Function \textit{Classify}:

\textbf{Input}: integer \(k\), data \(D\), instance \(x\).

\textbf{Output}: class set \(VS_k(D)(x)\) assigned to \(x\).

\begin{verbatim}
if \(k < 0\) then
  \quad \text{return } \emptyset;
\end{verbatim}

\begin{verbatim}
Y_{k-1} := Classify(k - 1, D, x);
if \(Y_{k-1} = \emptyset\) then
  \quad \text{if } \neg(\exists h \in H)\text{cons}_k(h, D) \text{ then}
  \quad \quad \text{return } \emptyset;
\end{verbatim}

\begin{verbatim}
Y_k := Y_{k-1};
\end{verbatim}

\begin{verbatim}
for each class \(y \in Y \setminus Y_{k-1}\) do
  \quad \text{if } (\exists h \in H)\text{cons}_k(h, D) \land \text{cons}_0(h, \{(x, y)\}) \text{ then}
  \quad \quad \text{Y}_k := Y_k \cup \{y\};
\end{verbatim}

\text{return} \(Y_k\).

Fig. 1. Classification function of \(k\)-version spaces based on the \(k\)-consistency tests

problem that consists of the \(k - 1\)-class-set problem and the general \(k0\)-consistency problem in the proposed order of computations.

So far we showed that the \(k\)-version-space classification problem consists of the \(k\)-collapse problem and the \(k\)-class-set problem. The \(k\)-collapse problem consists of the \(k - 1\)-collapse problem and the general \(k\)-consistency problem. The \(k\)-class-set problem consists of the \(k - 1\)-class-set problem and the general \(k0\)-consistency problem. Thus, since the \(k - 1\)-collapse problem and the \(k - 1\)-class-set problem form the \(k - 1\)-version-space classification problem, it follows that the \(k\)-version-space classification problem is a recursive problem that consists of the \(k - 1\)-version-space classification problem, the general \(k\)-consistency problem, and the general \(k0\)-consistency problem. This result implies that the \(k\)-version-space classification can be (tractably) implemented as soon as we can (tractably) test for \(k\)-consistency and \(k0\)-consistency in the given hypothesis space. The consistency tests can be applied to any class. Thus, we allow \(k\)-version spaces to be applied for multi-class classification tasks.

The classification function of \(k\)-version spaces based on the consistency tests is given in Figure 1. The function input includes data \(D\), integer \(k\), and instance \(x \in X\). The output is the \(k\)-class set \(VS_k(D)(x)\) for \(x\) provided according to Definition 3.

The function is recursive. It first checks whether \(k < 0\). If \(k < 0\), then by Corollary 2 the \(k\)-version space \(VS_k(D)\) is empty. This implies by Theorem 3 that the \(k\)-class set \(VS_k(D)(x)\) is empty. Thus, the function returns empty set.

If \(k \geq 0\), we note that the \(k\)-version-space classification problem includes the \(k - 1\)-version-space classification problem. Hence, the function calls itself recursively for \(k - 1\). The result of the call is the set \(Y_{k-1}\) of classes assigned by the \(k - 1\)-version space \(VS_{k-1}(D)\) to the instance \(x\). If the class set \(Y_{k-1}\) is empty, then by Theorem 3 the \(k - 1\)-version space \(VS_{k-1}(D)\) is empty. Thus, by Theorem 4 and Lemma 1 in order to decide whether the \(k\)-version space \(VS_k(D)\) is non-empty we solve the general \(k\)-consistency problem; i.e., we test whether there exists a scoring classifier in \(H\) that is
k-consistent with $D$. If the test is negative, by Definition 2 the $k$-version space $VS_k(D)$ is empty and by Definition 3 the function returns $\emptyset$. If the test is positive, by Definition 2 $VS_k(D)$ is non-empty. Therefore, the function continues the classification process by initializing the class set $Y_k$ (assigned to the instance $x$ by $VS_k(D)$). By Theorem 5 $Y_{k-1} \subseteq Y_k$. Thus, $Y_k$ is initialized equal to $Y_{k-1}$. Then the function tests whether the classes from $Y \setminus Y_{k-1}$ can be added to $Y_k$. We note that for each of these classes Lemma 2 holds. Thus, by Theorem 5 for each class $y \in Y \setminus Y_{k-1}$ we solve the general $k0$-consistency problem for the data $D$ and $\{(x, y)\}$. This is done by testing whether there exists a scoring classifier $h \in H$ that is $k$-consistent with $D$ and $0$-consistent with $\{(x, y)\}$. If so, then by Theorem 5 the class $y$ is added to the set $Y_k$. Once all the classes in $Y \setminus Y_{k-1}$ have been visited the class set $Y_k$ is outputted.

Let $T_k$ be the time complexity of the general $k$-consistency test and $T_{k0}$ be the time complexity of the general $k0$-consistency test. Assuming that $T_k < T_{k0}$ the worst-case time complexity of the classification function of $k$-version spaces equals:

$$O(T_k + k|Y|T_{k0}).$$

4 Consistency Algorithms

To implement the classification function of $k$-version spaces based on the consistency tests we need $k$-consistency-test algorithms and $k0$-consistency-test algorithms. Below we propose two approaches to implement these algorithms. The first one is for the case when there exists a $0$-consistent learning algorithm $l$ for the hypothesis space $H$. It allows designing consistency-test algorithms valid for the whole hypothesis space $H$. Hence, it is called hypothesis-unrestrictive approach. The second approach is for the case when there exists no $0$-consistent learning algorithm $l$ for the hypothesis space $H$. It allows designing consistency-test algorithms valid for a sub-space of the hypothesis space $H$. Hence, it is called hypothesis-restrictive approach.

4.1 Hypothesis-Unrestrictive Approach

The hypothesis-unrestrictive approach assumes that there exists a $0$-consistent learning algorithm $l$ for the hypothesis space $H$. Thus, $H$ contains hypothesis that is $0$-consistent with data $D$ iff $l$ succeeds; i.e., $l$ outputs for $D$ some hypothesis (that by definition is consistent with $D$). This implies that the $0$-consistency-test algorithm in this case is the $0$-consistent learning algorithm $l$ plus a success test. In the past (cf. [5]) $0$-consistency-test algorithms were proposed for different hypothesis spaces such as $1$-decision lists, monotone depth two formulas, halfspaces etc. They guarantee tractable $0$-version-space classification, if they are tractable.

By Definition 1 if we can test for $0$-consistency, we can test for $k$-consistency. Thus, given data $D$ and integer $k$, we design a $k$-consistency-test algorithm as follows. We start with $m = 0$ and then for each $D_m \subseteq D$ with size $|D|-m$ we apply a $0$-consistency-test algorithm. If the $0$-consistency-test algorithm identifies $0$-consistency for at least one $D_m$, by Theorem 1 there is $0$-consistency for some $D_k \subseteq D_m$ and we return value “true”. Otherwise, we continue with the next $D_m$ or increment $m$ in the boundary of $k$. If this is not possible, we return value “false”. Thus, the worst-case time complexity