DGA1032 - Examen Écrit
Partie Dissertation

Fábio Dittrich

Advisor: Robert Sabourin
Co-advisor: Éric Granger

August 8, 2013

Contents

Nomenclature

Δn Number of new samples since the last k_{LOO} update

Δn_{max} Maximum number of new samples

ϵ_i Element-wise error compute with K_0

Γ A parameter used to compute the intervals of confidence

$\hat{\mu}_S^k$ The mean of the k^{th} feature of sequence S

$\hat{\sigma}^k$ The standard deviation of the k^{th} feature

$\hat{\sigma}_S^k$ The standard deviation of the k^{th} feature of sequence S

$\hat{T}_i$ The instant at which the change was detected by CDT

Λ The set of classes $\{\omega_1, \omega_2\}$

λ The parameter used to divide the search space for $Tref$

μ'_t The i_{th} moment of the sample distribution (i=$\{$1,2,3,4,5,6$\}$)

$\Upsilon(C_i)$ The split operator. It splits C_i in C_i and C_{i-1}

$\varepsilon(C_i, C_j)$ The equivalence operator. It returns 1 if C_i and C_j are equivalent and 0 otherwise

ζ_i The i_{th} cumulant of the sample distribution (i=$\{$1,2,3,4,5,6$\}$)

C The set of concepts

C_i The i^{th} concept

CDT_ϵ Detects changes in the classification error

CDT_X Detects changes in the data distribution

d Dimensionality of the instances

$D(C_i)$ The concept drift operator. It detects changes in D_i

D_i Set of features to check for changes in the i^{th} concept

F_i Set of features that characterize the i^{th} concept

h_0 The exponent used for the power-law transform

I_S^k The interval of confidence of the k^{th} feature of sequence S

$K(Z, x_t)$ The classifier operator. It is trained with set of supervised samples Z and classifies x_t sample as $\widehat{y_t} \in \Lambda$

K_0 is a classifier used to compute the errors for P_i

k_i The i_{th} cumulant of the Gaussian approximated distribution (i=$\{1,2,3\}$)

k_{LOO} The value of k estimated using LOO method

KNN k-nearest neighbor

LOO Leave-one-out

$M(s)$ Mean of sequence s

n Number of samples in the knowledge base of the KNN

P_i Sequence with estimates of classification errors of subsequences of Z_i

S_0 The number of sequences extracted from the initial training set

T^* The real instant of change

T_0 Number of samples of the initial training set

$Tend_i$ The time instant at which the concept C_i ended

$Tref_i$ The refined instant

U Continuous update classifier

$U(C_i, R)$ The update operator. It updates the C_i concept with the set of recent inputs R

$V(s)$ Gaussian-approximated variance of sequence s

vE Number of supervised observations of each sequence used to compute classification errors for P_i

vX Number of observations of each sequence used to compute mean and variance features for F_i

W Short memory classifier with window size 40

Z_i Set of supervised samples of the i^{th} concept

Chapter 1

Introduction

The Just-In-Time (JIT) is a framework developed by Alippi et al. [1, 2, 3, 4, 5, 6, 7] to take advantage of supervised samples that arrive over time to improve the classification rate. Moreover, it also enables the detection of changes in the concepts that rule the data, making the classification of dynamic problems possible.

The original framework was presented in [1, 2]. The subsequent papers deal with different problems, such as: creating an ensemble of JIT classifiers [4], managing the slow drift case [5, 7] and detecting recurrent concepts [3].

This dissertation is divided in three parts, one for each question. Therefore, the next session presents the processes needed to use JIT in a static environment as well as the experimental protocol and the results in this scenario. The following chapter introduces the JIT for non-stationary environments, with results for the same experiments presented before, but using JIT with a change detection mechanism. Finally, the last part discusses some general concepts about changing environments.

Chapter 2

A

In a static environment the JIT framework works to improve the classification accuracy over time by exploiting available supervised labels.

The JIT is composed of a set of concepts and five operators. The set of concepts is denoted by C=$\{C_1, \ldots, C_i, \ldots, C_N\}$, where C_i indicates the i^{th} concept and contains three subsets. For the scenario considered in this question, only one subset of the concept and two operators are necessary:

- Z_i is the set of supervised samples.
- The update operator $U(C_i, R) \rightarrow C_i$ receives new samples (R) and updates the Z_i set with supervised couples.
- The classifier $K(Z, x_t) \rightarrow \Lambda$ is configured on the set of supervised samples Z and assigns label $\widehat{y_t} \in \Lambda$ to instance x_t. Λ is the set of classes ($\Lambda = \{\omega_1, \omega_2\}$).

In Algorithm 1 and in Figure 2.1 a general overview of the JIT framework for stationary scenarios and a block diagram are presented. Initially, the concept C_0 is trained with the training data, i.e. supervised samples are stored in Z_0. Then, during the operation phase of the system, new supervised samples (represented by $\{x_t, y_t\}$) are used to update the value of Z_i, which is used to retrain the classifier. This classifier classifies all unsupervised samples ($\{x_t\}$) assigning a value $\widehat{y_t}$ to them.

Algorithm 1 JIT formulation for stationary scenarios

```
Use initial training data to build C_0 = (Z_0) and train the classifier
i = 0
while x_t is available do
  if y_t is available then
    U(C_i, {x_t, y_t}) → C_i
  else
    ŷ_t = K(Z_i, x_t)
  end if
end while
```

