Handling Class Imbalance in k-Nearest Neighbor Classification by Balancing Prior Probabilities

Jonatan Møller Nuutinen Gøttcke[1][0000−0003−4104−0298] and Arthur Zimek[1][0000−0001−7713−4208]

Institute of Mathematics and Computer Science,
University of Southern Denmark, Odense, Denmark
{goettcke,zimek}@imada.sdu.dk

Abstract. It is well known that recall rather than precision is the performance measure to optimize in imbalanced classification problems, yet most existing methods that adjust for class imbalance do not particularly address the optimization of recall. Here we propose an elegant and straightforward variation of the *k*-nearest neighbor classifier to balance imbalanced classification problems internally in a probabilistic interpretation and show how this relates to the optimization of the recall. We evaluate this novel method against popular *k*-nearest neighbor-based class imbalance handling algorithms and compare them to general oversampling and undersampling techniques. We demonstrate that the performance of the proposed method is on par with SMOTE yet our method is much simpler and outperforms several competitors over a large selection of real-world and synthetic datasets and parameter choices while having the same complexity as the regular *k*-nearest neighbor classifier.

Keywords: Class imbalance · Bayesian learning · k-nearest neighbor classification

1 Introduction

In classification problems, skewed class distributions often result in poor accuracy when predicting instances of the minority classes. The problem is common and found in substantially different areas such as fraud detection, propaganda detection, and medical diagnosis. The typical class imbalance problem is often presented as a dichotomous classification problem, where it is crucial that the minority class is predicted correctly. A common example would be data about a rare disease, where the available training data contains many instances without the disease, and only few with the disease. The majority class dominates the training and potentially also the evaluation, if done naively. The imbalance ratio, IR, captures the severity of a problem by the ratio of the size of the (largest) majority class (c_{maj}) over the size of the (smallest) minority class (c_{min}):

$$IR = \frac{|c_{maj}|}{|c_{min}|} \tag{1}$$

Of course, the problem can vary in complexity by having multiple minority classes with different degrees of importance for each of the minority classes. The problem also becomes more difficult when the imbalance ratio (IR) is increased. In the literature, a value of $IR > 3.5$ is seen as signalling a high degree of imbalance in a dataset [26].

Besides the absolute value of IR, we can also distinguish *absolute* imbalance and *relative* imbalance. Absolute imbalance describes the case where there is a small absolute amount of minority instances. For example, if there are 5 instances in the minority class and 95 in the majority class, we get an IR of 19. Relative imbalance simply means that the IR is large, e.g., if there are 5000 instances in the minority class and 95000 in the majority. This would also result in an IR of 19. However, a re-sampling strategy should most likely be different for these two cases. Absolute and relative imbalance has been discussed in more detail by Bellinger et al. [3].

When studying the performance of algorithms on imbalanced classification problems, it is a fallacy to just report the accuracy, error rate, precision, or f1 measure. The ROC curve, although inherently accounting for imbalance, comes with its own problems as well [8, 13].

The *precision* measure reports, for one class against all other classes, the number of true positives (TP) divided by the false positives (FP) plus the true positives, i.e., $\frac{TP}{FP+TP}$. In class imbalanced problems, precision is not a viable measure since the majority classes will have relatively few false positives no matter how many of the minority points they predict as majority points. Examples for the minority classes on the other hand are rarely mistaken for majority points. The true positives will thus typically be divided by almost the same number, because there are no or few false positives, which leads to a high precision. The harmonic mean between precision and recall (f1) is also reported in some studies [10], but since precision is not a meaningful measure in class imbalanced problems its presence in the f1 measure only hides the algorithms' performance in terms of recall.

Some argue [2, 16] that accuracy and error rate are strongly biased to favor the majority class. The problem with accuracy and error rate is obvious when the class imbalances are extreme. If in a binary classification problem 99.9% belongs to the minority class, and only 0.01% belongs to the minority, then if we completely fail to predict the minority class the classifier still has an accuracy of 99.9%. Thus the G-mean score is a popular measure [4, 16, 18], that is the geometric mean over recall: $(\prod_{i=1}^{n} r_i)^{\frac{1}{n}} = \sqrt[n]{r_1 \cdot r_2 \cdot \ldots \cdot r_n}$, where r_i is the recall for class c_i.

