Reducing Computational Complexity
in $k$-NN based Adaptive Classifiers

Cesare Alippi and Manuel Roveri
Politecnico di Milano, DEI, Milano, Italy
{alippi,roveri}@elet.polimi.it

Abstract—Integrating new information in intelligent measurement systems during their operational life is always profitable from the accuracy point of view but it generally induces an increment in the complexity of the classifier. Adaptive classifiers, which provide adaptive mechanisms to update their knowledge base over time, are able to exploit fresh information to improve accuracy but, traditionally, do not consider complexity issues. In this paper we propose a design solution for adaptive classifiers able to reduce the computational complexity and the memory requirements of $k$-NN classifiers by including condensing editing techniques. Moreover, we propose a novel approach for estimating the incoming innovation content which allows us for not including redundant or superfluous information (thus minimizing the knowledge base size).

I. INTRODUCTION

The capability to integrate and exploit additional supervised information provided by the process during the operational life is a key issue in a wide range of applications (e.g., environmental and industrial monitoring, medical applications and character recognition).

In intelligent measurement systems developed for such applications, the presence of fresh knowledge coming from the process requires adaptive mechanisms to exploit the innovation content to improve (when possible) the classification accuracy and, hence, the performance of the application.

[1][2][3] address the asymptotic theory of adaptive classifiers (i.e., how the accuracy improves over incremental data sets), while a relative large literature exists on adaptive classifiers developed for specific application fields (e.g., manufacturing systems [4], aerial surveillance [5], condition monitoring [6] and handwritten character recognition [7]).


As explained in [13], the main drawbacks of the traditional approaches are the continuous updating of the knowledge base (even when it is not necessary) and the not adequate description of the effects of introducing fresh information into an already existing knowledge base [10] (leading to limited description to the relationship between the proposed approaches and the asymptotic behavior defined by the theory).

The authors proposed in [13] a methodology for designing Just-In-Time (JIT) adaptive classifiers which, by exploiting novel supervised information coming from the field during operational life (and constituting an incremental knowledge base), modify, whenever appropriate, the knowledge base associated with the classification system to maintain/improve its classification accuracy.

Unfortunately, the introduction of fresh knowledge in adaptive classifiers, which is always profitable from the accuracy point of view, induces an increment in the computational complexity and the memory requirements (reducing the effectiveness of the proposed approach in embedded systems).

The aim of this paper is to extend the JIT adaptive classifiers presented in [13] by including complexity reduction techniques to reduce the complexity of the classifier. Moreover, the novelty of the proposed approach resides also in the evaluation of the innovation content present in the additional knowledge to decide whether to introduce it in the knowledge base (or not).

The structure of the paper is as follows. Section II introduces the aspects related to the complexity reduction techniques in $k$-NN classifiers. The methodology addressing the novel adaptive classifiers implementing complexity reduction techniques is presented in Section III; experiments are finally given in Section IV.

II. REDUCING THE COMPLEXITY

As stated in [13], among the classifier families that support consistent learning rules (e.g., NNs, SVMs and $k$-NNs), $k$-NN classifiers are particularly appealing for their immediate training phase and easy knowledge management.

Unfortunately, the introduction of fresh information in the knowledge base of the $k$-NN classifier induces an increase both in the memory requirement and in the computational complexity of the classifier\(^1\). As such, it is generally profitable to reduce the cardinality of the knowledge base of the classifier:\(^1\)

\(^1\) The computational complexity and the memory requirement of the $k$-NN are $O(kn)$ and $O(n)$, respectively. $n$ is the training set size.
$k$-NN to speed-up the computation and reduce the memory occupation during the operational life of the classifier.

As presented in [16], the aim of condensing and editing techniques is to maintain the smallest set of training instances yet preserving the classification accuracy on the training set. This can be achieved by removing both noisy samples (that cause wrong classification) and superfluous samples (which are redundant, in the sense that they are dominated by others).

We considered several condensing and editing techniques which can be grouped into two categories reflecting either an heuristic or a geometric approach.

The heuristic approach, which includes Condensed Nearest Neighbor [17], Reduce Nearest Neighbor [18] and the Wilson Editing rule [19], is based on the concept of minimal consistent subset of the training set (which is the minimum subset of the training set that classifies correctly all the remaining samples of the training set). The heuristic techniques are generally simple and easy-to-use but do not preserve the decision boundary and are affected by the sample visiting order.