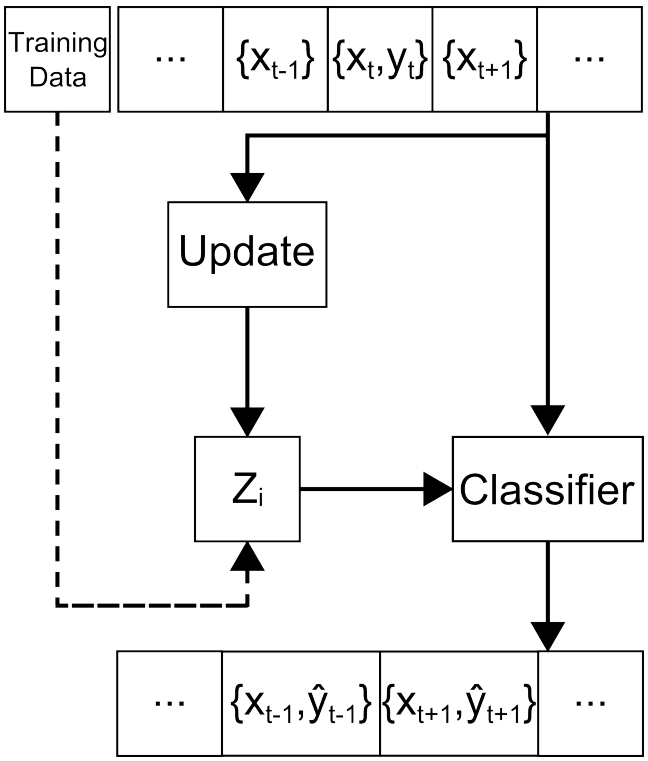


Figure 2.1: The simplified block diagram of the JIT framework for static environments

2.1 Classifier

The classifier used in the proposed JIT is the k-nearest neighbors (KNN). To choose the k value, we consider two options:

- $k = 1$: The value of k is fixed to 1 throughout the execution.
- Dynamic k: Change the value of k dynamically to achieve the best results.

In [2], Alippi et al. suggest using a recursive estimation of k based on the leave-one-out method (LOO). Basically, k is computed using the LOO procedure in the training set and afterwards, the following recursive formula is used to update the value of k:

$$k(n + \Delta n) = k_{LOO}(n) \left(\frac{n + \Delta n}{n} \right)^{\frac{4}{d+4}} \tag{2.1}$$

where $k_{LOO}(n)$ is the value of k computed with LOO using n samples in the knowledge base, Δn is the number of new supervised samples that were included in Z_i and d is the dimensionality of the instances. However, it is necessary to set a maximum number of samples Δn_{max} for which this formula can be safely used. The authors also suggest that this number should be around 25% of n. In other words, the LOO technique should be used every time the number of new samples gets bigger than 25% of the number of original samples in the KNN. This procedure is a good estimation of using k_{LOO} for every single update in the knowledge base, but with a much lower complexity cost. It is also more useful than other techniques to choose k, since it doesn't depend on a *priori* knowledge of the distribution of the data.

2.2 Experimental Protocol

In [3], Alippi et al. defined several synthetic dataset experiments in order to evaluate the performance of the JIT framework in comparison to other types of classifiers. In this work, four of those experiments are reproduced and used to achieve results comparable to theirs.

These experiments are denominated SCALAR_1, SCALAR_2, SCALAR_3 and SCALAR_4 and, as the names suggest, the data is scalar, i.e. the instances have only one feature. They model a two-class problem ($\Lambda = \{\omega_1, \omega_2\}$) and are composed of 10000 instances, 5000 from each class. One out of each 5 samples is supervised and the initial training set T_0 is composed of 400 samples (80 supervised). Also, concept drifts are present in the data and each pair class/concept is ruled by a Gaussian pdf presented in Table 2.1.

Table 2.1: Parameters of the pdf for each class/concept (mean and variance)

Concept	Class1	Class 2
C1	N(0,4)	N(2.5,4)
C2	N(2,4)	N(4.5,4)
C3	N(2.5,4)	N(0,4)

The differences between the experiments are:

- SCALAR_1: There is an abrupt concept drift at time $t = 5000$, when the concept changes from C_1 to C_2.
- SCALAR_2: Every 2000 samples the concept alternates between C_1 and C_2, which characterizes a recurrent concept drift.
- SCALAR_3: Every 2000 samples the concept alternates between C_1 and C_4, which characterizes a recurrent concept drift. In this scenario, the concept drift affects only the posterior probability $p(y|x)$.
- SCALAR_4: With initial concept C_1, every 2000 samples, the mean of each class is increased by 2.

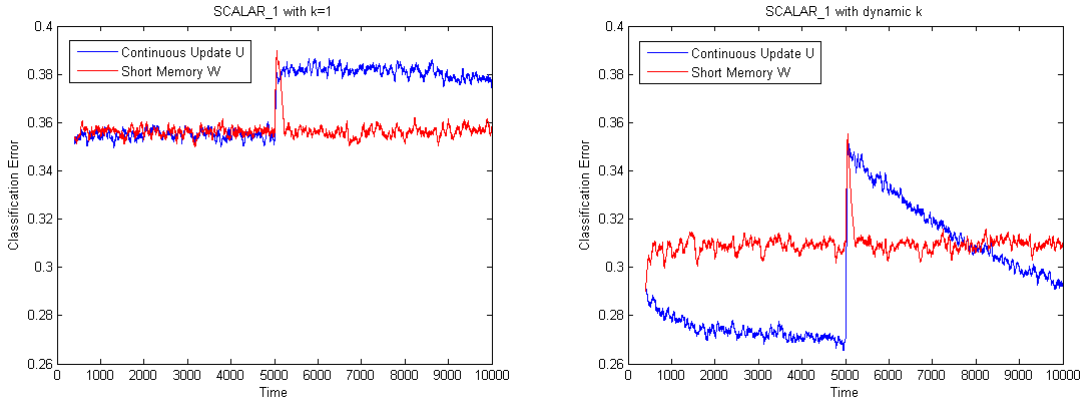
2.3 Results

Each experiment was run with dynamic k and $k = 1$ in order to show the gain we obtain by choosing k. Also, two different classifiers were tested:

- Short Memory Classifier W: This classifier is trained with the last 40 supervised samples and adapts naturally to concept drifts.
- Continuous Update Classifier U: Every time a new supervised instance arrives to the knowledge base, this classifier is retrained. For stationary conditions, this classifier achieves the best performance.

