Project Machine Learning

Effect of feature weighting in Fuzzy-Rough nearest neighbour classification algorithms

Victor Miclotte and Bram Verbeke

December 2017

Fuzzy-Rough set theory

- Similarity relation
- Fuzzy Quantifiers
- Classifiers

Feature Weights

- Calculating weights

Experiments

- Impact of K
- Performance comparison

Last Presentation

- Introduction to Fuzzy-Rough nearest neighbour classification
- Proposed several weighting methods

Last Presentation

Fuzzy set theory

Remember a fuzzy set is a mapping $A : X \to [0, 1]$.

For an element $x \in X$, $A(x)$ would represent the membership degree of x to the fuzzy set A.

Last Presentation

Rough set theory

Recall the problem of incomplete data.

Two objects could be indiscernible according to all features, but appeared to be different in class label.

If A is the set of all features, we can associate, with any subset $B \subseteq A$ a relation

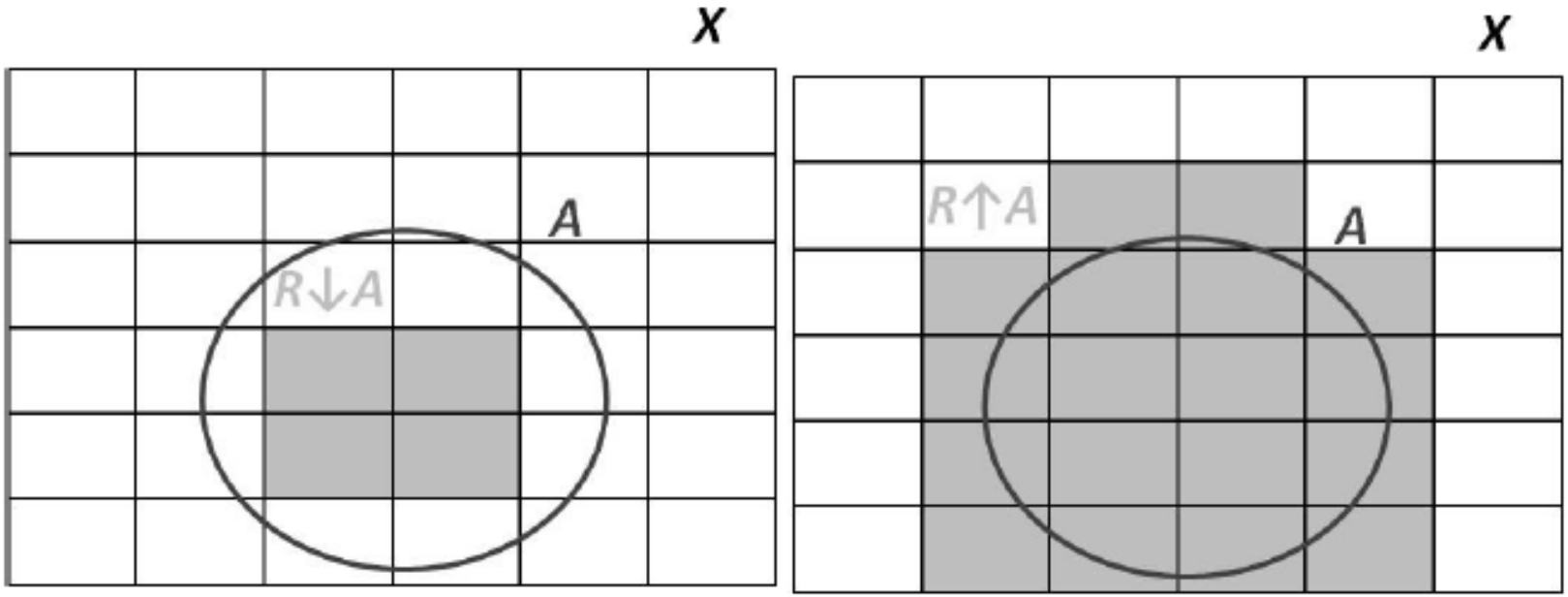
$$R_B = \{(x, y) \in X^2 | \forall a \in B : a(x) = a(y)\}$$

This is an equivalence relation. So we could partition our universe X into these equivalence classes $[x]_B$.

Last Presentation

Rough set theory

Remember we could approximate a decision class A by a lower and upper approximation.



$$R_B \downarrow A = \{x \in X | [x]_B \subseteq A\}$$
$$R_B \uparrow A = \{x \in X | [x]_B \cap A \neq \emptyset\}$$

Last Presentation

But now we can fuzzify this relation R_B as well

$$R_B = \{(x, y) \in X^2 | \forall a \in B : a(x) = a(y)\}$$

This would result in a fuzzy relation R where

$$R : X^2 \to [0, 1]$$

where $R(x, y)$ denotes to what degree x is equal to y. We call this a similarity relation.

Frame Title

There are many examples of similarity relations, in this project we used

$$R(x, y) = 1 - \frac{||x - y||}{d_{max}}$$

where $d_{max} = \max_{x,y \in X} ||x - y||$. Here $||x - y||$ was defined as the L^2-norm

$$||x - y|| = \sqrt{\sum_{a \in A} (a(x) - a(y))^2}$$

where we could insert our weight vector like this

$$||x - y|| = \sqrt{\sum_{a \in A} (w(a) \cdot (a(x) - a(y)))^2}$$

Fuzzy Quantifiers

To construct these approximations we used the Vaguely Quantified Rough Set model (VQRS).
Here is an example of a fuzzy quantifier

$$Q_{(\alpha,\beta)}(x) = \begin{cases} 0, & x \leq \alpha \\ \frac{2(x-\alpha)^2}{(\beta-\alpha)^2}, & \alpha \leq x \leq \frac{\alpha+\beta}{2} \\ 1 - \frac{2(x-\beta)^2}{(\beta-\alpha)^2}, & \frac{\alpha+\beta}{2} \leq x \leq \alpha \\ 1, & \beta \leq x \end{cases}$$

with parameters α and β.

