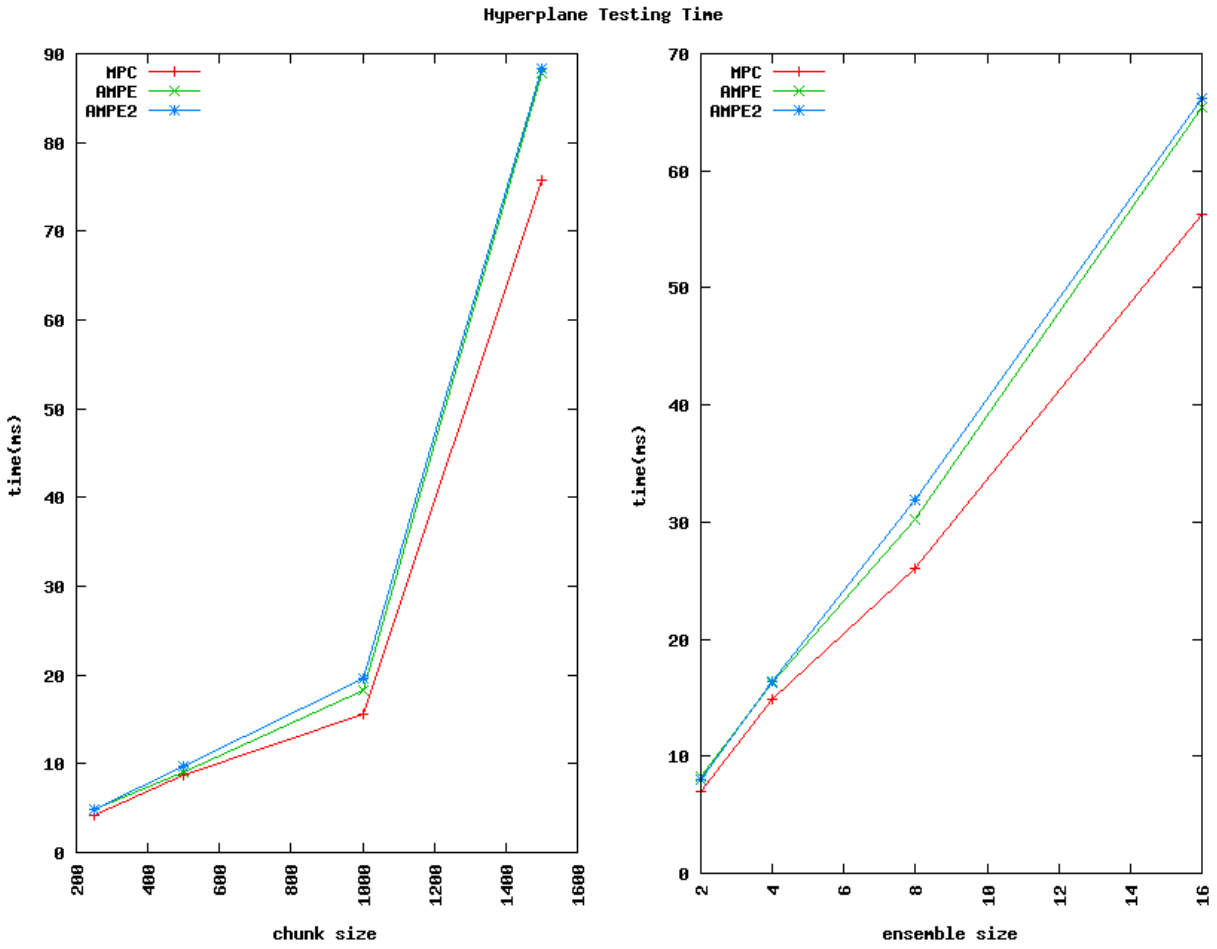

Figure 4.28: Testing Times for MPC, AMPE, & AMPE2 methods on Hyperplane Data Set.

SEA data set.

4.2.7 SEA Data Set Accuracy Results

Our SEA accuracy rates for the three methods(MPC, AMPE, & AMPE2) are plotted within Figure 4.29, and our initial "eyeball analysis" is that the MPC method appears to outperform both our AMPE and AMPE2 contributed techniques. The AMPE2 also appears to underperform both the MPC and AMPE algorithms. We test these assumptions with a couple of two way ANOVA tests that examine chunksize and ensemblesize influence on method accuracy. Our first ANOVA tests show a statistical difference between one of the methods' average accuracies($F = 6.13, p < 0.0051$), but it does not find an influence of chunksize selection on accuracy($F = 2.19, p > 0.121$). A test of factor independence also fails to show an interaction between chunksize and method($F = 1.48, p > 0.211$). Tukey HSD tests show a significant difference between the MPC and AMPE methods, a difference between the MPC and AMPE2 accuracies, but it does not report a significant difference between the AMPE and AMPE2 methods for this data set. Like our prior analysis of the existing methods and the SEA data set, we find the MPC method to outperform the rest. Unlike our prior analysis of the our contributed methods, we cannot distinguish our AMPE2 method to underperform the other methods.

Our ANOVA test for chunksize selection may have failed to show a correlation, but it is difficult to ignore the influence chunksize = 500 appears to have on the MPC and AMPE methods in Figure 4.29. Another reason we want to examine this a bit closer is our prior examination of the existing methods (AWE, MPC, and AUE) showed that the chunksize selection of 500 to also inversely influence classifier performance for the MPC and AUE methods while processing the SEA data set. This can be seen within Figure 4.13, and we see this same anomaly for the MPC and AMPE2 methods within Figure 4.29. Additionally, Table 4.9 shows the MPC method to have its lowest average accuracy rate at chunksize = 500, and our AMPE method to have its highest average accuracy rate at chunksize = 500.