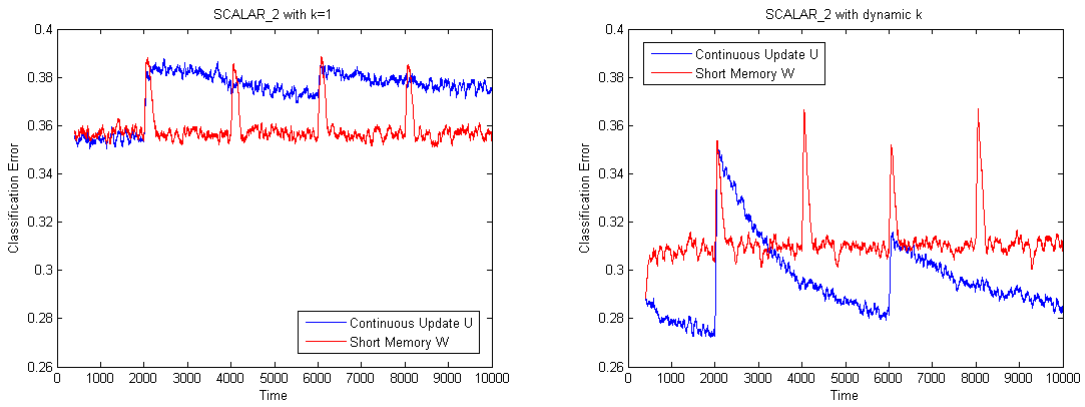
In Figure 2 of [3], we can observe a small difference in the performances of U and the JIT classifier in stationary conditions. However, this difference is due to the presence of false positives in the change detection mechanisms. As these mechanisms were not considered for this first question, the JIT classifier is equal to the continuous update classifier U.

The following Figures show the results in terms of classification error over time for all experiments and ks. They were computed as the average of 2000 replications. At each time t, the given error is the average of last 40 samples errors.



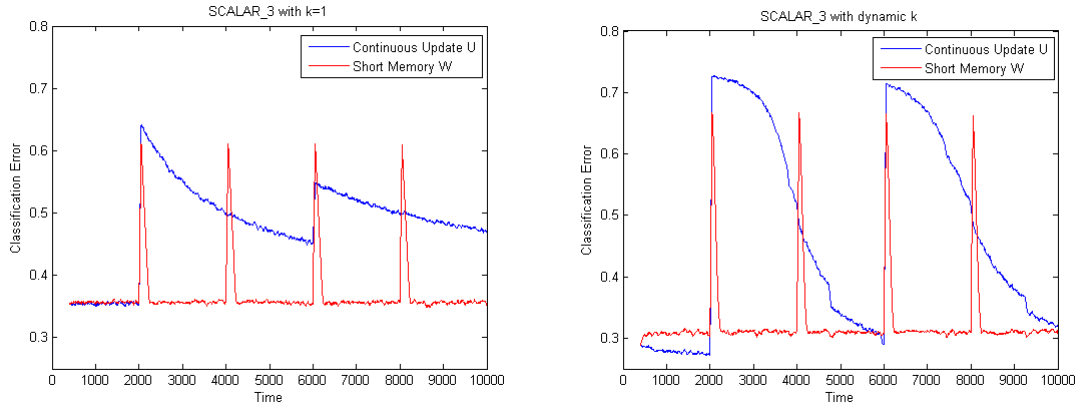
(a) (b)

Figure 2.2: The results of SCALAR_1 with (a) $k = 1$ and (b) dynamic k.



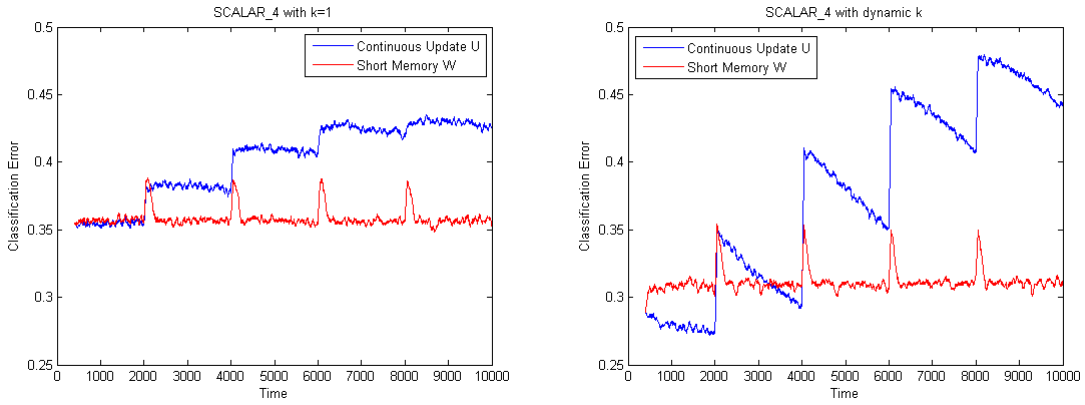
(a) (b)

Figure 2.3: The results of SCALAR_2 with (a) $k = 1$ and (b) dynamic k.



(a) (b)

Figure 2.4: The results of SCALAR_3 with (a) $k = 1$ and (b) dynamic k.



(a) (b)

Figure 2.5: The results of SCALAR_4 with (a) $k = 1$ and (b) dynamic k.

Table 2.2: Average classification error (%) of W and U for each experiment.

Experiment	**Short Memory (W)**	**Continuous Update (U)**
SCALAR_1 & k=1	35.66	36.84
SCALAR_1 & dynamic k	30.95	29.58
SCALAR_2 & k=1	35.83	37.44
SCALAR_2 & dynamic k	31.20	29.57
SCALAR_3 & k=1	36.87	48.27
SCALAR_3 & dynamic k	32.56	47.52
SCALAR_4 & k=1	35.83	40.15
SCALAR_4 & dynamic k	31.13	37.82

In Figures 2.2, 2.3, 2.4 and 2.5, we can observe the impact of choosing k properly. The error rate is decreased by a considerable amount. The improvement over time of the continuous update classifier is also noticeable, specially in *(b)*. All results computed with dynamic k are

comparable to those presented in Figure 2 of [3]. The table 2.2 presents the average classification error of the entire dataset for all experiments. These results are also similar to those presented in Table I of [3].