Lower and upper approximation

$$(R \downarrow A)(x) = Q_l \left(\frac{\sum_{y \in X} \min(R(x,y), A(y))}{\sum_{y \in X} \min R(x,y)} \right)$$

$$(R \uparrow A)(x) = Q_u \left(\frac{\sum_{y \in X} \min(R(x,y), A(y))}{\sum_{y \in X} \min R(x,y)} \right)$$

where $Q_l = Q_{(0.1,0.6)}$ and $Q_u = Q_{(0.2,1)}$.

AverageClassifier

Input: X: training data, C: set of decision classes, y: the object to be classified
Output: Classlabel for y
$N \leftarrow getNearestNeighbours(y, K)$
$\tau \leftarrow 0, Class \leftarrow \emptyset$
for all $C \in \mathbb{C}$ **do**
  if $((R \downarrow C)(y) + (R \uparrow C)(y))/2 \geq \tau$ **then**
    Class $\leftarrow C$
    $\tau \leftarrow ((R \downarrow C)(y) + (R \uparrow C)(y))/2$
  end if
end for
output *Class*

LowClassifier

Input: X: training data, C: set of decision classes, y: the object to be classified
Output: Classlabel for y
$N \leftarrow getNearestNeighbours(y, K)$
$\tau \leftarrow 0, Class \leftarrow \emptyset$
for all $C \in \mathbb{C}$ **do**
  if $((R \downarrow C)(y) + (R \uparrow C)(y))/2 \geq \tau$ **then**
    Class $\leftarrow C$
    $\tau \leftarrow ((R \downarrow C)(y) + (R \uparrow C)(y))/2$
  end if
end for

Input: X: training data, C: set of decision classes, y: the object to be classified
Output: Classlabel for y
$N \leftarrow getNearestNeighbours(y, K)$ (similarity relation R)
$\tau \leftarrow 0, Class \leftarrow \emptyset$
for all $C \in \mathbb{C}$ **do**
  if $(R \downarrow C)(y) \geq \tau$ **then**
    Class $\leftarrow C$
    $\tau \leftarrow (R \downarrow C)(y)$
  end if
end for

UpClassifier

Input: X: training data, C: set of decision classes, y: the object to be classified
Output: Classlabel for y
$N \leftarrow getNearestNeighbours(y, K)$
$\tau \leftarrow 0, Class \leftarrow \emptyset$
for all $C \in \mathbb{C}$ **do**
  if $(R \uparrow C)(y) \geq \tau$ **then**
    Class $\leftarrow C$
    $\tau \leftarrow (R \uparrow C)(y)$
  end if
end for
output *Class*

Calculating weights: Entropy-based

1. Entropy $H(X) := E\left[-\log(P(X))\right] = -\sum_x P(x)\log P(x)$
2. Information gain

$$IG(C, A) := H(C) - H(C|A)$$

3. Information gain ratio $IGR(C, A) := \dfrac{H(C) - H(C|A)}{H(A)}$
4. Symmetrical uncertainty coefficient

$$SymmU(C, A) := \frac{H(C)U(C|A) + H(A)U(A|C)}{H(C) + H(A)}$$

Calculating weights: Kullback-Leibler

1. Kullback-Leibler measure $\mathcal{KL}(C|a_{ij}) = \sum_c P(c|a_{ij}) \log \frac{P(c|a_{ij})}{P(c)}$
2. Weight of feature i, $wavg(i)$, is weighted average of the KL measures across the feature values.
3. $wavg(i) = \sum_j \frac{\#(a_{ij})}{N} \mathcal{KL}(C|a_{ij})$
4. $\#(a_{ij})$ is the number of instances that have value a_{ij} for feature i
5. N is the total number of training instances

Calculating weights: (Multiclass) Support Vector Machines

Binary-class:

1. Build a support vector machine
2. Use the weights from the SVM as feature weights

Multi-class:

1. Build pairwise SVM's (aka 1-vs-1)
2. Compute the average of weights across all SVM's for each feature
3. Use these averages as feature weights

Calculating weights: Relief feature selection

A feature selection algorithm that computes weights for every feature. Originally developed for binary-class data.

Algorithm 4 Relief

Input: X: data, $\mathcal{A}$: set of N attributes, $\mathcal{C}$: binary set of decision classes, M: number of samples, $w = (0, \ldots, 0)$: N-dimensional vector
Output: Weight vector $w = (w_{attr1}, \ldots, w_{attrN})$
for $m = 1 : M$ **do**
  randomly select $x \in X$
  find $NH(x)$ and $MH(x)$
  for all $A \in \mathcal{A}$ **do**
    $w_A =$
    $w_A - |A(x) - A(NH(x))| + |A(x) - A(NM(x))|$
  end for
end for
output Weight vector w

Calculating weights: Relief feature selection

1. $NH(x)$ is the "nearest hit" of x, the nearest instance with same class label
2. $MH(x)$ is the "nearest miss" of x, the nearest instance with a different class label
3. Multi-class: using `RELIEFF`, consider multiple near misses and average their contribution when calculating weights

Calculating weights: OneR

A simple classification algorithm that finds the most relevant attribute and builds a classification rule based on just this one attribute.

Algorithm 4 OneR

Input: X: training data, $\mathcal{A}$: set of attributes, $\mathcal{C}$: set of decision classes
Output: Optimal attribute and its rule
for all Attribute $A \in \mathcal{A}$ **do**
  create a rule R_A
  for all Value $a \in A$ **do**
    find the most frequent class C
    add rule to R_A: `if` $A = a$ `then` $\mathcal{C} = C$
  end for
  calculate the error rate of R_A
end for
output $A \in \mathcal{A}$ with lowest error rate for its rule R_A, R_A

Calculating weights: OneR

1. For every attribute A a rule R_A is created and an error rate err_A is computed.
2. We use $1 - err_A$ as weight for attribute A.
3. A low err_A indicates that A is rather relevant and it will get a higher weight.