Of interest within these results is to see that our AMPE method appears to show a bit of a performance gain when chunksize is equal to 500, whereas the other methods that appear influenced by chunksize exhibit performance decreases when chunksize is set to 500. Though hard to

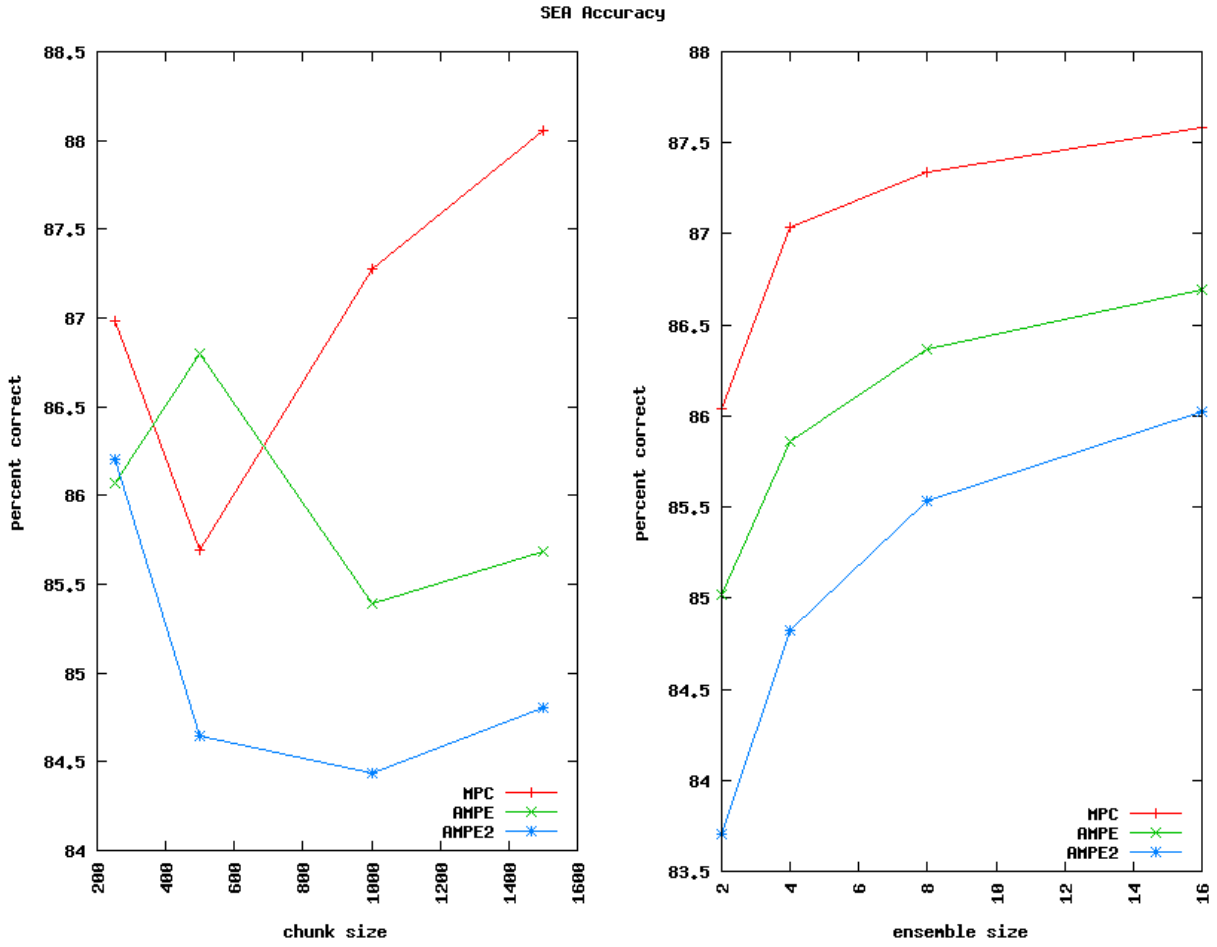

Figure 4.29: Accuracy for MPC, AMPE, & AMPE2 methods on SEA Data Set.

ignore, we still rely upon our ANOVA results that did not identify a statistical difference between chunksize selection on accuracy rates ($F = 2.19, p > 0.121$).

Our second two way ANOVA test also showed the MPC method to outperform our contributed methods($F = 8.50, p < 9.45E-04$), and it also failed to distinguish between the AMPE and AMPE2's accuracy rates. This second ANOVA test also showed ensemblesize selection had an influence on accuracy rates for the SEA data set ($F = 14.21, p < 2.86E-06$). A follow up Tukey test showed that the ensembles with 16 classifiers were on average more accurate than the classifiers with only 2. These findings are similar to earlier SEA data set results, as well as prior Hyperplane results. And we also see the now familiar monotonic growth pattern ensemblesize increases seem

to have on the methods' average accuracy rates. Next we focus on ensemble efficiency in order to gain an understanding of how the methods consume resources while processing the SEA data set.

4.2.8 SEA Data Set Efficiency Results

Memory consumption by the three methods can be seen plotted within Figure 4.30. We observe our AMPE method to appear to have the highest memory requirement of the three methods (MPC, AMPE, &AMPE2) for both the chunksize and ensemblesize trials. The AMPE2 method appears to require the least memory of the techniques. Table 4.11 and Table 4.12 also show this, but as can be seen within the measurement variances, additional analysis would be required to establish a clearly more efficient method. We decide not to follow this additional analysis given our prior accuracy results of the three techniques. For the fourth data set now we see that our contributed AMPE method requires additional computational resources, but without an associated increase in accuracy performance.

The training and testing times of the MPC, AMPE, and AMPE2 methods are graphed in Figures 4.31 and 4.32. These plots suggest our AMPE method requires the most in training and testing times for both the chunksize and ensemblesize experiments. The training and testing time results are very close, and Tables 4.13, 4.14, 4.15, and 4.16 show the AMPE2 method to have the lowest average times required for training and testing. They also show our AMPE method to have the highest average times for training and testing. Additional analysis would be required at this point to clearly establish which method was most efficient. We decide not to follow any additional analysis at this point due to the previous accuracy results for the algorithms.