Chapter 3

B

3.1 JIT for changing environments

The JIT framework presented in question A uses the continuous update classifier U to classify unsupervised samples. However, in changing environments (as seen in the experiments) this classifier does not perform well as it keeps data from all concepts in its knowledge base. In order to improve the classification accuracy, it is possible to try to detect concept drifts and then remove stale samples from the classifier's knowledge base.

For this question, the ability of detecting changes is added to the JIT. Therefore, we expect the framework to adapt to the changes, lowering the classification error. For this part of the dissertation, only classifiers with dynamic k values will be used, since they were proven better than fixing $k = 1$.

The JIT framework proposed by Alippi et al. in [3] uses two change detection tests (CDTs) that monitor changes in the data generation process and in the posterior probabilities $P(Y|X)$ and are called CDT_X and CDT_ϵ. If at least one of the CDTs states that a change occurred, a concept drift is assumed and several procedures are run to adapt the classification system. Both CDTs use the ICI-rule to detect changes.

As proposed in [8], both CDTs are formulated with an hierarchical scheme: In the first level, the ICI rule is used to detect a change, which is then confirmed by the second level using a Hotelling T^2 test.

As stated before, the JIT is composed of a set of concepts with three subsets and five operators. In the last chapter, only one of the subsets of the concept and two operators were introduced: Z_i, $U(C_i, R)$ and $K(Z, x_t)$ – those related to updating the classifier over time. Thus, the remaining concept subsets and operators are:

- F_i: the set of features that characterize the i^{th} concept. It consists of two features: mean and variance. Let the sequence S consist of vX samples $X_1, X_2, ..., X_n$, then the mean is given by:

$$M(S) = \frac{1}{vX} \sum_{t=1}^{vX} X_t \tag{3.1}$$

and the quadratic deviation by:

$$S(S) = \sum_{t=1}^{vX} (X_t - M(S))^2 \tag{3.2}$$

and finally the sample variance by:

$$V(S) = \frac{S(S)}{vX - 1} \tag{3.3}$$

While the mean $M(S)$ approaches a Gaussian distribution, the variance V(S) doesn't. Because approaching a Gaussian distribution is a requirement to perform change detection with the ICI rule , it is necessary to apply a power-law transform to the variance:

$$V(S) = (\frac{S(S)}{vX - 1})^{h_0} \tag{3.4}$$

Basically, this power-law transform consists in approximating the given distribution to a Gaussian. To compute the h_0 exponent Mudholkar and Trivedi [9] presented the following formula:

$$h_0 = 1 - \frac{k_1 k_3}{3k_2^2} \tag{3.5}$$

where k_i is the i^{th} cumulant of the gaussian approximated distribution. To calculate k_1, k_2 and k_3 it is necessary to compute the first six cumulants of the sample distribution ζ_1, ..., ζ_6 and use the formulas below:

$$k_1 = n - 1 \tag{3.6}$$

$$k_2 = (n-1)^2[\frac{\zeta_4}{n\zeta^2} + \frac{2}{n-1}] \tag{3.7}$$

$$k_3 = \frac{(n-1)^3}{\zeta_2^3}(\frac{\zeta_6}{n^2} + \frac{12\zeta_4\zeta_2}{n(n-1)} + \frac{4(n-2)\zeta_3^2}{n(n-1)^2} + \frac{8\zeta_2^3}{(n-1)^2}) \tag{3.8}$$

In order to compute the cumulants of the sample distribution, it is first necessary to find their moments, which are given by:

$$\mu_t' = E(X^t) \tag{3.9}$$

which can be considered as the mean.

By computing the moments, the cumulants can be calculated using the recursive formula:

$$\zeta_r = \mu'_r - \sum_{m=1}^{r-1} \binom{r-1}{m-1} \zeta_m \mu'_{r-m} \tag{3.10}$$

- D_i: the set of features to check for changes in the i^{th} concept. Since the change detection mechanism monitors nonstationarities in the features that characterize the concept, F_i is present in D_i. Another feature that is present in this set is P_i: a sequence with estimates of classification errors of subsequences of Z_i. In other words:

$$P_i = \left\{ \frac{1}{vE} \sum_{i=1}^{vE} \epsilon_i \right\} \tag{3.11}$$

  with the element-wise error ϵ_i as:

$$\epsilon_i = \begin{cases} 0, & \text{if } y_t = K_0(x_t) \\ 1, & \text{otherwise} \end{cases} \tag{3.12}$$

  K_0 is a classifier used to compute the errors for P_i. Finally, $D_i = (F_i, P_i)$.

- The concept drift operator $D(C_i) \rightarrow \{0, 1\}$ receives the features extracted from nonoverlapping sequences of inputs (D_i) and uses the ICI rule to determine whether a drift occurred or not. This operator uses CDT_X and CDT_ϵ:

$$D(C_i) = (CDT_X(F_i)||CDT_\epsilon(P_i)) \in \{0, 1\} \tag{3.13}$$

  The first level of the CDTs use the ICI rule [6], which basically consists in configuring the CDTs with the features extracted from a training set, creating intervals of confidence and updating those intervals according to new data. When at least one of the intervals becomes empty, a change is detected.