This results in a quality measure that heavily penalizes a low recall for any of the classes, reflecting the algorithm's inability to hypothesize that point x belongs to the minority class. Failing completely on one class results in a zero value for the overall G-mean score. We therefore argue for the sensibility in reporting the macro-averaged recall in addition to the G-mean score. For further discussion on performance assessment of imbalanced classification problems see the work by Japkowicz [16].

Considering the appropriateness of recall when working with imbalanced classification problems, in this paper we introduce a variant of the k-nearest neighbor classifier that balances the prior class probabilities. We show how this effectively resembles using a local recall as a classification rule, while the standard k-nearest neighbor classifier effectively uses the local precision as a decision criterion.

The remainder of the paper is organized as follows. In Section 2, we discuss related work. In Section 3 we introduce our method and discuss some properties. In Section 4 we perform an experimental evaluation of our method against state-of-the-art competitors. In Section 5 we summarize and give perspectives for future work.

2 Related work

There exist many approaches to handling imbalanced classification problems. The approaches can be divided into three main categories, namely external, internal, and cost-sensitive approaches.

2.1 External approaches

External approaches alter the dataset a classifier is trained on. The alterations are typically different re-sampling techniques such as majority undersampling, minority oversampling, or a combination of both. In oversampling, the minority class domain is extended by adding real points or synthetic points to the existing training data. The simplest such approaches are Random Undersampling (RUS) [15, 25] which randomly removes points from the majority class until a uniform class distribution is reached. A similar oversampling method exists, namely random oversampling (ROS) [15], which picks random real points from the minority classes and oversamples these until a uniform class distribution is reached.

SMOTE [6] is likely the most popular oversampling technique. It adds additional data points to the dataset by inserting new synthetic samples within the convex hull of the minority class. The synthetic sample is positioned on the straight line between two minority points, i.e., $a + (b - a) \cdot \alpha$, where a is the feature vector of the point under consideration, b is a feature vector of a random instance of the nearest neighbors and $\alpha \in [0, 1]$ is a random value determining where on the line the point should be positioned. The algorithm iterates through all data points of the minority class and oversamples each point by finding the k nearest neighbors and picking n randomly, where n is the oversample rate. A data point is inserted at a random point on each line between these n sets of two. Chawla and Nitesh demonstrated that SMOTE improved performance over random oversampling and that SMOTE results in reduced decision tree sizes, when used in combination with C4.5 [24]. A plethora of slightly modified SMOTE variations have been developed since the original. Noteworthy mentions are Borderline SMOTE [12] and ADASYN [14].

2.2 Internal approaches and modifications of the kNN classifier

Internal approaches modify an existing classifier to account for the class imbalance. Over the past two decades, several attempts have been proposed to modify specifically the *k*NN classifier to account for an imbalanced class distribution. Song et al. [27] introduced IkNN, where they employ information about the distance from a query point to its nearest neighbors to determine the most informative nearest neighbors. Kriminger et al. [17] created a class imbalance handling *k*NN variation which also works on imbalanced data streams. Liu and Chawla [20] introduce a class confidence weighted *k*NN rule by employing Gaussian mixture models, and Bayesian networks to estimate class weights.

Dubey and Pudi [10] claim to improve performance over these previous internal modifications. They modify the existing *k*NN algorithm to be sensitive to class imbalance by observing the class distributions within the *k*NN hyperballs of a subset of the neighbors' neighbors. This information is used to determine if the query point is a local minority given this new sense of locality and its new prior probability distribution.