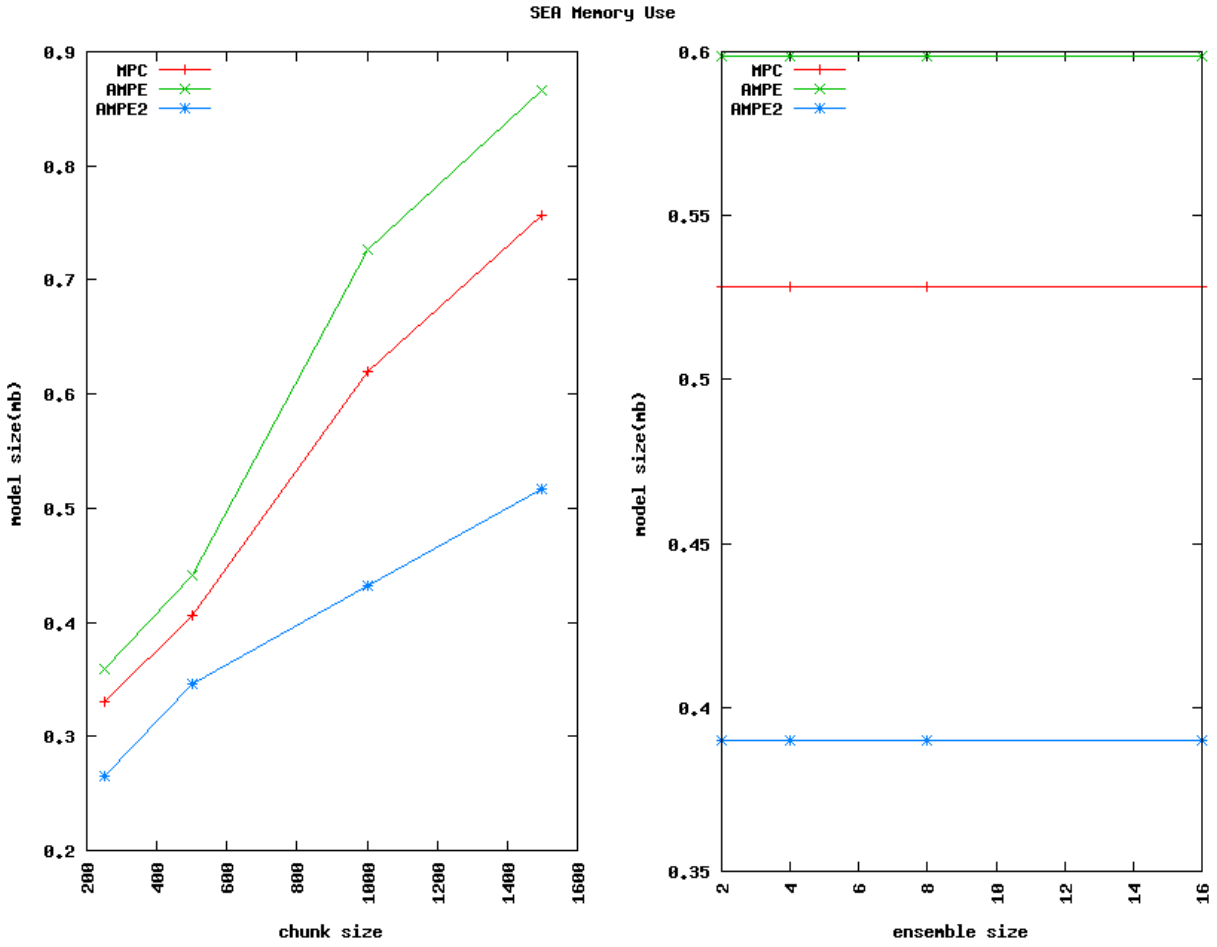

Figure 4.30: Ensemble Model Size for MPC, AMPE, & AMPE2 methods on SEA Data Set.

4.2.9 Summary of Contributed Ensemble Method Results

Our two contributed methods, the AMPE and AMPE2 algorithms, were designed to better understand ensemble methods and concept drift. Our AMPE method added a concept drift detection mechanism to dynamically adjust partitions of chunks when concept drift was determined to be prevalent. We hypothesized that these additions would yield an accuracy improvement over the MPC technique. Our experimental results and subsequent analysis do not support our hypothesis.