Preprocessing: discretization

1. Calculation of probabilities $P(A = a), P(C = c|A = a)$
2. A certain value will rarely occur multiple times, when the attribute is continuous
3. We discretized every attribute using k-means with $k = numClasses$.
4. Discretization did not improve the performance of any of the algorithms
5. We decided to work with raw data

Experiments

Table: Datasets

Data set	number of features	number of instances	type
Appendicitis	7	106	numeric
Glass	9	214	numeric
Heart	13	294	numeric
Iris	4	150	numeric
Water2	38	521	numeric
Wine	13	178	numeric

Experiments

For every dataset, we made 10 folds to conduct cross-validation.

These folds were stored in separate files so we would experiment on the same folds every time.

In the following figures

- LowClassifier is pointed in black
- AverageClassifier in green
- and UpClassifier in blue.

Impact of K

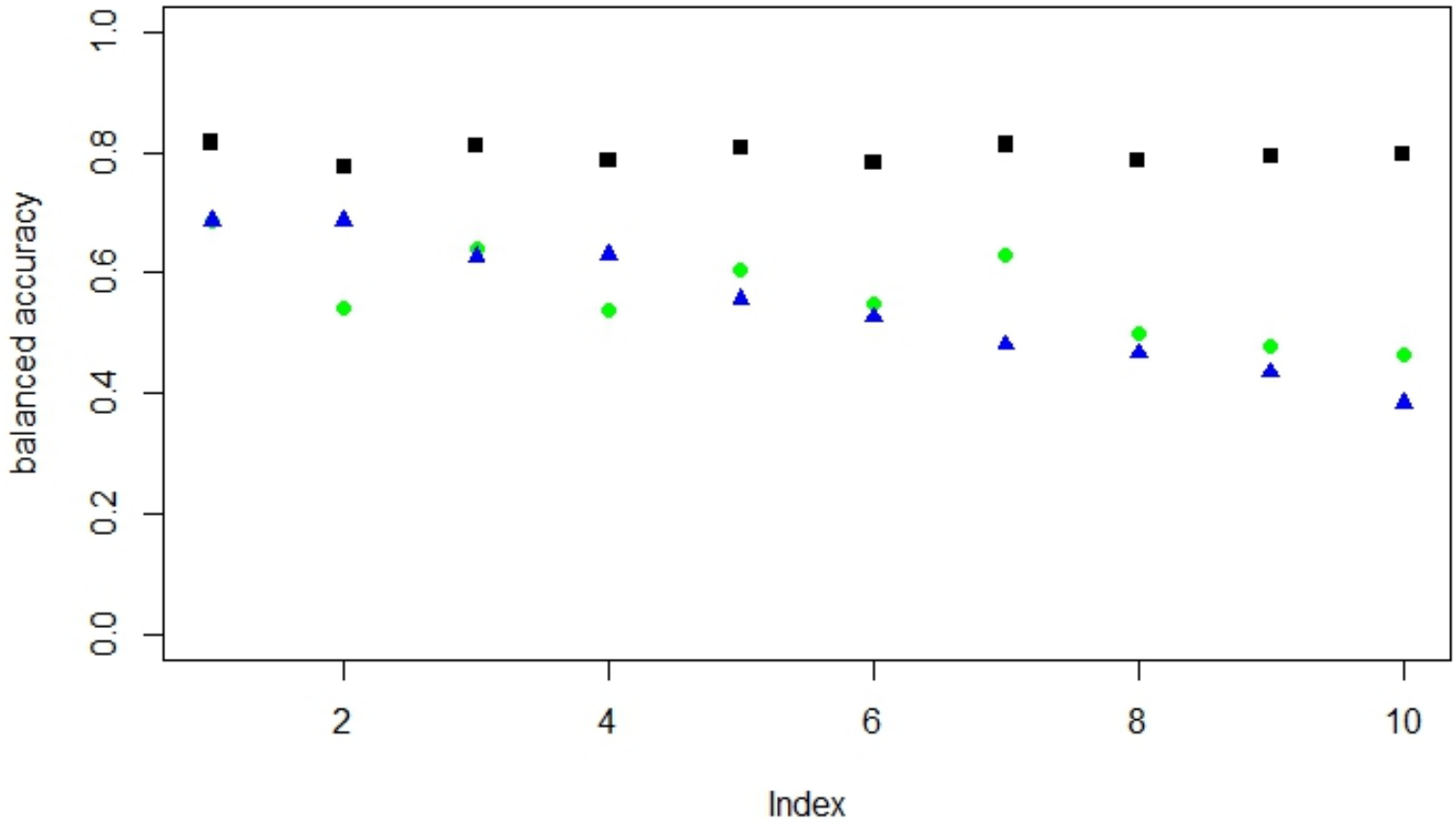


Figure: Balanced accuracy, dataset: Water2

Impact of K

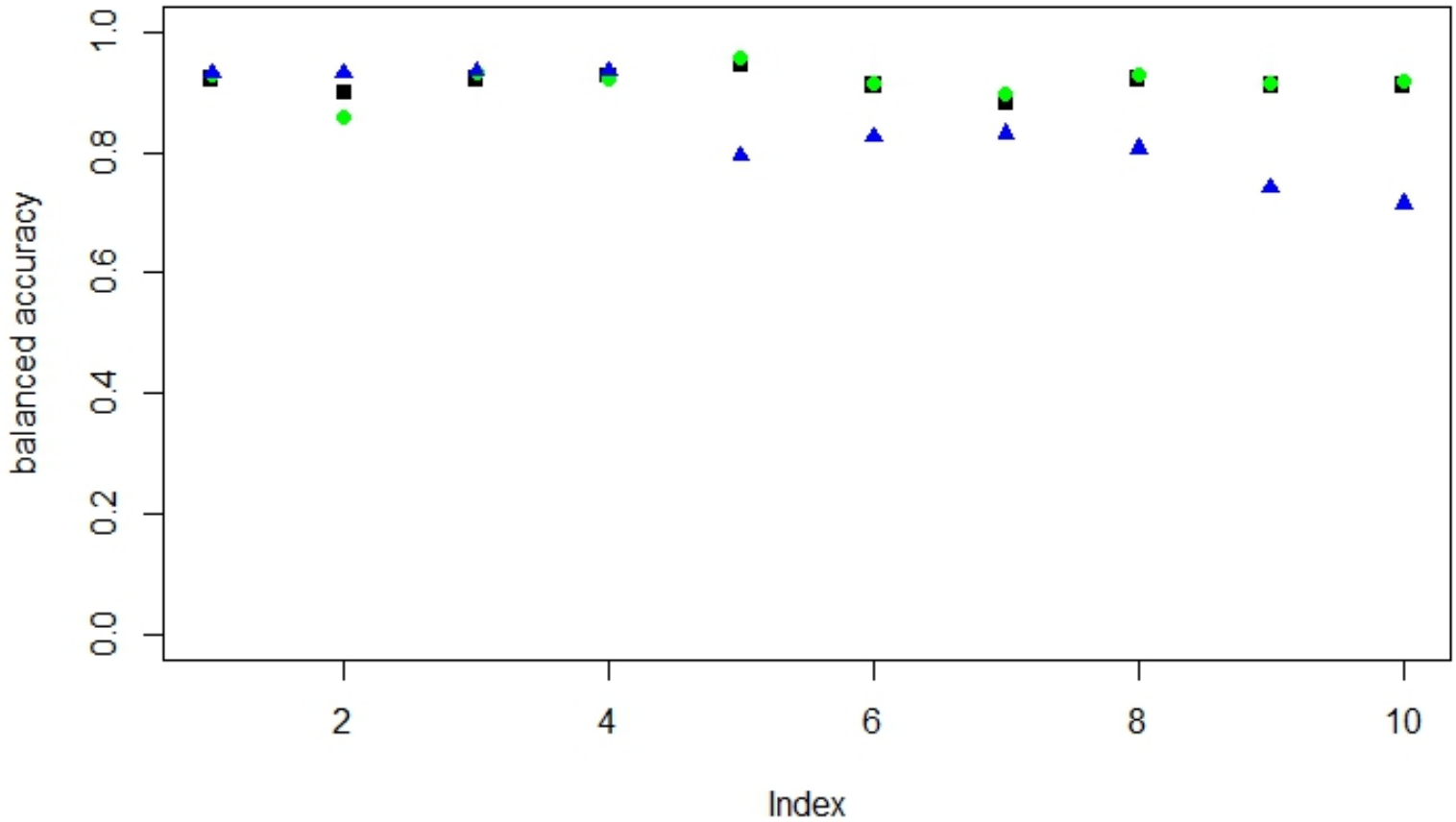


Figure: Balanced accuracy, dataset: Wine

Impact of K

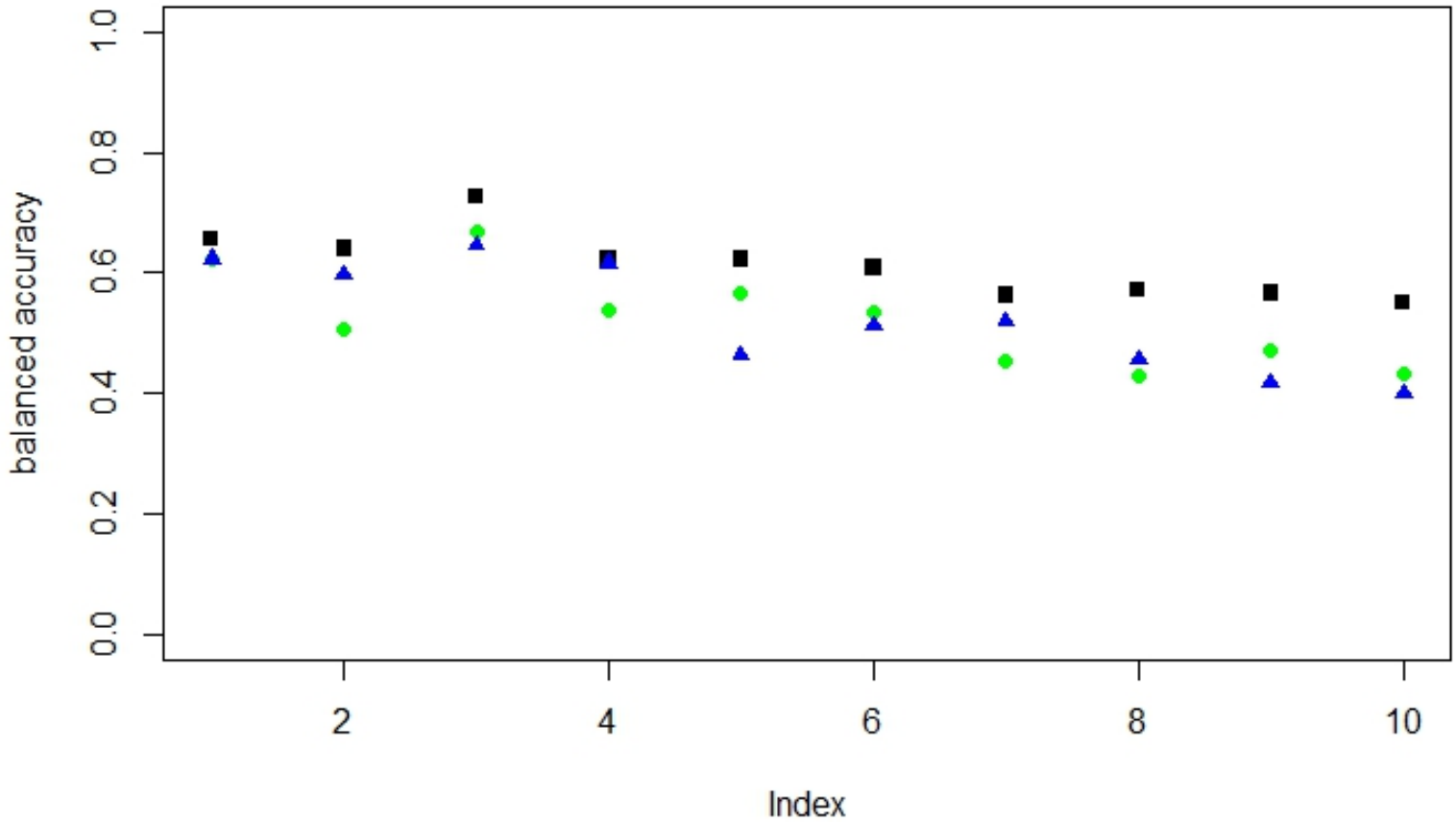


Figure: Balanced accuracy, dataset: Glass

From now on we will compare the performance of our weighting methods with a fixed number of neighbours $K = 3$.

