TNI Journal of Engineering and Technology
Vol.9 No.1 January - June 2021

A Comparison of Real-Time Data Analytics Algorithms

Ekarat Rattagan[1*]

[1*]*Graduate School of Applied Statistics, National Institute of Development Administration, Bangkok, Thailand*

*Corresponding Author. E-mail address: ekarat@as.nida.ac.th

Received: 10 May 2021; Revised: 17 May 2021; Accepted: 24 May2021
Published online: 25 June 2021

Abstract

Today's world is overwhelmed with the stream of data generated by IoT sensors, Smartphone applications, E-commerce transactions, etc. Data streaming and real-time analytics tools are necessary to apply for various purposes such as financial fraud detection, recommended products, or disaster warning systems. Existing real-time data analytics tools such as StreamDM, Scikit-multiflow, or Massive Online Analysis (MOA) play a significant role in this field. There is, however, still a lack of well-comparisons among streaming algorithms in these tools. In this paper, we aim to study and compare the performance of the streaming algorithms provided by Scikit-multiflow, one of the most popular tools. In the experiment, we compare various algorithms on classification and regression problems in terms of accuracy, model size, memory, etc. The synthesized and real-world datasets are both employed for the experiment. The experimental results illustrate that the Hoeffding-Tree algorithm shows the best performance among other algorithms.

Keywords: Streaming, Algorithms, Real-Time, Analytics, Concept drift

I. INTRODUCTION

With the integration of high- speed network technology, e. g. , Wi- Fi and cellular network, and mobile technology, e.g., mobile devices and Internet-of-Things (IoT), the physical objects have connected with other things seamlessly. With these connections, people can do several things online via applications (apps), especially mobile apps. For instance, they use mobile apps to spend their living costs, pay the monthly home mortgage, have an online meeting, etc. With these connections, machines can communicate with other machines. For example, IoT devices embedded in the factorial machinery can send/receive data to/from other IoT devices via a low-power network. Food- delivery drivers can keep connecting with their customers via GPS technology. These scenarios of a modern society generate a massive of real-time data streaming in every unit of time.

This data streaming pushes data scientists, data engineers, or business analysts have to discover new knowledge from various sources of data- driven economics. Traditionally, their daily- routine tasks involved the pipeline process, including collecting, cleaning, training mathematical models (models for short) , and analyzing data, iteratively. The data-analytic tools for this pipeline process are traditionally invented to work with a complete set of data or bound data, namely the batch processing approach, which contrasts with the other direction, i. e. , the stream processing approach. Figure 1 depicts the overall diagram of batch and stream processing approaches.

The stream processing approach is appropriate for processing the streaming data, or unbounded data, which is naturally fast, imbalanced, and infinite; this means the analysts do not know when it will finish streaming [1]. Meanwhile, batch processing tools are insufficient to handle the streaming data because they lack streaming algorithms. Therefore, the models generated from the batch processing may show low performance

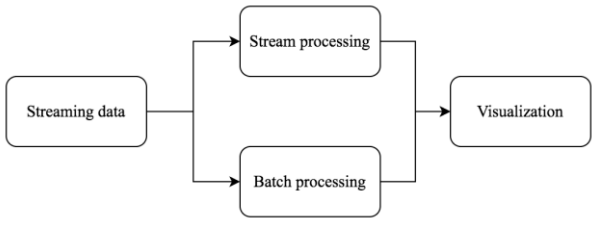

Figure 1 Batch VS Streaming processing

in predicting the sudden change of data or concept drifts, e.g., the fraud behaviors in financial transactions. This is because the models are trained by the outdated data. To squeeze the highest capability of models, we should frequently update the generated models with a small set of data to get ready for use in real-time.

Currently, there are a couple of software tools invented for stream processing analytics, for instance, Massive Online Analysis (MOA) [2], Scikit-learn Multiflow [3], and StreamDM of Huawei [4]. In particular, these tools provide online machine learning algorithms for generating models for analyzing streaming data. The online algorithms, for instance, include Hoeffding Tree Classifier, Stochastic Gradient Descent Classifier, Adaptive Windowing drift detection, etc. Figure 2 illustrates the process of an online machine learning algorithm in the Scikit-multiflow framework.