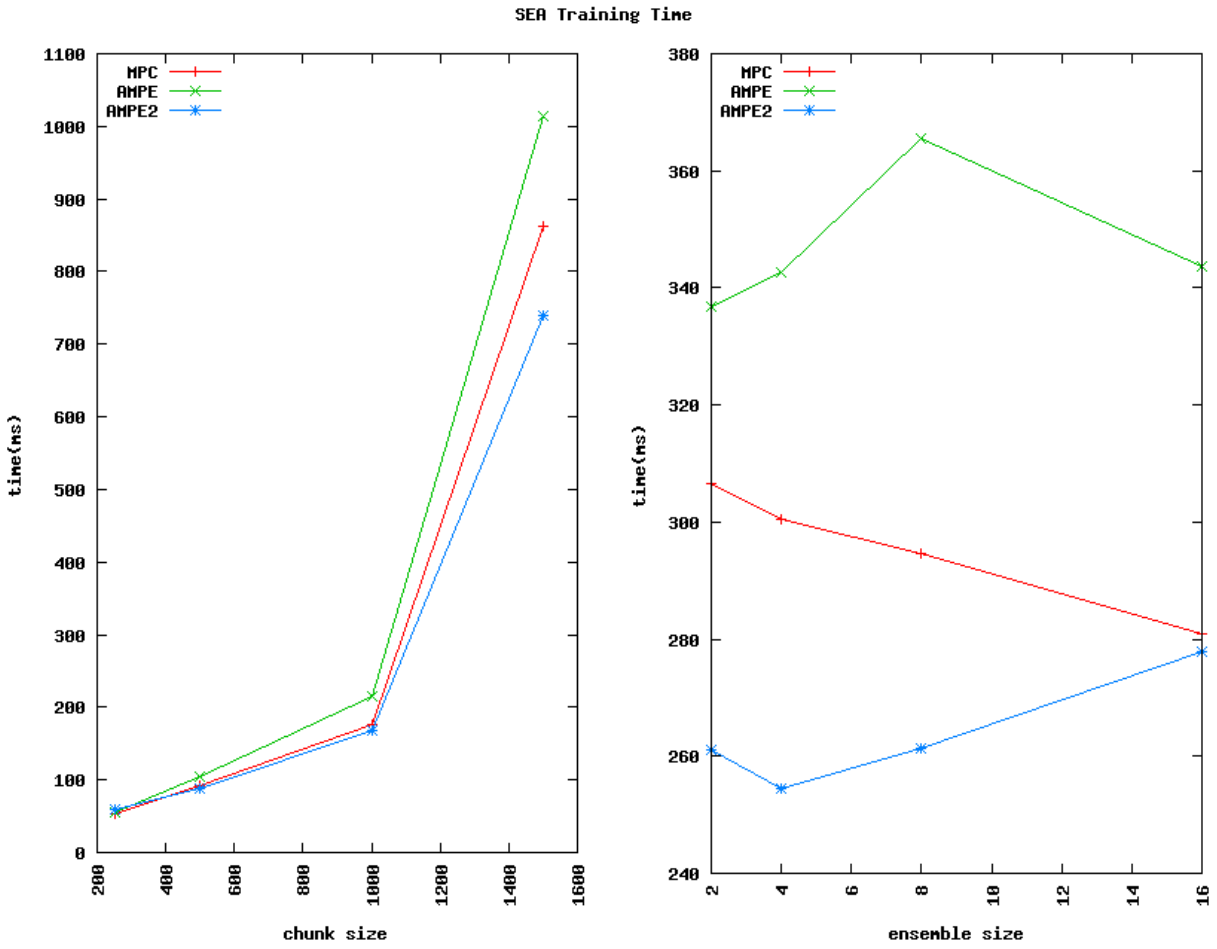

Figure 4.31: Training Times for MPC, AMPE, & AMPE2 methods on SEA Data Set.

From our results, using these data sets, we cannot establish a performance increase of our contributed AMPE method over the existing state of the art method: the MPC technique. We found our AMPE method to perform comparably to the MPC algorithm, but we also found our AMPE method to consume significantly more computational resources than the MPC ensemble. From this we learn that more sophisticated adaptation may be required to handle changes within the underlying class distribution.

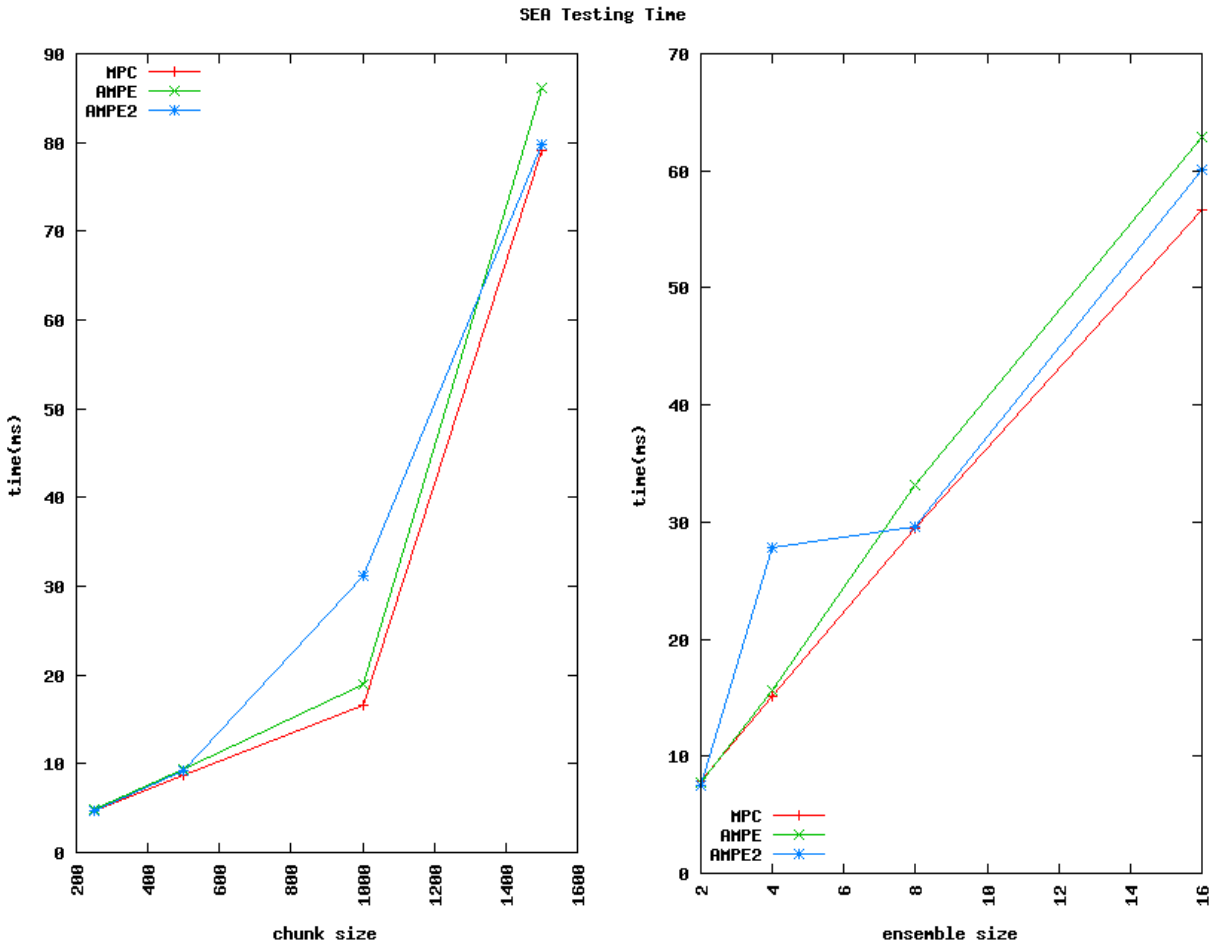

Figure 4.32: Testing Times for MPC, AMPE, & AMPE2 methods on SEA Data Set.