  For instance, given $\{M(s), s = 1, \ldots, S_0\}$ with S_0 as the number of sequences of size v extracted from the initial training set, the configuration phase is done by computing the interval $I^M_{S_0}$:

$$\hat{\mu}^M_{S_0} = \sum_{s=1}^{S_0} \frac{M(s)}{S_0} \tag{3.14}$$

$$\hat{\sigma}^M = \sqrt{\sum_{s=1}^{S_0} \frac{(M(s) - \hat{\mu}^M_{S_0})^2}{S_0 - 1}} \tag{3.15}$$

$$\hat{\sigma}^M_{S_0} = \frac{\hat{\sigma}^M}{\sqrt{S_0}} \tag{3.16}$$

$$I_{S_0}^M = \left[\hat{\mu}_{S_0}^M - \Gamma\hat{\sigma}_{S_0}^M; \hat{\mu}_{S_0}^M + \Gamma\hat{\sigma}_{S_0}^M\right] \tag{3.17}$$

Afterwards, for each new sequence of size v, the new value of $M(s)$ is computed as well as the values of $\hat{\mu}_S^M$ and $\hat{\sigma}_{S_0}^M$ and the new interval I_S^M:

$$\hat{\mu}_S^M = \frac{(S-1) * \hat{\mu}_{S-1}^M + M(s)}{s} \tag{3.18}$$

$$\hat{\sigma}_S^M = \frac{\hat{\sigma}^M}{\sqrt{S}} \tag{3.19}$$

$$I_S^M = \left[\hat{\mu}_S^M - \Gamma\hat{\sigma}_S^M; \hat{\mu}_S^M + \Gamma\hat{\sigma}_S^M\right] \bigcap I_{S-1}^M \tag{3.20}$$

When I_S^M becomes an empty set ($\emptyset$), a change is detected in S. By changing the feature $M(s)$, we can generalize the configuration and update of the CDT.

To improve the estimation of the time change, a refinement procedure is applied [6]. Considering the detection happened in time $\hat{T}_i$, the refinement procedure computes $Tref_i$, a more precise estimate of T^* (the real instant of the change), by dividing the interval $[\hat{T}_{i-1}; \hat{T}_i]$ using a constant λ and $j = 1$ initially in:

$$T_j = \hat{T}_{i-1} + (\hat{T}_i - \hat{T}_{i-1})/\lambda \tag{3.21}$$

The CDT is used to detect a change $\hat{T}_j$ in $[Tref_{i-1}; \hat{T}_{i-1}] \bigcup [T_j; \hat{T}_i]$ where $Tref_{i-1}$ is the refinement of $\hat{T}_{i-1}$ or 0 in the case of the first detection. The procedure repeats itself by dividing the search space again. The break condition is finding the minimal value of $\hat{T}_j$ (T_{min}) that is smaller than the beginning of the next search space (T_j). Finally, $Tref_i = T_{min}$.

The last step of this operator is to confirm the change using the newly acquired $Tref_i$. Two sequences of data are considered: the new concept candidate samples $[Tref_i; \hat{T}_i]$ and the previous concept observations $[Tref_{i-1}; \hat{T}_{i-1}]$. Considering the means of those sequences as $\bar{F}_i$ and $\bar{F}_{i-1}$, respectively, a Hotelling T^2 test is done by inspecting the null hypothesis H_0: $\bar{F}_i - \bar{F}_{i-1} = [0; 0]$. If the hypothesis is rejected with a confidence level α, the change is confirmed. Otherwise, the change is discarded and the CTD is reconfigured with the samples from $[Tref_{i-1}; \hat{T}_{i-1}]$.

- The split operator $\Upsilon(C_i) \rightarrow (C_j, C_k)$ separates a given concept into two others. To split the concept properly, using $Tref_i$ is not an option as it overestimates T^*. The solution is to define the time at which the concept ended. This can be done by configuring the CDT with the new data $[Tref_i; \hat{T}_i]$ and using it backwards to find a change in the first concept $[Tref_i; Tref_{i-1}]$. The detection $Tend_{i-1}$, which underestimates T^* is computed. Finally, the concept is split:

$\Upsilon(C_i) \rightarrow (C_i, C_{i-1})$ where

$$\begin{cases} C_i = (Z_{i[Tref_i;\hat{T}_i]}, F_{i[Tref_i;\hat{T}_i]}) \\ C_{i-1} = (Z_{i[Tref_{i-1};Tend_{i-1}]}, F_{i[Tref_{i-1};Tend_{i-1}]}) \end{cases} \tag{3.22}$$

- The equivalence operator $\varepsilon(C_i, C_j) \rightarrow \{0, 1\}$ determines if two distinct concepts are equivalent, i.e. generated by the same distribution. This feature is used to support recurrent concepts. To determine if two concepts are equivalent, both F_i and Z_i must be equivalent:

$$\varepsilon(C_i, C_j) = \begin{cases} 1, & \text{if } F_i \text{ is equivalent to } F_j \text{ and } Z_i \text{ is equivalent to } Z_j \\ 0, & \text{otherwise} \end{cases} \tag{3.23}$$

  To compare F_i to F_j, two one-sided tests are used. Similarly to the Hotelling T^2 test in the change validation procedure, these tests compare the means of the sets by formulating a null and an alternative hypothesis. In order to compare, Z_i to Z_j, both subsets are split in TS_i and VS_i and TS_j and VS_j, respectively, with $|TS_i| = |TS_j|$. The classifiers K_i and K_j are trained using TS_i and TS_j and used to classify the samples of $VS = max(VS_i, VS_j)$. If the percentage of samples q for which K_i and K_j provide the same classes on VS is bigger than τ, Z_i and Z_j are equivalent.

3.2 Simplified JIT

For this dissertation, a simplified version of the JIT proposed in [3] was implemented. Instead of having CDT_X and CDT_ϵ, only the first one was used. Also, the configuration parameters are equal to those proposed in the original work.

In Algorithm 2 and in Figure 3.1 a general overview of the simplified JIT framework for dynamic scenarios and a block diagram are presented. The concept $C_0 = (F_0, Z_0)$ is trained with the training data (dashed line in Figure 3.1). Then, during the operation phase of the system, new supervised samples (represented by $\{x_t, y_t\}$) are used to update the value of Z_i, which is used to retrain the classifier. Also, new samples (supervised or not) are used to compute the features of F_i. The change detection test monitors changes in F_i and when a drift occurs, the current concept is split. The reconfiguration of the CDT_X uses the new F_i set and the equivalence procedure adds supervised samples of similar concepts in Z_i. Finally the classifier assigns the label $\widehat{y}_t$ to all unsupervised samples ($\{x_t\}$).