As these existing tools are getting attention in stream processing communities, it is necessary to survey the strongest and weakest points of them. This paper aim to compare the online algorithms provided by the Scikit-multiflow framework, which shows the most attention in the community. The target algorithms have experimented in various

datasets, both synthesized and real-world streaming data. Moreover, different error metrics, including accuracy, kappa, f1, model size, training, and testing time, are used to justify the performance of the target algorithms.

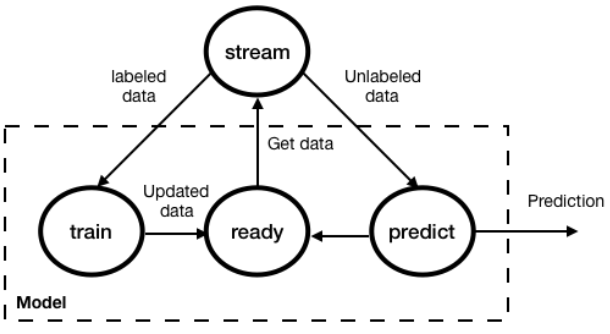

Figure 2 Online machine learning in Scikit-multiflow [3]

The remainder of this paper is organized as follows: Section II provides the background to this works; Section III gives the problem statement; Section IV and V presents the experimental results and concludes this work, respectively.

II. BACKGROUND

A. Stream processing tools

MOA is a software environment for developing and evaluating algorithms for online learning from data streams. It was designed to deal with the scaling up algorithms in the big data era. This tool provides algorithms for solving classification and clustering algorithms. For example, Hoeffding Trees, boosting, bagging for classification, StreamKM++, CluStream, and Clustree for clustering. MOA also provides several stream generators and evaluators for an experiment on the new algorithm design. It also cooperates with Waikato Environment for Knowledge Analysis (WEKA) tools for data analytics.

Scikit-multiflow is similar to MOA. It, however, provides multiple online algorithms for different stream learning issues, including single-output, multi-output, and multi-label issues. It is worth noting that multi-label is about predicting various output variables (features) instead of a single target variable, the unique feature of this tool. One of the main reasons for making the Scikit-multiflow framework popular is its python friendliness, while MOA is only for JAVA.

StreamDM is also an open-source stream processing tool created at Huawei-Noah's Ark lab. This tool is built on top of the Apache spark streaming technology, which is the extension of the core Spark API and leverages the benefit of Hadoop opensource. StreamDM is implemented in Scala programming language, and it is claimed as the first tool that implements advanced methods on top of the Apache spark streaming.

B. Concept Drift

In dynamic and non-stationary environments, the concept drift [5] is the streaming data whose relation between the input data and the target variable (data means) changes over time. The concept drift harms learning models generated from the batch processing because the models generated from the batch processing causes very high errors, which is impractical in real usage. Figure 3 illustrates the behaviors of the various concept drift as follow:

1) Abrupt concept drift means the transition between the new and old concept (or data mean) is minimal, as shown in Figure 3a. The stock market shows this behavior when an unexpected event is coming.

2) Gradual concept drift means the intermediate concept continuously appears during the transition between new and old ones, as shown in Figure 3b. This behavior is quite natural to happen and addressed by time series models.

3) Incremental concept drift means the old and new concepts alternately concur during the transition period. Figure 3c depicts this behavior.

4) Recurring concept drift means the old concept is seen again, as shown in Figure 3d. The specific time in a year, such as New year, causes this behavior.