Similar to our results and analysis of the MPC, AWE, and AUE methods, we were able to again discern characteristics of the data sets and concept drift from our analysis of our contributed techniques. And again similar to our prior results, we were able to see how chunksize selection influenced some of the methods, and again on only some of the data sets.

Our ANOVA tests of independence that examined interactions between chunksize/ensemblesize and method also showed some interactions for some of the methods on some of the data set trials. Whereas the results of the three existing ensemble methods showed an interaction for both chunksize and ensemblesize on two data sets, the results for our contributed methods only showed an interaction between chunksize and method on the LED data set. And similarly to our conclusions regarding the existing ensemble methods, we cannot discount the influence both chunksize *and* method selection have on classifier performance. And again these observations begin to beckon questions of how we might utilize these differences in ensemble accuracies to improve overall system performance.

The second contributed method, our AMPE2 algorithm, was designed to learn about the influence prior examples have upon classification performance when concept drift is determined to be present in the data stream. Our first contributed method adjusted the size of partitions of chunks used to train and test with based upon concept drift detection and error thresholds. Our AMPE method increased partition size when concept drift is first detected, and its training and testing phases are then more rigorous where concept drift is within the data stream. Our AMPE2 method then inverses when to increase training and testing partitions, and does so when concept drift is determined to not be present. The AMPE2 algorithm ensemble is then comprised of classifiers that favor a quiescent system: when concept is not present, and the relative errors are low. We formed our second hypothesis around the importance of training and testing with examples where concept drift is found, and where it is not. We hypothesized that a method that trains and tests with data closer to where concept drift is found would be more accurate than a method that used data further away from concept drift. Specifically, our AMPE method would outperform our AMPE2 technique.

Our experimental results and statistical analysis support this hypothesis. We were able to show a significant statistical difference between method average accuracy rates for three of the four data sets in our trials. Our analysis supports our hypothesis that classifiers in an ensemble that train and test with data more relevant to where concept drift occurs would be more accurate than a method that used data further from where concept drift is detected. Our results and analysis not only support our hypothesis, but we are also able to learn new characteristics of classification when concept drift occurs in the data stream. An example can be found within our AMPE2 results. What was surprising within these results is just how well our AMPE2 method performed at times. Arguably our AMPE2 method performed significantly worse than all of the combined (AWE, MPC, AUE) and contributed methods for the smaller data sets(Electricity, LED), but surprisingly it performed better on the larger data sets (Hyperplane, SEA). The AMPE2 method performed on par with the AWE method, and even outperformed the AUE method for both the Hyperplane and SEA data sets. The AMPE2 also defied conventional wisdom by not only being as accurate as some of the state of the art methods for the larger data sets, but when it was as accurate, it was also as efficient, if not more than the other ensemble methods. This leads us to believe that classifier accuracy rates can be influenced by data found nearer concept drift, but also by examples found in the data stream where concept drift is not occurring. And similar to our prior findings, our results begin inciting additional research questions. Can we also combine the use of training and testing examples found further from concept drift with training and testing examples found nearer concept drift? Future research would examine additional drift detection techniques, as well as chunksize selection combinations, and how to combine training and testing examples from different data stream locations.

Table 4.9: Classification Accuracy vs. Chunksize for MPC, AMPE, & AMPE2 methods.

$Algorithm_{ChunkSize}$	Electricity	LED	Hyperplane	SEA
MPC_{250}	70.17 ±2.10	73.57 ±0.72	85.90 ±1.11	86.98 ±1.09
MPC_{500}	69.67 ±0.64	73.95 ±0.05	86.46 ±1.07	85.69 ±0.99
MPC_{1000}	68.29 ±0.83	73.96 ±0.007	85.15 ±0.53	87.28 ±0.49
MPC_{1500}	67.16 ±0.38	73.98 ±0.006	85.51 ±0.36	88.06 ±0.35
$MPC_{avg.}$	68.83 ±1.36	73.87 ±0.20	85.76 ±0.56	**87.00** ±0.99
$AMPE_{250}$	70.47 ±1.78	73.93 ±0.07	85.92 ±1.11	86.07 ±1.09
$AMPE_{500}$	70.04 ±1.63	73.95 ±0.06	86.50 ±1.05	86.80 ±1.00
$AMPE_{1000}$	68.87 ±0.99	73.96 ±0.007	85.16 ±0.51	85.39 ±0.48
$AMPE_{1500}$	67.51 ±1.05	73.98 ±0.005	85.54 ±0.36	85.68 ±0.35
$AMPE_{avg.}$	**69.22** ±1.33	**73.96** ±0.02	**85.78** ±0.57	85.99 ±0.61
$AMPE2_{250}$	42.92 ±1.33	72.78 ±0.80	85.70 ±1.52	86.20 ±1.52
$AMPE2_{500}$	39.83 ±0.08	72.43 ±0.23	83.96 ±0.79	84.64 ±0.79
$AMPE2_{1000}$	39.68 ±0.0003	72.30 ±1.94	83.05 ±1.08	84.44 ±0.96
$AMPE2_{1500}$	38.36 ± < 0.00001	70.41 ±0.42	83.37 ±0.86	84.80 ±0.77
$AMPE2_{avg.}$	*40.20* ±1.93	*71.98* ±1.01	*84.02* ±1.18	*85.02* ±0.80

Table 4.10: Classification Accuracy vs. Ensemblesize for MPC, AMPE, & AMPE2 methods.