3.3 Results

In this section, the experiments followed the same experimental protocol as before, but with a new classifier: the simplified JIT classifier. The values of all parameters are equal to those presented in the original work. Four graphics, one for each scalar problem, are presented in

Algorithm 2 JIT formulation for dynamic scenarios

Use initial training data to build $C_0 = (Z_0, F_0)$
Train K with Z_0 and configure CDT_X with F_0
$v_X = 20$ and $MX = 4$
$i = 0$ and $Q_X = \emptyset$
while x_t is available **do**
  $Q_X = Q_X \bigcup \{x_t\}$
  if y_t is available **then**
    $Z_i = Z_i \bigcup \{x_t, y_t\}$
    Update k value of K
  end if
  if $\#Q_X = v_X$ **then**
    Compute new features from Q_X and insert into F_i
    $[rx, Tref] = CDT_x(F_i)$
    $Q_X = \emptyset$
  end if
  if $rx = 1$ **then**
    $i = i + 1$
    Estimate $Tend_{i-1}$ with CDT_x configured on C_i operating backwards on F_i
    Split C_i and C_{i-1}
    Train K with Z_i
    $reconfigure = 1$
  end if
  if $reconfigure = 1$ and $\#F_i \geq MX$ **then**
    Configure CDT_X with F_i
    $Z_{rec} = \emptyset$
    for $j = 0; j < i; j++$ **do**
      if $\varepsilon(C_i, C_j) = 1$ **then**
        $Z_{rec} = Z_{rec} \bigcup Z_j$
      end if
    end for
    Train K with $Z_i \bigcup Z_{rec}$
    $reconfigure = 0$
  end if
  if y_t is not available **then**
    $\widehat{y_t} = K(Z_i, x_t)$
  end if
end while

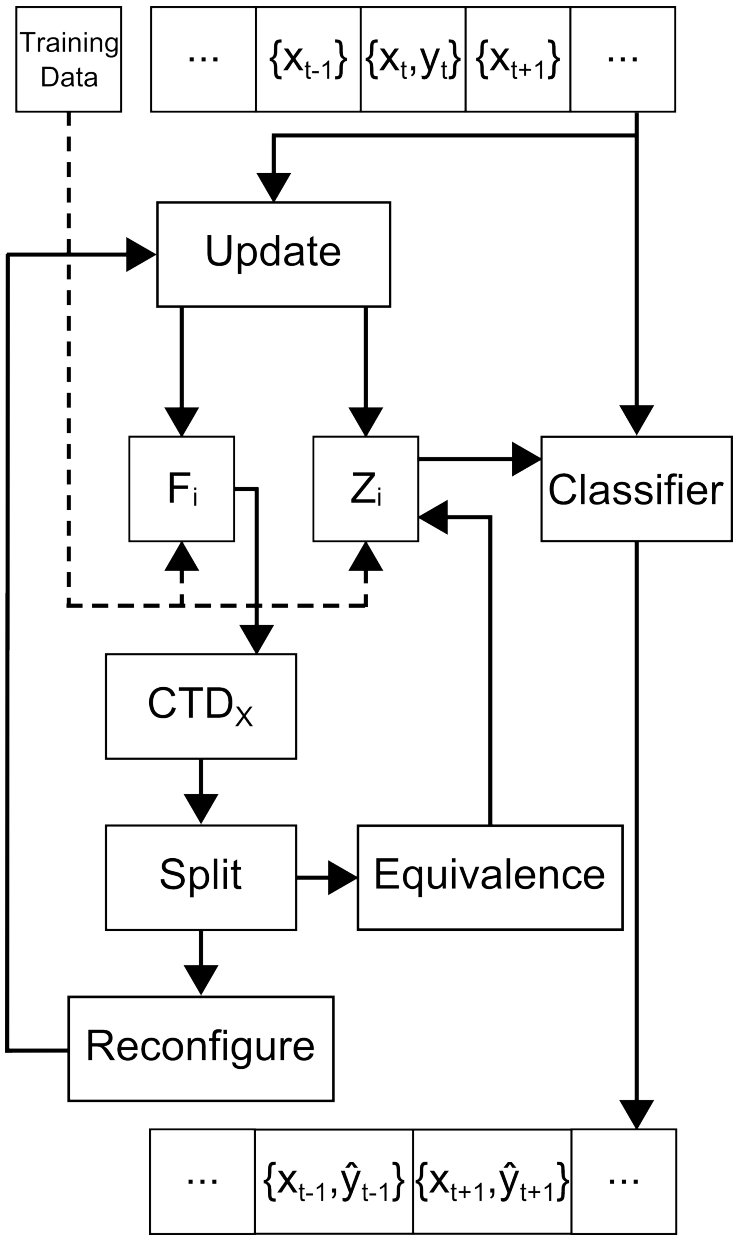


Figure 3.1: The simplified block diagram of the JIT framework for static environments

following Figures. Also, the table 3.1 presents the average classification error of the entire dataset for each classifier.

In general, the results are similar to those reported in [3], but for the third case, SCALAR_3. This is due to the fact that, in this dataset, there is no difference between the distribution of the data of each concept, since a class swap occurred. To be able to properly identify the concept drift, the CDT_ϵ would be necessary.

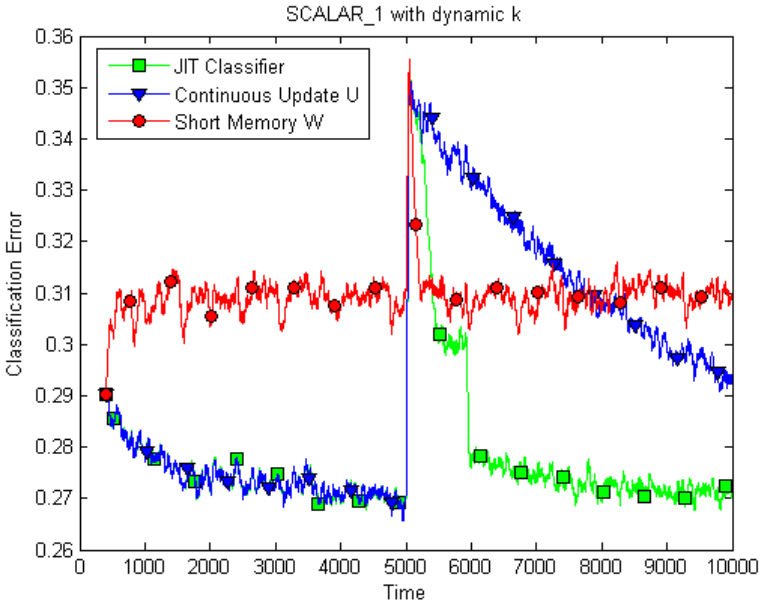


Figure 3.2: The results of SCALAR_1 for JIT for changing environments.