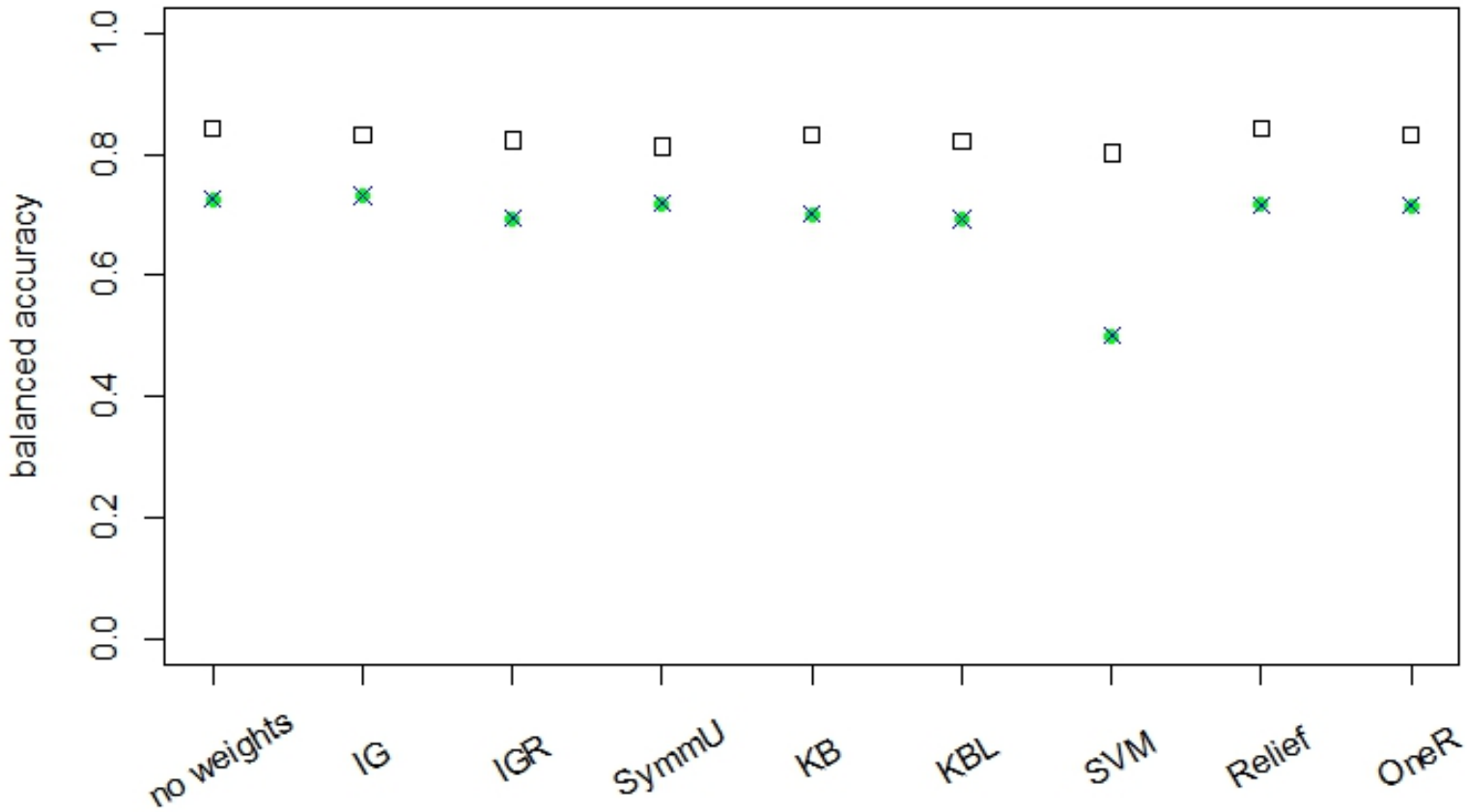


Figure: Balanced accuracy, dataset: Appendicitis

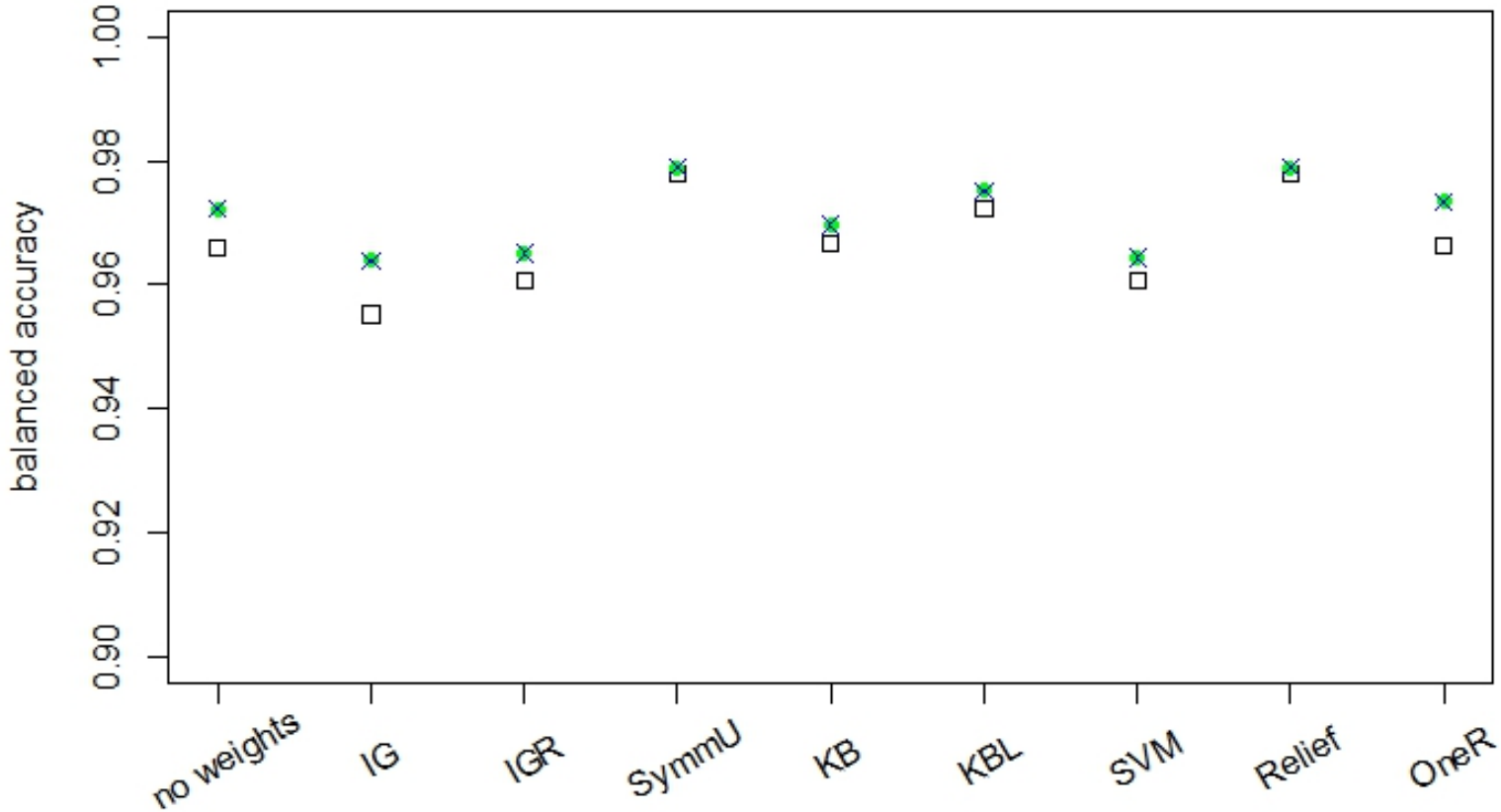


Figure: Balanced accuracy, dataset: Wine

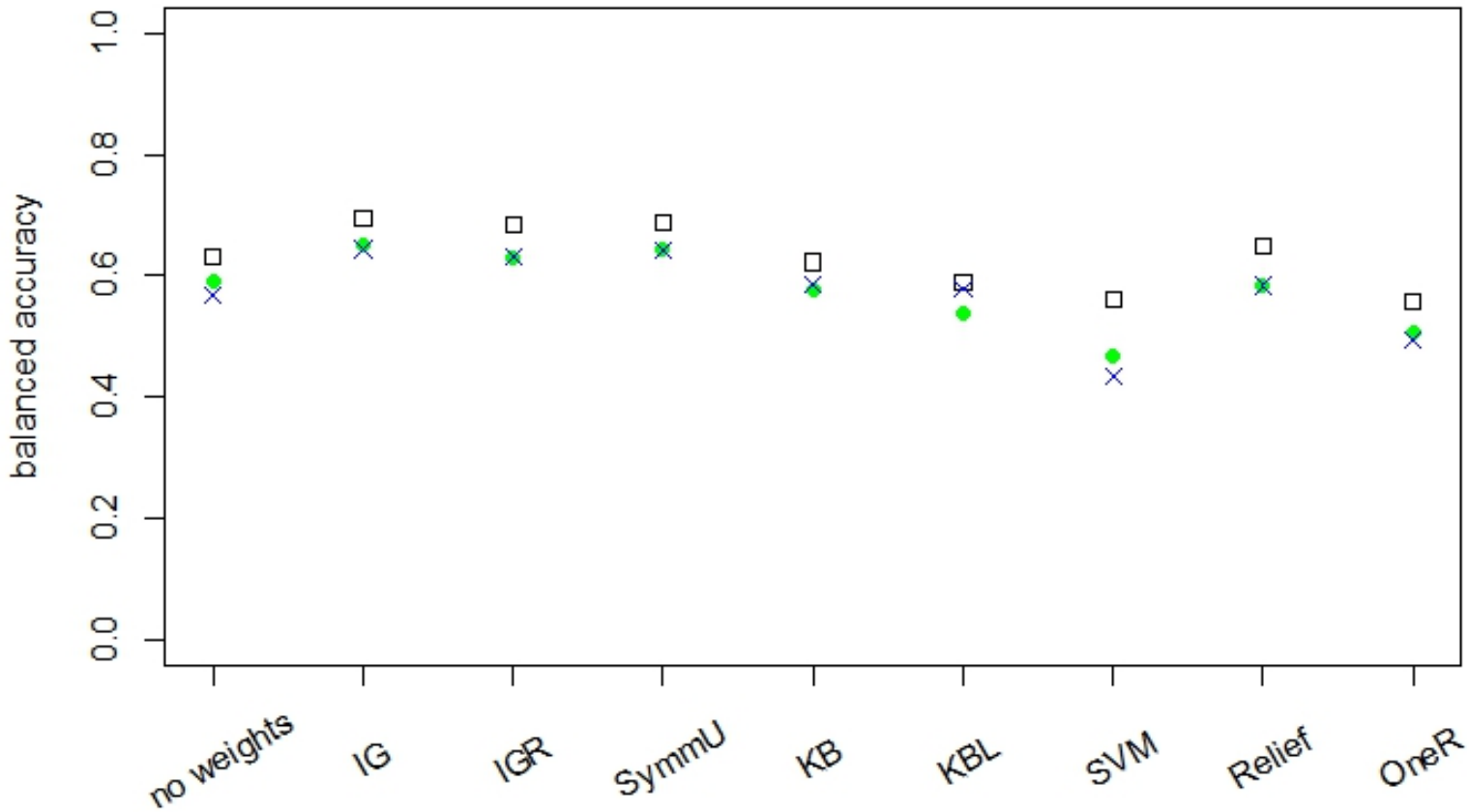