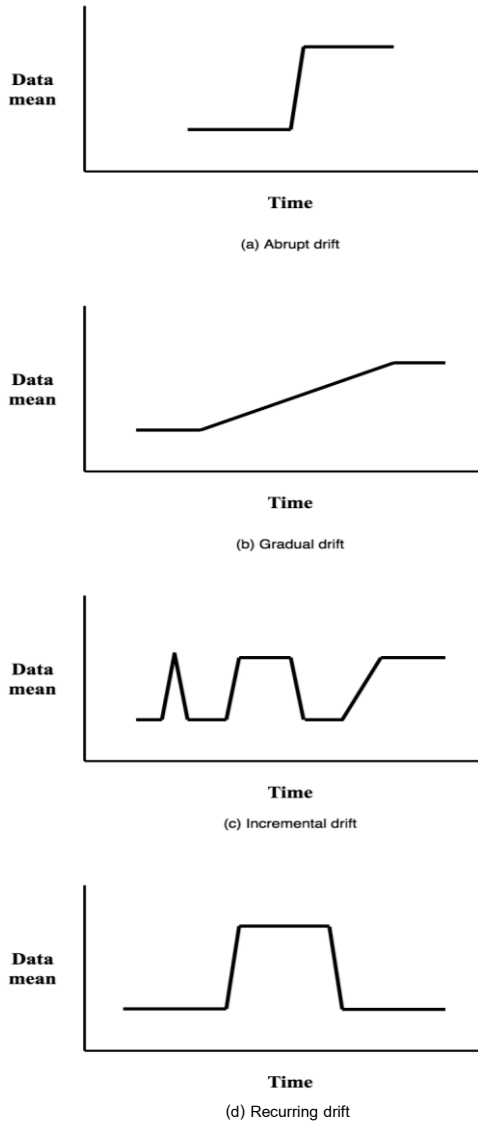

Figure 3 Concept drift behaviors

C. Evaluation method for Streaming processing

1) Holdout validation is the most popular and straightforward evaluation method. This method requires separate datasets into two sets: training and testing set, e.g., 70% and 30%, respectively. The holdout validation method usually uses less computational time; it, however, generates the overfit problems.

2) Prequential evaluation [6]. For each streaming instance, this method requires first to predict the data, and if the label is known, it uses the predicted value and label for evaluation and then trains the model, as shown in Figure 2. For this reason, the accuracy of the existing model can be incrementally updated. The benefit of this method is that it does not require separating the data into two sets like the holdout method. Thus, this method can save more space than other methods. The limitation of this method is that it cannot depict changes in class distribution, such as concept drift. Eq (1) shows the prequential error at time i is based on the accumulated sum of a loss function l between the prediction $y_k^{'}$ and actual value y_k.

$$p_e(i) = \frac{1}{i}\sum_{k=1}^{i} l(y_k^{'}, y_k) \quad (1)$$

3) Kappa [7] is the evaluation metric used for assessing the agreement between two raters who each classify N items into C mutually exclusive categorical items. This method is appropriated for evaluating the performance of the classification problems, especially with imbalanced data. The definition of Kappa is defined as

$$kappa = \frac{p_0 - p_c}{1 - p_c} \quad (2)$$

where $p_0 = \frac{Sum\ of\ all\ agreements}{N}$ is the relative observed agreement among raters, and $p_c = \frac{1}{N^2}\sum_k n_{k1} n_{k2}$ is the hypothetical probability of chance agreement, using the observed data to calculate the probabilities of each observer

randomly seeing each category, where k is categories (e.g., yes or no), N is a number of observations, and n_{ki} is the number of times rater i predicted category k.

The kappa's value is varied from a negative value to 1. If the value is 1, it means all raters are in complete agreement. If the value is 0, that means there is no agreement among all raters. If the value is negative, it means the agreement is worse than the random event.

To illustrate an example, let's assume the situation that two teachers, ***T1*** and ***T2***, have to classify 50 students into two groups for some purposes. And, the result shows in the table below:

Table 1 Kappa example

		T2	
		yes	no
T1	yes	25	10
	no	10	5

Base on calculation, we found

$$p_0 = \frac{25+5}{50} = 0.6, \text{ and}$$

$$p_c = \left(\frac{n_{yes,T1}}{N} x \frac{n_{yes,T2}}{N}\right) + \left(\frac{n_{no,T1}}{N} x \frac{n_{no,T2}}{N}\right)$$

$$= \left(\frac{35}{50} \text{ x} \frac{35}{50}\right) + \left(\frac{15}{50} \text{ x} \frac{15}{50}\right)$$

$$= 0.49 + 0.09 = 0.58$$

Based on eq (1) calculation, $kappa = \frac{0.6-0.58}{1-0.58} = \frac{0.02}{0.42} = 0.05$. This value shows high disagreement between ***T1*** and ***T2*** for their classified decision.