2.3 Cost-sensitive learning and k-nearest neighbors

A way of making an arbitrary classifier sensitive to the class imbalance problem without modifying the dataset as done in the external approaches, is by employing *cost-sensitive learning*. In cost-sensitive learning each possible prediction is associated with some misclassification cost. The goal is then to minimize the total misclassification cost over the test dataset. Elkan [11] generalized cost-sensitive learning to the goal of minimizing the conditional risk as:

$$R(x, c_i) = \sum_j \Pr(c_j|x) C(i, j) \tag{2}$$

where x is an example, c_i and c_j class labels, and the $C(i, j)$ entry in the cost matrix is the cost of predicting class c_j when the true class is c_i. Picking the prediction that minimizes the conditional risk leads to decisions that are not necessarily the most probable outcome. Improving the sensitivity towards the minority class in cost sensitive learning can be achieved by increasing the cost of misclassifying minority instances. Domingos [9] proposed a method which can make any classifier cost sensitive, by employing ensemble learning. Qin et al. [23] and Zhang [28] proposed a cost-sensitive *k*NN classifier based directly on Elkan's formulation of conditional risk. The schema for their *Direct-CS-kNN* classifier is given by

$$\mathcal{L}(x, c_i) = \sum_{c_j \in C} \Pr(c_j|x) C(i, j) \tag{3}$$

The conditional risk is described as a loss function $\mathcal{L}(x, c_i)$, describing the loss of predicting class c_i, given query point x:

$$h(x) = \arg\min_{c_i \in C} \mathcal{L}(x, c_i) \tag{4}$$

If we alter Equation 3 to sum over the probabilities that it is not class c_j multiplied by a cost of predicting class c_j instead of predicting class c_i, the weights can become more understandable and we obtain some nice properties with respect to the minimization:

$$h(x) = \arg\min_{c_i \in C} \sum_{c_j \in C} 1 - \frac{\Pr(c_j|x)}{\sum_{c \in C} 1 - \Pr(c_c|x)} \cdot C(i,j) \tag{5}$$

The modification ensures several nice properties such as, if the cost matrix is equal to the identity matrix $C = I$, then we get the conventional majority vote kNN decision rule. If we use a diagonal matrix $\mathcal{D}$ with positive weights greater than or equal to 1, then if we pick $w_i = \frac{1}{D(i,i)}$ we obtain the basic weighted kNN decision rule.

2.4 Summary

In summary, although various methods exist to adjust classifiers in general or the kNN classifier in particular to imbalanced classification problems, none of these methods tackles the particular problem of considering the recall as an important objective of imbalanced classification. In the following, we introduce an elegant and straightforward way to do so.

3 Class-balanced k-nearest neighbors classification

The kNN classifier is an instance-based learning method that classifies an instance x from the input space by applying a decision rule to the set of k-nearest neighbors of x in the training data space. The decision rule is conventionally the majority vote, but could also be a weighted majority vote to handle a difference in the importance of attributes or to give higher weight to closer neighbors. As we have seen above, weights can also be associated with different class labels as an approach to handle the class imbalance problem [10].

3.1 Basic weighted kNN

The most intuitive approach to handling the class imbalance problem is perhaps to add importance to instances belonging to the minority classes. This can be done with a modification of the decision rule by multiplying the observed number of minority instances with a positive weight greater than 1, or the majority instances with a weight between 0 and 1, or a combination of both. To ensure the *relative importance* of each class is uniform despite accounting for different proportions of the dataset, the weight could be defined as:

$$w_i = \frac{|\{x|x \in c_{maj}\}|}{|\{x|x \in c_i\}|} \tag{6}$$

where c_{maj} is the majority class. This weighting scheme was generally proposed by Japkowicz [15].

3.2 Balancing a probabilistic k-nearest neighbor classifier

Choosing the class of the majority among the k nearest neighbors is from the point of view of probabilistic learning equivalent to choosing the maximum a posteriori (MAP) hypothesis when estimating the class probabilities for the different classes $c_i \in C$, given the query instance:

$$h_{\mathrm{MAP}}(x) = \arg\max_{c_i \in C} \Pr(c_i|x) \tag{7}$$

where we can find $\Pr(c_i|x)$ by Bayes' rule as:

$$\Pr(c_i|x) = \frac{\Pr(x|c_i) \cdot \Pr(c_i)}{\sum_{j=1}^{m} \Pr(x|c_j) \cdot \Pr(c_j)} \tag{8}$$