Figure: Balanced accuracy, dataset: Glass

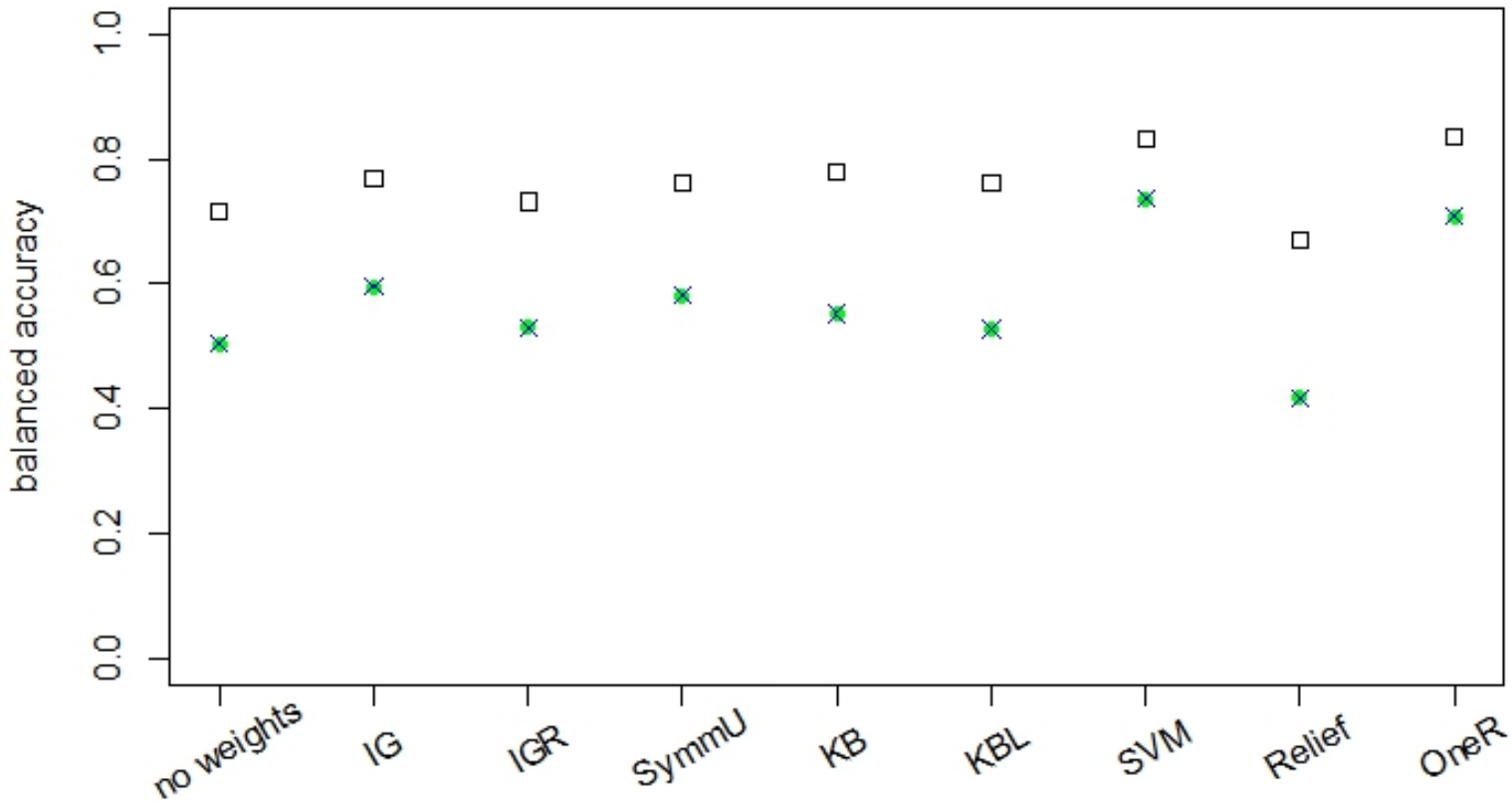


Figure: Balanced accuracy, dataset: Water2

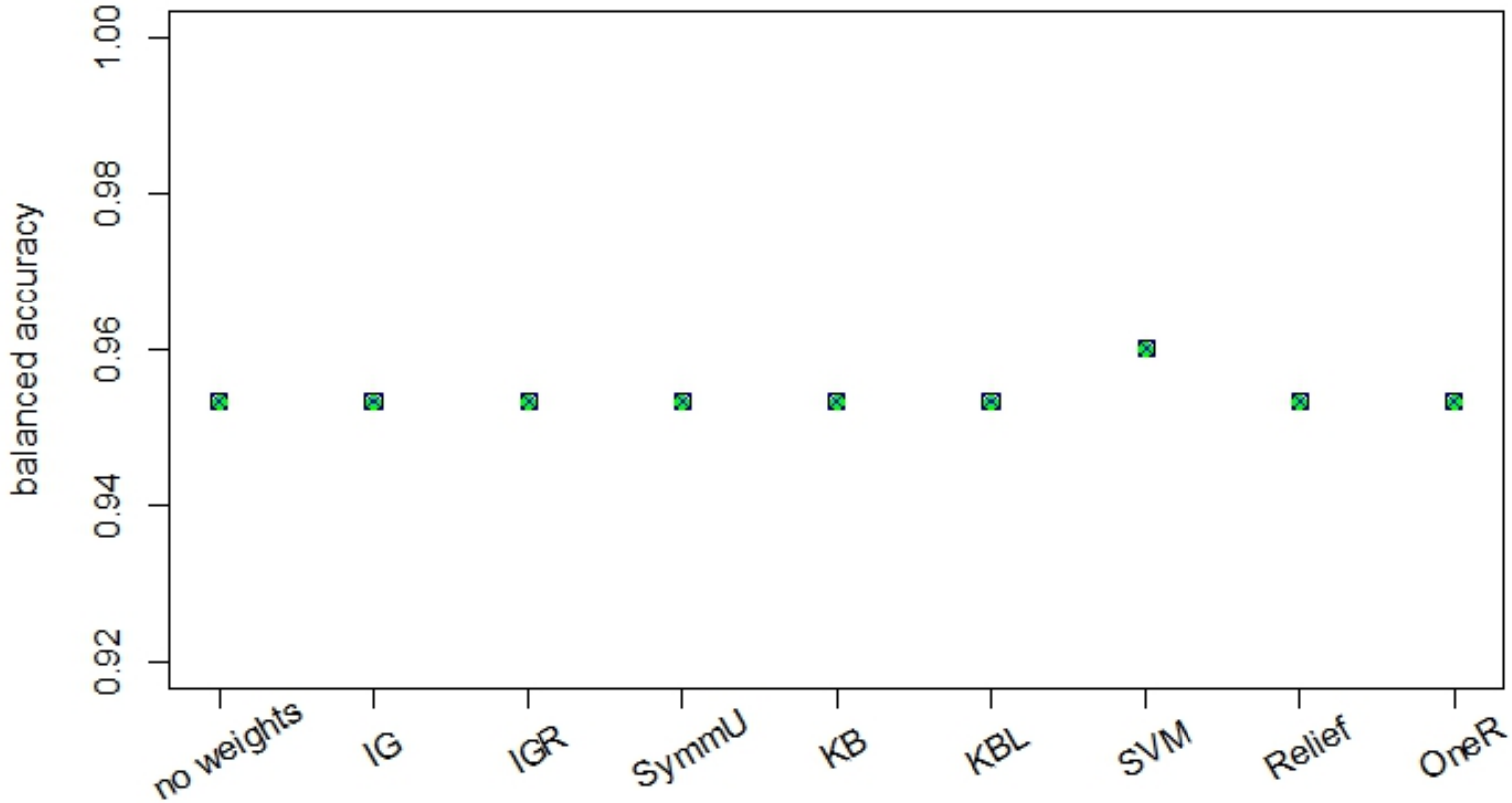


Figure: Balanced accuracy, dataset: Iris

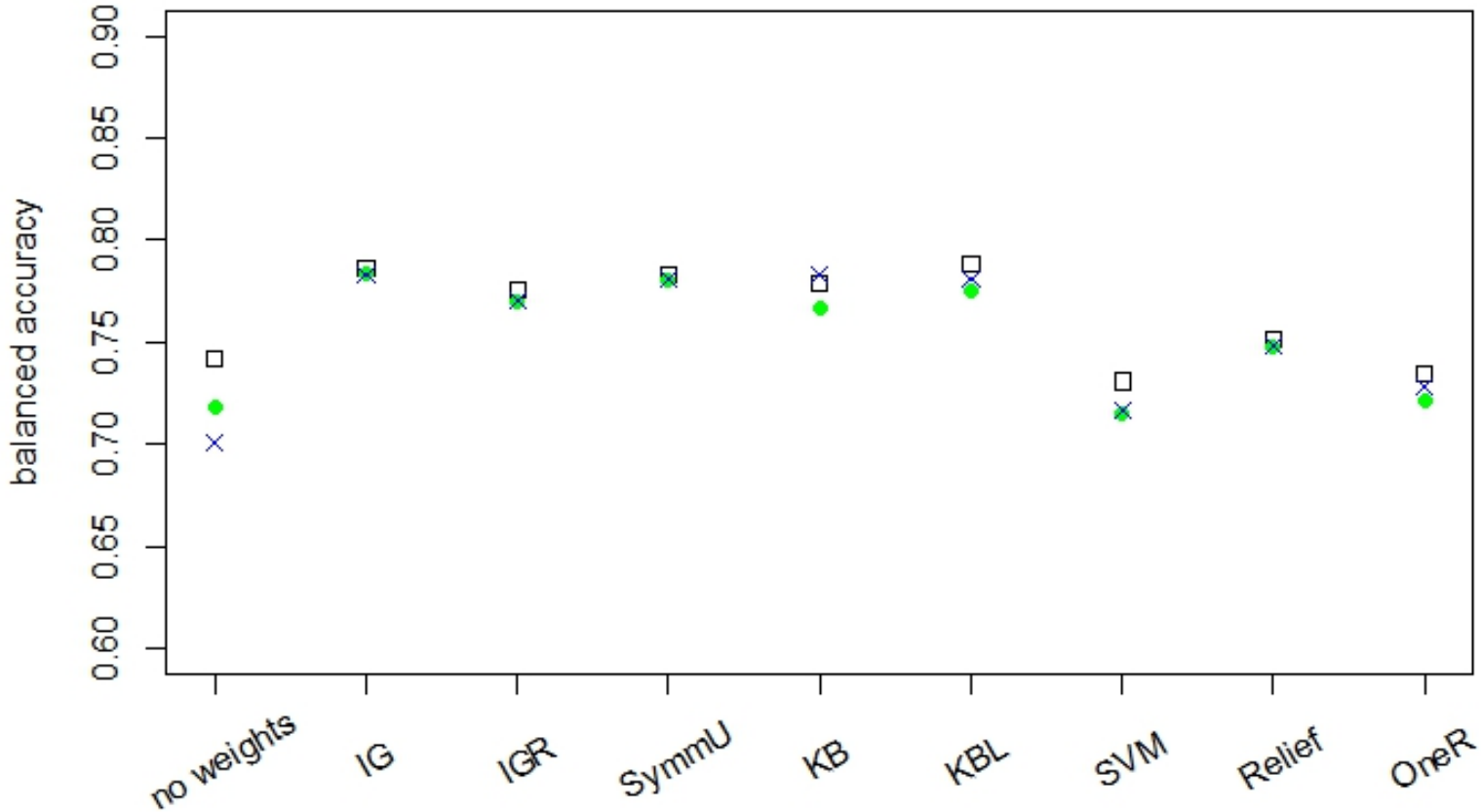


Figure: Balanced accuracy, dataset: Heart

Conclusion

1. Feature weighting has the potential to improve Fuzzy-Rough nearest neighbour classification algorithms
2. Multiple feature weighting methods have been tested but none of them consistently improve the classification algorithms nor make them perform worse
3. Consider using different feature weighting methods to optimize the performance of Fuzzy-Rough nearest neighbour classification algorithms