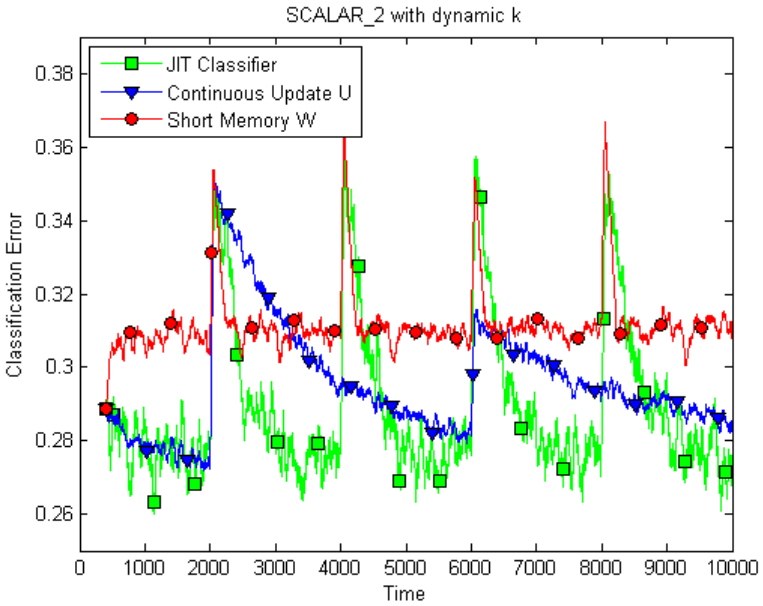


Figure 3.3: The results of SCALAR_2 for JIT for changing environments.

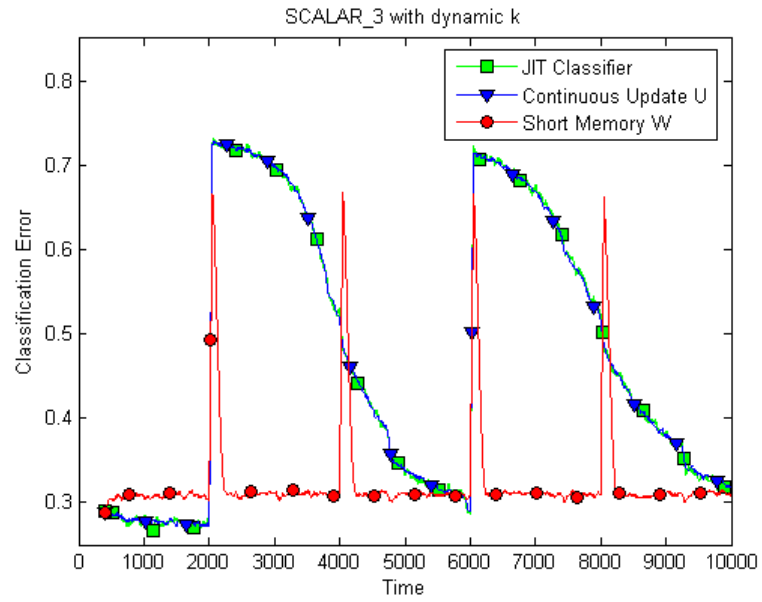


Figure 3.4: The results of SCALAR_3 for JIT for changing environments.

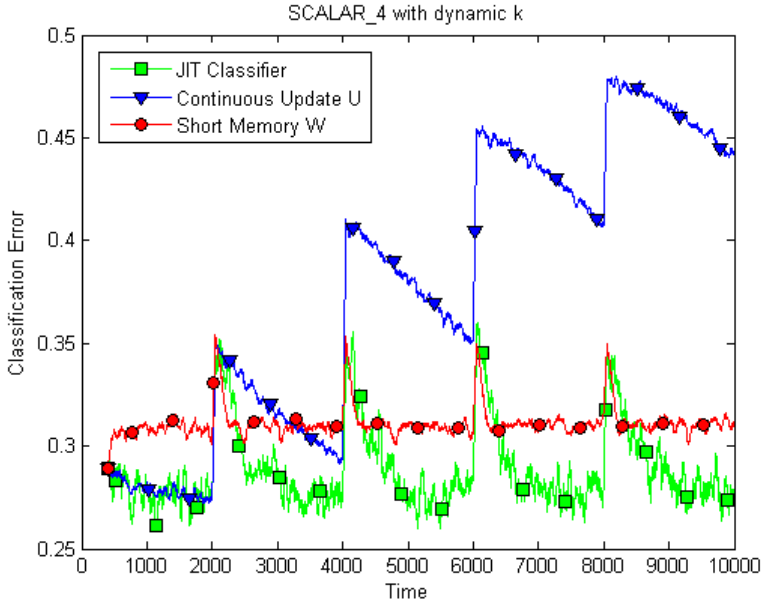


Figure 3.5: The results of SCALAR_4 for JIT for changing environments.

Table 3.1: Average classification error (%) of W, U and JIT for each experiment.

Experiment	**Short Memory (W)**	**Continuous Update (U)**	**JIT Classifier**
SCALAR_1	30.95	29.58	27.79
SCALAR_2	31.20	29.57	28.91
SCALAR_3	32.56	47.52	47.52
SCALAR_4	31.13	37.82	28.87

Chapter 4

C

4.1 1)

In pattern recognition, the concept drift is a common phenomenon in classification systems. Over time, the data being analyzed can have its statistical properties altered for several reasons such as problems with the sensor, noises or real changes in the measured variable. Often, these alterations lead to a decrease in the accuracy of the system, which was trained to work with data before the drift occurred. Therefore, the term concept drift refers to the changes that can affect the classes of the problem.