$Algorithm_{EnsembleSize}$	Electricity	LED	Hyperplane	SEA
MPC_2	67.78 ±0.95	73.57 ±0.73	84.74 ±0.35	86.04 ±0.42
MPC_4	68.83 ±1.33	73.94 ±0.07	85.62 ±0.46	87.04 ±0.56
MPC_8	69.12 ±1.81	73.97 ±0.02	86.16 ±0.72	87.34 ±0.79
MPC_{16}	69.56 ±2.19	73.97 ± < 0.007	86.50 ±0.91	87.58 ±0.94
$MPC_{avg.}$	68.82 ±0.76	73.86 ±0.20	85.76 ±0.77	**87.00** ±0.68
$AMPE_2$	67.61 ±0.66	73.58 ±0.72	84.79 ±0.38	85.02 ±0.40
$AMPE_4$	68.79 ±0.81	73.94 ±0.07	85.63 ±0.46	85.86 ±0.54
$AMPE_8$	69.92 ±0.99	73.97 ±0.02	86.17 ±0.72	86.37 ±0.77
$AMPE_{16}$	70.57 ±1.39	73.97 ± < 0.0092	86.53 ±0.93	86.69 ±0.93
$AMPE_{avg.}$	**69.22** ±1.30	**73.87** ±0.19	**85.78** ±0.76	85.99 ±0.73
$AMPE2_2$	39.91 ±1.21	71.10 ±0.97	82.66 ±0.89	83.70 ±0.44
$AMPE2_4$	40.04 ±1.50	71.55 ±1.34	83.77 ±1.13	84.82 ±0.71
$AMPE2_8$	40.13 ±1.71	72.52 ±1.52	84.53 ±1.33	85.53 ±0.97
$AMPE2_{16}$	40.71 ±2.90	72.75 ±1.29	85.12 ±1.45	86.02 ±1.14
$AMPE2_{avg.}$	*40.20* ±0.35	*71.98* ±0.78	*84.02* ±1.06	*85.02* ±1.01

Table 4.11: Model Size vs. Chunksize for MPC, AMPE, & AMPE2 methods.

$Algorithm_{ChunkSize}$	Electricity	LED	Hyperplane	SEA
MPC_{250}	0.27	0.75	0.33	0.33
MPC_{500}	0.30	0.87	0.41	0.41
MPC_{1000}	0.24	1.12	0.62	0.62
MPC_{1500}	0.19	1.36	0.76	0.76
$MPC_{avg.}$	**0.25** ±0.05	1.03 ±0.27	0.53 ±0.20	0.53 ±0.20
$AMPE_{250}$	0.32	0.65	0.36	0.36
$AMPE_{500}$	0.42	2.17	0.44	0.44
$AMPE_{1000}$	0.50	0.65	0.73	0.73
$AMPE_{1500}$	0.59	5.19	0.87	0.87
$AMPE_{avg.}$	*0.46* ±0.12	*2.17* ±2.14	*0.60* ±0.24	*0.60* ±0.24
$AMPE2_{250}$	0.31	0.61	0.27	0.27
$AMPE2_{500}$	0.46	0.63	0.35	0.35
$AMPE2_{1000}$	0.23	0.63	0.43	0.43
$AMPE2_{1500}$	0.27	0.63	0.52	0.52
$AMPE2_{avg.}$	0.32 ±0.10	**0.63** ±0.01	**0.40** ±0.11	**0.40** ±0.11

Table 4.12: Model Size vs. Ensemblesize for MPC, AMPE, & AMPE2 methods.

$Algorithm_{EnsembleSize}$	Electricity	LED	Hyperplane	SEA
MPC_2	0.25	1.03	0.53	0.53
MPC_4	0.25	1.03	0.53	0.53
MPC_8	0.25	1.03	0.53	0.53
MPC_{16}	0.25	1.03	0.53	0.53
$MPC_{avg.}$	**0.25** ±0.00	1.03 ±0.00	0.53 ±0.00	0.53 ±0.00
$AMPE_2$	0.46	2.17	0.60	0.60
$AMPE_4$	0.46	2.17	0.60	0.60
$AMPE_8$	0.46	2.17	0.60	0.60
$AMPE_{16}$	0.46	2.17	0.60	0.60
$AMPE_{avg.}$	*0.46* ±0.00	*2.17* ±0.00	*0.6* ±0.00	*0.6* ±0.00
$AMPE2_2$	0.32	0.62	0.39	0.39
$AMPE2_4$	0.32	0.62	0.39	0.39
$AMPE2_8$	0.32	0.62	0.39	0.39
$AMPE2_{16}$	0.32	0.62	0.39	0.39
$AMPE2_{avg.}$	0.32 ±0.00	**0.62** ±0.00	**0.39** ±0.00	**0.39** ±0.00

Table 4.13: Training Time vs. Chunksize for MPC, AMPE, & AMPE2 methods.

$Algorithm_{ChunkSize}$	Electricity	LED	Hyperplane	SEA
MPC_{250}	898.86	95.20	47.87	52.88
MPC_{500}	2740.91	175.18	91.98	93.01
MPC_{1000}	8842.59	346.72	175.67	175.30
MPC_{1500}	16958.33	1702.38	799.00	861.37
$MPC_{avg.}$	7360.17 ±7243.58	*579.87* ±755.66	**278.63** ±350.94	295.64 ±380.58
$AMPE_{250}$	996.59	84.73	54.87	54.92
$AMPE_{500}$	4181.82	167.82	99.60	104.27
$AMPE_{1000}$	17296.30	333.79	208.26	215.99
$AMPE_{1500}$	41541.67	1509.26	1036.88	1013.33
$AMPE_{avg.}$	*16004.10* ±18428.5	**523.90** ±665.02	*349.90* ±462.49	*347.13* ±449.22
$AMPE2_{250}$	734.09	113.23	47.57	59.33
$AMPE2_{500}$	1781.82	190.94	92.07	87.65
$AMPE2_{1000}$	4833.33	467.36	186.26	167.80
$AMPE2_{1500}$	10520.83	1395.34	823.60	739.73
$AMPE2_{avg.}$	**4467.52** ±4394.23	541.72 ±589.02	287.38 ±362.13	**263.63** ±320.71

Table 4.14: Training Time vs. Ensemblesize for MPC, AMPE, & AMPE2 methods.