From this it is obvious that the prior class probabilities have some influence on the decision rule, and the core idea for our method is to not estimate these prior probabilities from the training sample but to define them as required by fairness. Intuitively, balancing an imbalanced classification problem means in this perspective to require uniform prior class probabilities, i.e., the decision of the classifier (Eq. 7) for m classes should use

$$\Pr(c_i|x) = \frac{\Pr(x|c_i) \cdot \frac{1}{m}}{\sum_{j=1}^{m} \Pr(x|c_j) \cdot \frac{1}{m}} \tag{9}$$

in order to treat all classes fair in a balanced way.

Interestingly, this addresses, locally, the need for optimizing the recall instead of the precision, as the decision rule turns out to be choosing the class that is captured to the largest proportion among the k nearest neighbors:

Theorem 1. *Given some query object x in a classification problem with a set C of m classes, let k_i be the number of instances among the k nearest neighbors of x that belong to class c_i, let n_i be the number of instances that belong to class c_i overall (i.e., $n_i = |c_i|$). For the k nearest neighbor classifier, adjusting the prior class probabilities such that all classes are equally likely, i.e., $\forall_i \Pr(c_i) = \frac{1}{m}$, is equivalent to choosing $\arg\max_{c_i \in C} \left(\frac{k_i}{n_i}\right)$, which is the local recall for x.*

Proof. The proxy for the probability $\Pr(x|c_i)$ is the density estimation given by the k nearest neighbors, conditional on class c_i, that we can describe as

$$\Pr(x|c_i) \propto \frac{k_i}{n_i V(x)} \tag{10}$$

where $V(x)$ is the volume, centered at x, required to capture k nearest neighbors of x. We can therefore rewrite Equation 8 as follows:

$$\Pr(c_i|x) \propto \frac{\frac{k_i}{n_i V(x)} \cdot \Pr(c_i)}{\sum_{j=1}^{m} \frac{k_j}{n_j V(x)} \cdot \Pr(c_j)} \tag{11}$$

Choosing equal prior class probabilities results in:

$$\Pr(c_i|x) \propto \frac{\frac{k_i}{n_i V(x)} \cdot \frac{1}{m}}{\sum_{j=1}^{m} \frac{k_j}{n_j V(x)} \cdot \frac{1}{m}} \tag{12}$$

which simplifies to

$$\Pr(c_i|x) \propto \frac{\frac{k_i}{n_i}}{\sum_{j=1}^{m} \frac{k_j}{n_j}} \tag{13}$$

where the denominator is obviously identical for all classes. We therefore have

$$\arg\max_{c_i \in C} \Pr(c_i|x) = \arg\max_{c_i \in C} \left(\frac{k_i}{n_i} \right) \tag{14}$$

□

The novel decision rule described in Equation 9 has therefore a straightforward practical interpretation and is easy to compute. The decision rule determines how large a fraction of the points with a specific class label in the domain is present in a given neighborhood query. Effectively, this decision rule adds a variable, local neighborhood-dependent class weight.

3.3 On the difference between weighted *k*NN and adjustment of prior class probabilities

In the following we show that the same effect of modifying the prior probabilities in the probabilistic interpretation cannot, in general, be achieved by using any weight in the basic weighted *k*NN approach discussed in Section 3.1.

Theorem 2. *A probabilistic kNN classifier with uniform prior probabilities is not equivalent to a class-based weighted kNN classifier.*

Proof. A weighted version of the probabilistic interpretation of *k*NN, taking Equation 11 as a starting point, can be formulated as:

$$w_i \cdot \Pr(c_i|x) = w_i \cdot \frac{\frac{k_i}{n_i} \cdot \Pr(c_i)}{\sum_{j=1}^{m} \frac{k_j}{n_j} \Pr(c_j)} \tag{15}$$

However, unless the prior probabilities are equal for all classes, there is no possible choice of w_i that also balances all $\Pr(c_j)$ in the denominator. □

Intuitively, our proposed method is effectively employing locally adaptive weights, which is not possible to model with a weight- or cost-based approach.