4.2 2)

The need to guarantee a good and stable performance throughout the entire operation phase of a classification system has inspired several authors to propose methods to adapt these systems in dynamic environments. The techniques available in the literature can be divided in two groups regarding the way they react to the concept drifts. Passive classifiers don't detect the changes directly, but usually adapt to it by modifying the weights of the fusion function used to create an ensemble of classifiers and by retraining for every new supervised sample. The short memory classifier W is an example of a passive classifier. Active classifiers monitor the data to find when changing points occur. Once this detection happens, the classifier is retrained with samples of the current concept only. The JIT framework uses active change detection tests. Some of the best known techniques for detecting concept drifts are:

In [10], Gama et al. present the method called Statistical Process Control (SPC), which monitors the error rate and defines a warning threshold and a detection threshold. When the error becomes larger than the warning threshold, the time instant is stored. If the error lowers, the warning is lifted. But, if the error continues to increase, surpassing the detection threshold, a change is detected. The classifier is updated with samples collected between the warning and the change confirmation.

Bifet and Gavaldà [11] propose a system called ADWIN, which uses an adaptive window W. The size of this window is recomputed according to the rate of change observed in the

data. During stationary periods, the window will increase in size and when a change occurs, it will shrink to discard old data. To detect the change, the averages of two sub-windows are computed and compared. If they are distinct enough, regarding a cut threshold, the older one is discarded.

Sebastião et al. [12] introduce the Fixed Cumulative Windows Model (FCWM), which creates the reference window, which contains old data, and the current window, that possesses new data. These two windows are compared using the Kullback-Leibler divergence. If the divergence is bigger than a predefined threshold, the change is detected.

In [13], Page explains the Page-Hinkley Test (PHT). This method computes the differences between each value (x_t) of a sequence of data and its mean ($\bar{x_t}$). These differences are accumulated (U_t) and a change is detected if the difference between the current variable of differences U_t and the previous minimum computed value is higher than a predefined threshold.

The main advantage of the JIT proposed by Alippi et al. in [3] is that it monitors changes in the input data and in the classification error, which allows it to be more robust to changes. Another advantage of the JIT framework is that the change detection mechanism is independent of the classifier being used. There are several approaches that create change detection mechanisms that are bound to a certain types of classifiers. On the other hand, the main disadvantage of the JIT is configuration of the CDTs. This requires a minimum number of training samples, which can delay the availability of the CDT right after a change is detected. Also, to be able to use CDT_ϵ many supervised observations are required.

4.3 3)

Analyzing Figure 3.2, we can observe the behaviour of the continuous update classifier U and the JIT classifier, which contains a change detection mechanism. At time $t = 5000$ a concept drift occurs, which increases the classification error of both classifiers. The continuous update classifier continues to update its knowledge base with supervised samples of the new concept. However, observations from the previous concept are still used to classify unlabelled samples, making the decrease of the error slow. On the other hand, the change detection test of the JIT classifier identifies the concept drift and discards instances belonging to the old concept from the knowledge base. This procedure contributes to a much faster reduction of the error. Therefore, change detection is relevant to maintaining the accuracy of a classifier system.

Bibliography

[1] C. Alippi and M. Roveri, "Just-in-time adaptive classifiers - part i: Detecting nonstationary changes," *Trans. Neur. Netw.*, vol. 19, no. 7, pp. 1145–1153, 2008.

[2] C. Alippi and M. Roveri, "Just-in-time adaptive classifiers - part ii: Designing the classifier," *Trans. Neur. Netw.*, vol. 19, no. 12, pp. 2053–2064, 2008.

[3] Cesare Alippi, Giacomo Boracchi, and Manuel Roveri, "Just-in-time classifiers for recurrent concepts," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 24, no. 4, pp. 620–634, 2013.

[4] Cesare Alippi, Giacomo Boracchi, and Manuel Roveri, "Just-in-time ensemble of classifiers," in *International Joint Conference on Neural Networks*, 2012, pp. 1–8.

[5] Cesare Alippi, Giacomo Boracchi, and Manuel Roveri, "Just in time classifiers: Managing the slow drift case," in *International Joint Conference on Neural Networks*, 2009, pp. 114–120.

[6] Cesare Alippi, Giacomo Boracchi, and Manuel Roveri, "A just-in-time adaptive classification system based on the intersection of confidence intervals rule," *International Joint Conference on Neural Networks*, vol. 24, no. 8, pp. 791–800, 2011.

[7] Cesare Alippi, Giacomo Boracchi, and Manuel Roveri, "An effective just-in-time adaptive classifier for gradual concept drifts," in *International Joint Conference on Neural Networks*, 2011, pp. 1675–1682.

[8] C. Alippi, G. Boracchi, and M. Roveri, "A hierarchical, nonparametric, sequential change-detection test," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, 2011, pp. 2889–2896.

[9] Govind S. Mudholkar and Madhusudan C. Trivedi, "A gaussian approximation to the distribution of the sample variance for nonnormal populations," *Journal of the American Statistical Association*, vol. 76, no. 374, pp. 479–485, 1981.

[10] João Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues, "Learning with drift detection," in *In SBIA Brazilian Symposium on Artificial Intelligence*, 2004, pp. 286–295.

[11] Albert Bifet and Ricard Gavaldà, "Learning from time-changing data with adaptive windowing," in *In SIAM International Conference on Data Mining*, 2007.

[12] Raquel Sebastião, João Gama, Pedro Pereira Rodrigues, and João Bernardes, "Monitoring incremental histogram distribution for change detection in data streams," in *Proceedings of the Second international conference on Knowledge Discovery from Sensor Data*, 2010, pp. 25–42.

[13] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1, pp. 100–115, 1954.