III. PROBLEM STATEMENT

In this section, the problem statement is stated as follow,

(1) given, a set of algorithm A, including both online and batch algorithm, i.e., regressor, including K-Nearest Neighbor (K-NN), Adaptive Random Forest Regressor (ARF_RE), Hoeffding Tree Regressor (HT_RE), and classifier, including Hoeffding Adaptive Tree (HAT) classifier, and Sup9port Vector Machine (SVM) classifier.

(2) A target dataset D, which is comprised of synthesized and real-world data, both datasets are with/without concept drift behaviors.

(3) A collection of evaluation metrics M including Mean Squared Error, Kappa, F1-score, Model size (kB), and Training and Testing time (sec).

The objective of this study is to compare the performance of algorithms in set A, tested on the dataset D, and evaluated by the error metrics M.

IV. EXPERIMENTAL RESULT

In this section, we show the results on different experiments. We first compared the algorithm on the regression problem and then classification problem. We finally compared the algorithm on the concept drifts dataset, both synthesized and real-world data.

A. Regression problem Comparison

In this subsection, we compare three regression algorithms provided by Scikit-multiflow, including KNN (KNN), Adaptive Random Forest (ARF_RE), and Hoeffding Tree regressor (HT_REG). These three algorithms were first trained with 200 data samples and then evaluated on 1000 data samples generated by the built-in generators; the data contains 100 features and one target variable. We used three error metrics for the performance evaluation, including model size, running time, and Mean Squared Error (MSE). Figure 4 depicts that HT_REG shows the best performance since it has the lowest mean of MSE (29726.70), while ARF_RE shows the highest mean of MSE (38489.42). However, KNN shows the smallest

model size and running time over HT_REG and ARF_R, respectively.

B. Classification problem Comparison

In this subsection, we compared two classification algorithms, including Hoeffding Adaptive Tree (HAT), and Support Vector Machine (SVM), provided by Scikit- multiflow. We used Stream Ensemble Algorithm (SEA) generator to generate 20,000 samples. Each sample includes three features, one target, and two classes. These two algorithms were first trained with 200 samples and running time as the evaluation metrics for this classification comparison. Figure 5 illustrates that HAT shows the better F1 and kappa scores over the SVM algorithm. However, SVM is better than HAT in terms of model size and total running time.

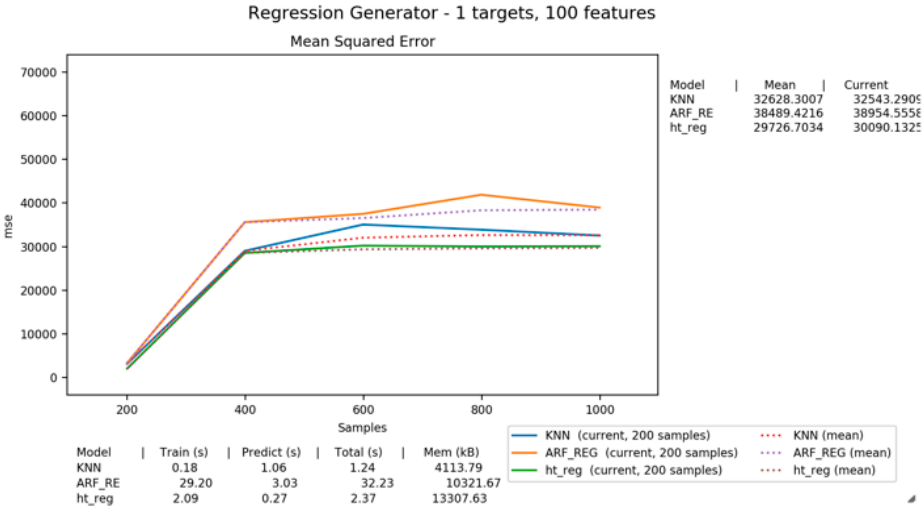

Figure 4 Comparing regression algorithms, including KNN, Adaptive Random Forest, and Hoeffding Tree algorithms