Differently, the geometric techniques are particularly elegant solutions which aim at reducing the knowledge base by maintaining the decision boundary [20]. Such techniques, which are based on the Voronoi diagram of the training set, remove nodes of the Voronoi regions whose boundaries do not contribute to the decision boundary. This method preserves the decision boundary and generates the smallest, boundary-preserving edited set. Unfortunately, computing the Voronoi diagram in high dimensions is very costly; for practical applications we suggest to consider some approximate solutions. Both the Gabriel Graph Rule [20] and the Relative Neighbor Graph Rule [20] provide such competence preservation algorithms with different levels of approximation.

We focus on the Gabriel Graph Rule for its efficacy and its reduced computational complexity\(^2\). The Gabriel Graph rule, which is an approximate solution of the Voronoi one, provides higher reduction capabilities (than the Voronoi editing rule) but does not preserve the original decision boundary. However, the differences in the decision boundary (between the Voronoi algorithm and the Gabriel Graph rule) generally occur outside the zones of interest (see Figure 1).

As presented in [16], the availability of a priori information about process (e.g., the probability density function as well as the class separation boundary -or Bayes boundary-) allows the classifier designer for selecting the most appropriate condensing and editing technique.

---

\(^2\) The computational complexity of the Gabriel Graph rule is $O(n^2)$, while an heuristic version reduces the complexity to $O(n^3)$.

### III. THE JIT-G ADAPTIVE CLASSIFIER

To exploit advantages both of the JIT adaptive classifiers (in terms of accuracy) and of the Gabriel’s rule (in terms of complexity reduction), we defined the JIT-G adaptive classifier by extending the JIT adaptive classifier and including the Gabriel Graph complexity reduction technique. In addition, we developed a novel and intuitive evaluation of the innovation content provided by additional supervised samples which is based on the boundary preserving property of the Gabriel Graph rule:

1. The initial knowledge base is edited with the Gabriel Graph rule.
2. If additional knowledge updates the current decision boundary, it is included in the knowledge base of the classifier; otherwise, discarded.

The additional information is thus included in the knowledge base of the classifier (augmenting the complexity of the classifier) only if it provides a meaningful innovation content to the classifier.

Algorithm 1 summarizes the proposed adaptive classifier implementing the Gabriel Graph rule.

<table>
<thead>
<tr>
<th>Algorithm 1 : JIT Adaptive Classifier with Gabriel Graph rule (JIT-G)</th>
</tr>
</thead>
</table>
| 1. $KB_{GG} = $ Gabriel Graph rule ($KB_{init}$);  
2. while (1) {  
3. if (new knowledge is available) {  
4. $KB_{tmp} = $ Gabriel Graph rule ($KB_{GG} \cup IKB$);  
5. if (decision boundary of $KB_{tmp} = $ decision boundary of $KB_{GG}$) {  
6. $KB_{GG} = KB_{tmp}$;  
7. else  
8. Discard $IKB$;  
9. }  
10. $k$-NN classifier ($s, KB_{GG}$); }  |

More in detail, the initial knowledge base $KB_{init}$ is edited with the Gabriel Graph rule to obtain $KB_{GG}$ characterizing the initial $k$-NN classifier (step 1).
When new knowledge $IKB$ is provided to the classifier (step 3), it is not automatically inserted in the knowledge base of the classifier (differently from approaches present in the literature that force a continuous updating of the knowledge). The Gabriel Graph edited set of $KB_{GG} \cup IKB$ is computed (step 4) to obtain $KB_{tmp}$. If the decision boundary of $KB_{tmp}$ differs from the $KB_{GG}$ one, the additional knowledge $IKB$ has a meaningful innovation content (updating the current decision boundary –step 5-) and the current Gabriel Graph edited knowledge base $KB_{GG}$ is updated with $KB_{tmp}$ (step 6). If the additional knowledge does not modify the decision boundary (step 7), it can be considered superfluous since it is dominated by other knowledge already present in the knowledge base. In this case $IKB$ is neglectable and hence can be discarded.

Finally, new incoming input samples $x$ are classified with the $k$-NN classifier based on $KB_{GG}$.

IV. EXPERIMENTAL RESULTS

To validate the effectiveness of the suggested algorithm, we considered two applications:

Application - D1- refers to a synthetic two-dimensional classification problem where two equiprobable classes $(\omega_1, \omega_2)$ configure the process generating the data and are ruled by 2-dimensional Gaussian distributions (each experiment lasts 1000 samples):

$$p(x | \omega_1) = N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}\right), \quad p(x | \omega_2) = N\left(\begin{bmatrix} 10 \\ 0 \end{bmatrix}, \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}\right).$$

Application -D2- refers to gas -Self-Assembled-Monolayers (SAM) [21] sensors. The application considers a set of five SAM gas sensors with two features (the sensor measurement and its gradient) extracted from each signal. Each experiment lasts 1000 samples.