$Algorithm_{EnsembleSize}$	Electricity	LED	Hyperplane	SEA
MPC_2	7496.63	551.18	274.31	306.52
MPC_4	7201.96	560.13	294.20	300.39
MPC_8	7336.68	597.05	265.85	294.71
MPC_{16}	7405.43	611.11	280.16	280.94
$MPC_{avg.}$	7360.18 ±124.17	*579.87* ±28.77	**278.63** ±11.93	295.64 ±10.92
$AMPE_2$	16100.00	532.68	354.50	336.71
$AMPE_4$	15854.69	511.47	354.17	342.76
$AMPE_8$	16121.38	510.45	333.12	365.49
$AMPE_{16}$	15940.30	541.00	357.81	343.57
$AMPE_{avg.}$	*16004.10* ±128.25	**523.9** ±15.33	*349.90* ±11.31	*347.13* ±12.62
$AMPE2_2$	4472.29	518.48	280.28	261.00
$AMPE2_4$	4387.92	488.31	291.02	254.45
$AMPE2_8$	4503.56	586.68	286.25	261.23
$AMPE2_{16}$	4506.31	573.40	291.96	277.84
$AMPE2_{avg.}$	**4467.52** ±4394.23	541.72 ±589.02	287.38 ±362.13	**263.63** ±320.71

Table 4.15: Testing Time vs. Chunksize for MPC, AMPE, & AMPE2 methods.

$Algorithm_{ChunkSize}$	Electricity	LED	Hyperplane	SEA
MPC_{250}	76.59	10.0	4.23	4.65
MPC_{500}	247.73	17.50	8.63	8.72
MPC_{1000}	944.44	37.84	15.63	16.60
MPC_{1500}	1895.83	192.77	75.72	79.10
$MPC_{avg.}$	791.148 ±826.58	64.53 ±86.30	**26.05** ±33.44	**27.27** ±34.91
$AMPE_{250}$	85.23	10.35	4.81	4.85
$AMPE_{500}$	411.36	20.94	9.12	9.43
$AMPE_{1000}$	2657.41	40.72	18.30	19.00
$AMPE_{1500}$	7000.00	184.12	87.85	86.18
$AMPE_{avg.}$	*2538.5* ±3186.55	**64.03** ±81.04	30.02 ±38.96	29.87 ±38.00
$AMPE2_{250}$	37.50	11.51	4.90	4.70
$AMPE2_{500}$	11.36	23.40	9.67	9.26
$AMPE2_{1000}$	92.59	58.43	19.54	31.14
$AMPE2_{1500}$	250.00	169.73	88.39	79.82
$AMPE2_{avg.}$	**97.86** ±106.93	*65.77* ±72.11	*30.63* ±38.99	*31.23* ±34.39

Table 4.16: Testing Time vs. Ensemblesize for MPC, AMPE, & AMPE2 methods.

$Algorithm_{EnsembleSize}$	Electricity	LED	Hyperplane	SEA
MPC_2	336.32	13.71	7.01	7.87
MPC_4	505.01	35.00	14.85	15.11
MPC_8	896.55	69.15	26.05	29.48
MPC_{16}	1426.72	140.24	56.30	56.61
$MPC_{avg.}$	791.15 ±484.36	64.53 ±55.40	**26.05** ±21.63	**27.27** ±21.52
$AMPE_2$	934.22	16.40	8.25	7.81
$AMPE_4$	1696.95	33.40	16.20	15.65
$AMPE_8$	3151.18	66.65	30.21	33.18
$AMPE_{16}$	4371.65	139.68	65.42	62.83
$AMPE_{avg.}$	*2538.50* ±1529.46	**64.03** ±54.58	30.02 ±25.29	29.87 ±24.23
$AMPE2_2$	50.88	15.90	8.00	7.46
$AMPE2_4$	76.87	31.39	16.39	27.86
$AMPE2_8$	150.86	74.14	31.87	29.57
$AMPE2_{16}$	112.84	141.64	66.23	60.04
$AMPE2_{avg.}$	**97.86** ±43.52	*65.77* ±56.26	*30.62* ±25.72	*31.23* ±21.67

Chapter 5

Conclusions

Prior studies of the AWE, MPC, and AUE techniques were not very comprehensive, nor were the experimental procedures between the experiments consistent. In this research, we carefully compared the three existing methods to show how they perform differently under specific conditions. Our primary research contribution was a comprehensive comparison of the AWE, MPC, and AUE ensemble methods. Our second research contribution is the proposal of two new ensemble methods designed to further our understanding of concept drift and the algorithmic factors that influence performance when concept drift is determined to be present. In this chapter, we summarize our research findings and contributions, and describe possible future experiments that should address questions opened by our studies.

5.1 Contributions

Recall that our motivations for our study were to perform a comprehensive study of three ensemble methods (AWE, MPC, and AUE), and to examine concept drift characteristics through our contributed AMPE and AMPE2 methods. We used classifier accuracy, training/testing times, and memory consumption as metrics to compare the methods. Our experimental method leveraged portions of previous comparative experiments, and by doing so, we provided an equal playing field to compare the AWE, MPC, and AUE techniques.

Our first hypothesis was that one of the three ensemble methods would outperform the others.

Our experimental results do not statistically support this hypothesis as most of the algorithms performed similarly, the exception being the AUE method. We were not able to show a clear performance leader between the methods to support our hypothesis, but we did learn that the AUE method performed not as well as the rest on the very large data sets used within our studies.

Our second hypothesis was that by adjusting partition size according to the amount of error observed within the data stream, an improved classification accuracy would result, whilst maintaining reasonable efficiency. In other words, our AMPE method that adjusts partition size according to error will outperform a method that does not adjust partition size, or the MPC method. The results from our experiments do not support this hypothesis either. Our contributed AMPE method performed similarly to the MPC method, but we cannot show a statistically significant performance gain the AMPE method had over the MPC method. Our accuracy results reveal that our contributed AMPE method may have performed on par with the MPC technique, but our efficiency results suggest the additional resource requirements of the AMPE method cannot be justified by its classification accuracy results.