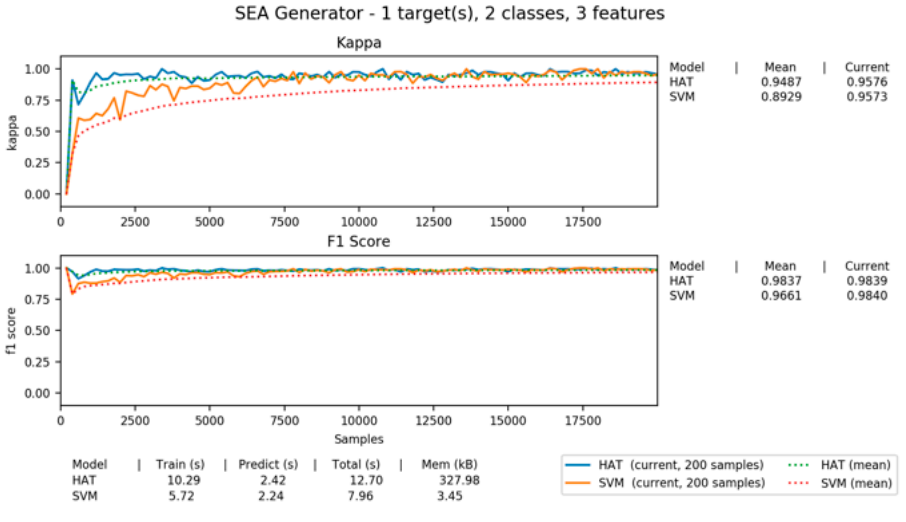

Figure 5 A comparison between Hoeffding Adaptive Tree (HAT) and Support Vector Machine (SVM) Algorithms in a classification problem

C. Concept Drift Comparison

In this subsection, we compared the two algorithms from the previous subsection, i.e., HAT and SVM, on the classification problem with two different datasets, including synthesized and real-world data integrated with the concept drift behaviors.

The synthesized concept-drift data was generated by a drifting Agrawal generator [8], composed of nine features, one target with two classes. The dataset is about to determine whether the loan should be approved or not. Figure 6 depicts that HAT shows 200% improvement over SVM on accuracy and F1 scores, while nearly 1000% improvement over SVM on the kappa metric.

In particular, SVM's kappa shows a negative value which represents disagreement between actual and predicted values. However, the model's size of HAT is much larger than SVM, which is not the significant point of the streaming analytics.