In a polytomous and heavily imbalanced classification problem where the largest majority class contains 1000 points, one of two minority classes, say c_2, contains 10 points, and the other minority class contains 50 points, a query with

Table 1. Dataset information for the real datasets. Abbreviations: dimensionality (Dim.), imbalance ratio (IR), number of classes (CL) and number of points (n)

Dataset	n	Dim	IR	Cl
appendicitis	106	7	4.05	2
balance	625	4	5.88	3
cleveland	297	13	12.31	5
coil 2000	9822	85	15.76	2
dermatology	358	34	5.55	6
ecoli	336	7	71.5	7
glass	214	9	8.44	6
haberman	306	3	2.78	2
hayes roth	160	4	2.10	3
hepatitis	80	19	5.15	2
marketing	6876	13	2.49	9
new thyroid	215	5	5.00	3

Dataset	n	Dim	IR	Cl
page-blocks	5472	10	175.46	5
phoneme	5404	5	2.41	2
satimage	6435	36	2.45	6
spectfheart	267	44	3.85	2
shuttle	58000	9	4558.60	7
thyroid	7200	21	40.16	3
titanic	2201	3	2.10	2
wine-red	1599	11	68.10	6
wine-white	4898	11	439.60	7
yeast	1484	8	92.60	10
usps	1500	50	4.00	2

2 minority points will be weighted differently dependent on which of the classes are present in the query:

$$\Pr(c_2|x_1) = \frac{\frac{2}{10}}{\frac{2}{10} + \frac{1}{1000} + \frac{0}{50}} = 0.995,\ \Pr(c_2|x_2) = \frac{\frac{2}{10}}{\frac{2}{10} + \frac{0}{1000} + \frac{1}{50}} = 0.909$$

This exemplifies how the weight for some class is dependent on the neighborhood of the query point.

4 Experimental Evaluation

4.1 Datasets

The datasets have been picked from the Keel dataset repository [1], and the USPS dataset is from Chapelle & Schölkopf [5]. The datasets were picked from these repositories, based on their imbalance ratio (IR) which is larger than 2. All datasets have numerical attributes. An overview on the used datasets is given in Table 1.

4.2 Compared methods

We compare our method "k-Nearest Neighbors with Balanced Prior Probabilities" (kNN-BPP) against representatives for the different categories of approaches, as discussed in Section 2. An overview is given in Table 2. As competitors, we include the conventional kNN classifier as well as more complex kNN-based algorithms that have been designed with the class imbalance problem in mind. The algorithms are also evaluated against the most common re-sampling strategies with a regular kNN-classifier. As can be seen in Table 2, several of the algorithm implementations are currently not publicly available. The original authors were contacted but did not provide the implementations. All implemented

Table 2. Compared methods

Method	**Short Name**	**Impl. Source**
k-Nearest Neighbors classifier [7]	kNN	Scikit-Learn [22]
Random undersampling [15, 25]	RUS	Imblearn [19]
Synthetic Minority Oversampling Technique [6]	SMOTE	Imblearn [19]
Class Based Weighted k-Nearest Neighbor over Imbalance Dataset [10]	CW-kNN	`https://github.com/Goettcke/kNN_BPP`
Direct Cost Sensitive kNN Classifier [23, 28]	Direct-CS-kNN	`https://github.com/Goettcke/kNN_BPP`
k-Nearest Neighbors with Balanced Prior Probabilities	kNN-BPP	`https://github.com/Goettcke/kNN_BPP`

algorithms will be made available with this paper. The algorithms were written following the requirements for *Scikit-Learn* implementations and written in Python.

4.3 Parameter Selection

We evaluate the methods using stratified 10-fold cross-validation. The k-value in these experiments varied for each dataset between 3 and 35.