Algorithm 2: Experiments on just-in-time adaptive classifiers, stationary conditions

1. The initial knowledge base $KB_{init}$ of the JIT-G classifier is defined as the 10% of the training set;
2. Apply the JIT-G classifier to incoming data;
3. The remaining samples of the training set (and its supervised classification) are provided to the JIT-G classifier one at a time;
4. After each new sample, the accuracy of the JIT-G classifier is computed on the validation set.

To evaluate the proposed adaptive classifiers we defined three figures of merit:

- Accuracy: accuracy is computed by evaluating, at time $t$, the percentage of correct classification of a previous time window of 200 samples;
- Size of the knowledge base, i.e., the number of samples that compose the knowledge base of classifier at time $t$;
- Computational time: it provides an estimate of the execution time (in ms) needed to perform the experiment (reference platform: Intel Centrino 1.7 GHz, 1Gb RAM, Windows Xp, all unnecessary processes aborted).

For each application, experiments are organized as presented in Algorithm 2. To compare the performance of the JIT-G adaptive classifier we also considered a reference $k$-NN classifier trained on a knowledge base composed (at time $t$) by the initial training set and all the samples provided to the JIT-G classifier up to the $t$-th instant. All experiments have been averaged over 150 runs.

In Table 1 the average simulation results of the two classification approaches (with $k$=1,3,5) are given for application D1 and D2.

Table 1 – Simulation results comparison between the JIT-G classifier and the traditional $k$-NN

<table>
<thead>
<tr>
<th></th>
<th>JIT-G</th>
<th>$k$-NN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$k=1$</td>
<td>$k=3$</td>
</tr>
<tr>
<td><strong>D1</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Acc. (%)</strong></td>
<td>91.5</td>
<td>92.9</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>18.4</td>
<td>18.9</td>
</tr>
<tr>
<td><strong>Time (ms)</strong></td>
<td>63.0</td>
<td>64.9</td>
</tr>
<tr>
<td><strong>D2</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Acc. (%)</strong></td>
<td>90.2</td>
<td>90.5</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>46.1</td>
<td>50.8</td>
</tr>
<tr>
<td><strong>Time (ms)</strong></td>
<td>21.8</td>
<td>22.2</td>
</tr>
</tbody>
</table>

As presented in Table 1, simulation results are very satisfactory both in application D1 and in application D2. By accepting a very small loss in accuracy (less than 2%), the JIT-G is able to reduce significantly both the number of samples stored in the knowledge base (70% reduction) and the computational time (35% reduction).

Obviously, in case of JIT-G adaptive classifiers we have to consider also the computational overhead caused by the adaptive Gabriel Graph rule.

Table 2 – Computational overhead caused by the adaptive Gabriel Graph rule

<table>
<thead>
<tr>
<th></th>
<th>JIT-G</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D1</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Time (ms)</strong></td>
<td>1.4</td>
</tr>
<tr>
<td><strong>D2</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Time (ms)</strong></td>
<td>4.4</td>
</tr>
</tbody>
</table>
We appreciate the fact that, as presented in Table 2, the Gabriel Graph rule overhead is almost neglectable (w.r.t. the classifier computational complexity). The reason of this behavior is the reduced number of samples in the knowledge base that is guaranteed by the usage, during the experiment, of the Gabriel Graph rule itself.

The great advantages provided by JIT-G approach in terms of computational complexity and memory requirements are also evident by considering Figures 2 and 3. These figures compare the average number of samples that compose the knowledge base and the average computational time during Application D1 (case $k=1$) for the JIT-G adaptive classifier and the traditional $k$-NN adaptive classifier. Results for application D2 and the other $k$ s are in line with what here presented.

![Figure 2 – App. D1 – Size of the knowledge base during the experiment (with $k=1$)](image1)

![Figure 3 – App. D1 – Computational time during the experiment (with $k=1$)](image2)

V. CONCLUSIONS

The paper addresses a novel approach to the development of just-in-time adaptive classifiers implementing complexity reduction techniques. The additional information is inserted in the knowledge base only if it provides a meaningful innovation content (otherwise it is discarded). This allows us for reducing the increase in complexity due to the fresh knowledge yet maintaining/improving the accuracy. Simulation results show the effectiveness of the proposed approach.

REFERENCES