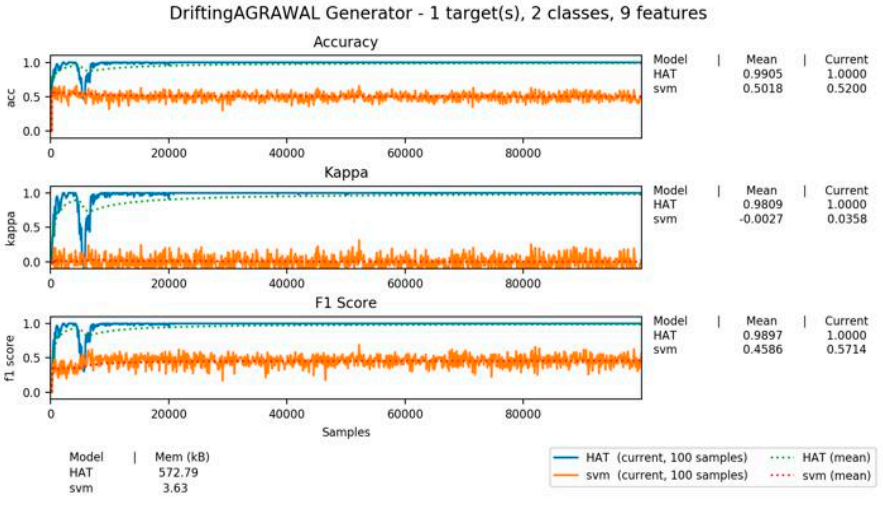

Figure 6 Real-world data of Concept drift comparison between HAT and SVM

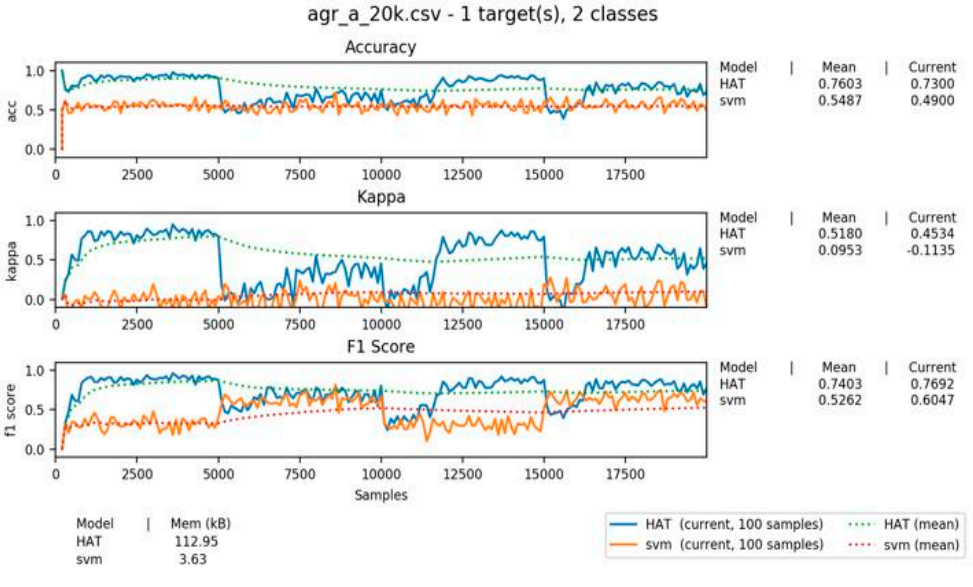

Figure 7 Synthesized data of Concept drift comparison between HAT and SVM

For the real-world streaming data, we used the agr_a_20k.csv file. Figure 7 depicts that HAT still shows about 40% improvement over SVM accuracy and F1 score, respectively. For the kappa metric, HAT shows a gain of nearly 500% over SVM.

However, in this experiment, SVM's kappa shows a positive value representing less agreement between actual and predicted values. However, the model's size of HAT is still larger than SVM. The results show that the different types of data, i.e., synthesized and real-world data, do not affect the testing algorithms.

From all experiments, the results illustrate that streaming algorithm such as Hoeffding Adaptive Tree (HAT) has the most performance compared to Support Vector Machine in the classification problem. At the same time, Hoeffding Tree Regressor delivers the most performance over K-Nearest Neighbor and Adaptive Random Forest algorithm in the regression problem. This is because the HAT algorithm is designed to handle the streaming data.

V. CONCLUSION

This study compares and evaluates streaming and traditional machine learning algorithms, including regression algorithms, such as Hoeffding Tree Regressor, KNN regressor, and classification algorithms, such as Hoeffding Adaptive Tree (Hat) algorithm, SVM, on streaming data with and without concept drift behaviors. We evaluate the performances of algorithms by several streaming error metrics, including Kappa, F1-score, etc. The results reveal that the Hoeffding Tree family outperformed the other algorithms because of the nature of the streaming algorithm, which incrementally updates its model's parameters based on data change. Future works will be conducted more on other streaming algorithms with several datasets.

REFERENCES

[1] A. Kejariwal, S. Kulkarni, and K. Ramasamy, "Real time analytics: Algorithms and systems," *VLDB Endowment*, vol. 8, no. 12, pp. 2040–2041, Aug. 2015.

[2] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive online analysis," *Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.

[3] J. Montiel, J. Read, A. Bifet, and T. Abdessalem, "Scikit-multiflow: A multi-output streaming framework," *Journal of Machine Learning Research*, vol. 19, pp. 2915–2914, 2018.

[4] A. Bifet, S. Maniu, J. Qian, G. Tian, C. He, and W. Fan, "Stream DM: Advanced data mining in spark streaming," in *Proc. IEEE Int. Conf. Data Mining Workshop (ICDMW)*, Atlantic City, NJ, USA, Nov. 14–17, 2015, pp. 1608–1611.

[5] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, pp. 1–37, Apr. 2014.

[6] A. P. Dawid, "Present position and potential developments: Some personal views statistical theory the prequential approach," *Journal of the Royal Statistical Society, Series A*, vol. 147, no. 2, pp. 278–290, 1984.

[7] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, Apr. 1960.

[8] R. Agrawal, T. Imielinski, and A. Swami, "Database Mining: A performance perspective," *IEEE Trans. Knowledge and Data Engineering,* vol. 5, no. 6, pp. 914–925, Dec. 1993.