For the *Direct-CS-kNN* classifier, a cost-matrix has to be defined. Since no method to generate such a cost matrix was proposed in the original papers we use the following cost-matrix: We construct an asymmetrical matrix that ensures the cost of predicting class c_j instead of c_i is proportional to the imbalance ratio between these two classes $C(i,j) = \frac{|c_i|}{|c_j|}$. Notice that, if class c_i is larger than c_j then the weight is greater than 1 which can be interpreted as it being costly to make this mistake. However, if c_i is a minority class, then $C(i,j) \in [0,1]$. This can be interpreted as a discount to making this type of mistake.

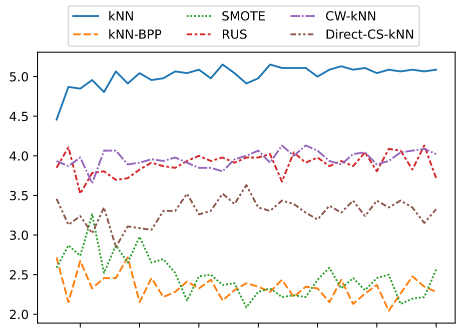
For *SMOTE* the number of nearest neighbors to include in the oversampling set was set to 6 as per the default in the Imblearn package. For random undersampling (RUS) and SMOTE, re-sampling was done to achieve uniform class distributions.

4.4 Evaluation measures

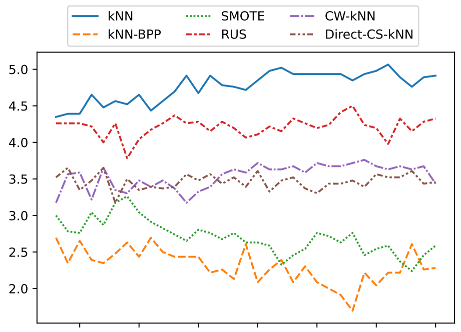
We evaluate the methods in terms of recall, taking the geometric mean over the classes (G-mean) and the macro average. We also tested precision, where all the methods show only minor differences, thus these results are not included.

4.5 Results

Ranking distribution over datasets and parameter values As a first overview we compare the methods performance as the average ranking for each



(a) G-mean score



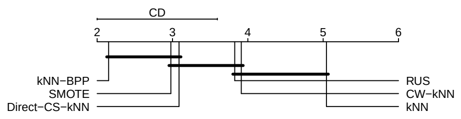
(b) Macro averaged recall

Fig. 1. Performance for $k \in [3, 35]$ in terms of the mean rank over all datasets

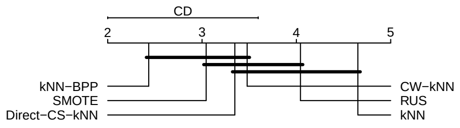
choice of k over all datasets. The rankings over all tested values of k are shown in Figure 1. Notice that a lower rank indicates better performance. The plots show how the evaluated algorithms seem to form 3 groups. The first group with the highest rank is the unmodified kNN algorithm. The second group contains the modified kNN algorithms from [10] and [28] as well as random undersampling [15]. In the third and best performing group we have SMOTE [6] and the proposed kNN-BPP method.

Analysis of statistical performance differences We performed a statistical analysis of the ranking differences for $k = 10, 20, 30$. The results are shown in critical difference plots in Figure 2. The plots show the average ranking of methods over all datasets together with a bar connecting methods that are not performing differently with statistical significance. We see our method on top or, in one case, second to SMOTE, although their performance is only different with statistical significance from some other methods in most cases. The classic, unchanged kNN classifier is typically worst, and several of the improved versions are not better with statistical significance. kNN-BPP is significantly better than the unchanged kNN classifier in all tests and better than RUS and CW-kNN in several, also in cases where SMOTE is not significantly better. In summary, the critical difference (CD) plots indicate that kNN-BPP performs as well or slightly better than a non-trivial oversampling technique, but without adding runtime to the original k-nearest neighbor classifier.