Our final hypothesis focused on better understanding the influence of selecting training examples closer to where concept drift occurs has on classifier performance. We wanted to learn the importance of training an ensemble method with data elements from where concept drift is occurring versus when the system is quiescent. To do this, we modified our AMPE method to place emphasis on training more when concept drift was not evident. Our empirical results support our hypothesis that training an ensemble method's classifiers with data that is closer to concept drift (AMPE) is more important for an accurate system than an ensemble technique that trains with data selected while the system is not experiencing concept drift (AMPE2). Our results suggest that our AMPE2 method significantly underperformed all of the methods we examined for data sets with significant concept drift. What was surprising within our results was the AMPE2 method performed comparable to, or better than several of the other techniques we examined for very large data sets. For example, our AMPE2 method's mean accuracy results were not only higher than AUE's for half of our data set trials, but our AMPE2 method also required less resources in terms

of memory use and times required for training and testing the classifiers.

After examining several state of the art ensemble methods, and attempting to improve upon them with our AMPE method, we see that there are multitudes of factors that can influence which algorithm performs best for a particular data set. We found it impossible to generalize which method or parameter selection was best for all the data sets we used within our experiments. Similar to both Wang et al.'s and Masud et al.'s results, we too see that chunksize and ensemblesize selection influence performance. But unlike those studies, we also observed interactions between both chunksize/ensemblesize selections and method. This tells us that not only is it difficult to generalize which method performs best, but it is even more difficult to narrow a set of parameters that are optimal for each method. Results like these begin to raise more questions than answers about characteristics of concept drift, and how we may accommodate it.

5.2 Future Work

Future research would pursue learning more about concept drift and how one may accommodate it. Example questions that were raised within this research range from how characteristics within concept drift can be leveraged to how changes within current approaches could be beneficial to mitigating concept drift.

Our results showed that some chunksizes and ensemblesizes were shown to produce better classification accuracy results, and the optimal chunk and ensemblesizes differed across data set experiments. We also observed that some methods and selections of chunksize and ensemblesize have an interdependence and influence classifier performance. Could an ensemble method leverage this knowledge to adjust dynamically? A possible future study would create a new ensemble method that tries to leverage chunksize, ensemblesize, and possibly the interaction of them with method selection to increase classification performance.

We also saw within our results that for very large data sets the AUE technique did not perform as well as the other methods. Recall that the AUE method maintains classifiers within the ensemble by updating them versus replacing like the other techniques we examined. Do the AUE classifiers become diluted by updating versus replacement in very large data streams? Can an ensemble method recognize when adding any additional context would only dilute the class distribution? And how can this be combined with our new found knowledge that training the classifiers with data nearer to context drift is more important than training the classifiers with data collected when the system is stable. A possible future experiment would examine a new ensemble technique that attempts to recognize when adding any additional context only makes performance worse.

Our AMPE2 trials yielded surprising results that begged many additional queries. We determined that context closer to where concept drift occurs is more important for an accurate ensemble method, but our AMPE2 method's results suggest that there is also some importance within context not as near to concept drift. Can we combine the two for an even better classifier ensemble? Future work would then include examining additional drift detection methods, experiments with varying the number of chunks within a partition and different combinations of the two. We would also like to study additional techniques for determining drift thresholds, and additional data partitioning strategies.

Bibliography

[1] Manuel Baena-Garca, Jos Del Campo-vila, Raul Fidalgo, Albert Bifet, Ricard Gavald, and Rafael Morales-Bueno. Early drift detection method. *Fourth International Workshop on Knowledge Discovery from Data Streams*, 6:77–86, 2006.

[2] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *Journal of Machine Learning Research*, 11:1601–1604, August 2010.

[3] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby, and Ricard Gavaldà. New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 139–148, New York, NY, USA, 2009. ACM.

[4] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[5] Dariusz Brzezinski and Jerzy Stefanowski. Accuracy updated ensemble for data streams with concept drift. In *Hybrid Artificial Intelligent Systems: 6th International Conference*, pages 155–163, 2011.

[6] Dariusz Brzeziski. Mining data streams with concept drift. Master's thesis, Poznan University of Technology, 2010.

[7] Wei Fan. Systematic data selection to mine concept-drifting data streams. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 128–137, New York, NY, USA, 2004. ACM.

[8] João Gama, Ricardo Rocha, and Pedro Medas. Accurate decision trees for mining high-speed data streams. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 523–528, New York, NY, USA, 2003. ACM.

[9] Michael Harries. SPLICE-2 Comparative Evaluation: Electricity Pricing. Technical report, The University of South Wales, 1999.

[10] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 97–106, New York, NY, USA, 2001. ACM.

[11] Ralf Klinkenberg and Stefan Rping. Concept drift and the importance of examples. In *Text Mining Theoretical Aspects and Applications*, pages 55–77. Physica-Verlag, 2002.

[12] Mohammad Masud, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani Thuraisingham. A multi-partition multi-chunk ensemble technique to classify concept-drifting data streams. In Thanaruk Theeramunkong, Boonserm Kijsirikul, Nick Cercone, and Tu-Bao Ho, editors, *Advances in Knowledge Discovery and Data Mining*, volume 5476 of *Lecture Notes in Computer Science*, pages 363–375. Springer Berlin / Heidelberg, 2009.

[13] Jeffrey C. Schlimmer and Richard H. Granger, Jr. Incremental learning from noisy data. *Machine Learning*, 1:317–354, March 1986.

[14] Martin Scholz and Ralf Klinkenberg. An ensemble classifier for drifting concepts. In *In Proceedings of the Second International Workshop on Knowledge Discovery in Data Streams*, pages 53–64, 2005.

[15] W. Nick Street and YongSeog Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 377–382, New York, NY, USA, 2001. ACM.

[16] Alexey Tsymbal. The Problem of Concept Drift: Definitions and Related Work. Technical report, Departament of Computer Science Trinity College, Dublin, 2004.

[17] Kagan Tumer and Joydeep Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection Science*, 8(3-4):385–404, 1996.

[18] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 226–235, New York, NY, USA, 2003. ACM.

[19] Gerhard Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. In *Machine Learning*, pages 69–101, 1996.

[20] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

[21] Kun Zhang and Wei Fan. Forecasting skewed biased stochastic ozone days: analyses, solutions and beyond. *Knowledge and Information Systems*, 14:299–326, 2008. 10.1007/s10115-007-0095-1.