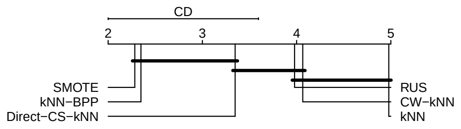
Distribution of raw performance To complement the picture, we also show the distribution of performance in terms of raw recall (G-mean and macro average) values over the datasets for $k = 10, 20, 30$ in Figure 3 (here higher values are better). We see that the distributions overlap strongly, but we can identify a tendency of SMOTE and our method to perform better than others.



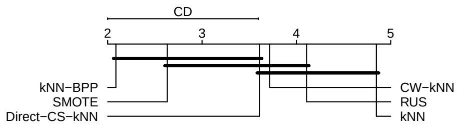
(a) G-mean $k = 10$



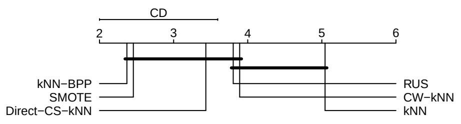
(b) Macro averaged recall $k = 10$



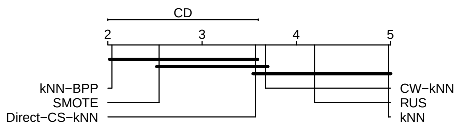
(c) G-mean $k = 20$



(d) Macro averaged recall $k = 20$

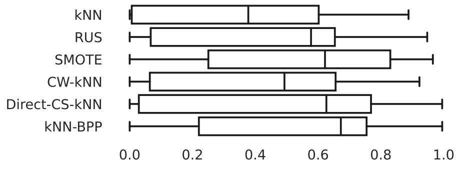


(e) G-mean $k = 30$

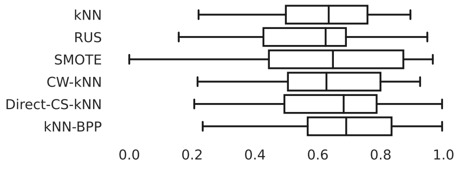


(f) Macro averaged recall $k = 30$

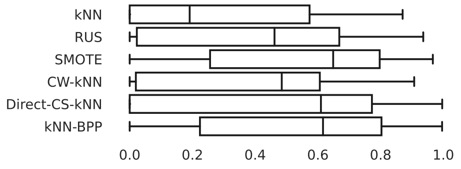
Fig. 2. Statistical assessment of performance differences: critical difference plots



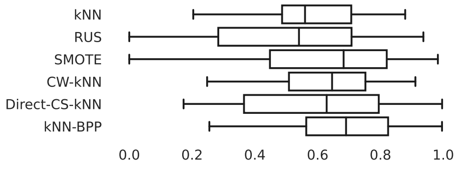
(a) G-mean $k = 10$



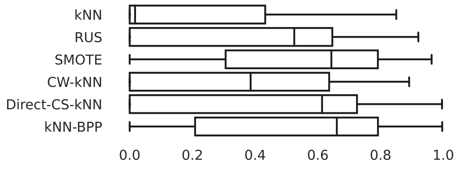
(b) Macro averaged recall $k = 10$



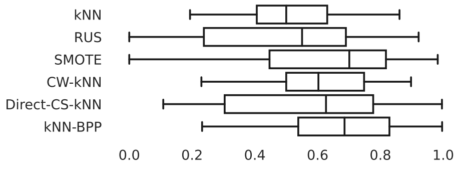
(c) G-mean $k = 20$



(d) Macro averaged recall $k = 20$

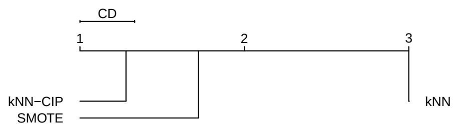


(e) G-mean $k = 30$

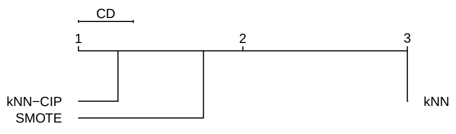


(f) Macro averaged recall $k = 30$

Fig. 3. Distribution of recall (G-mean, macro average) over the datasets



(a) G-mean



(b) Macro averaged recall

Fig. 4. Critical difference plots showing the significant disadvantage of SMOTE on multimodal minority class datasets in terms of G-mean and macro averaged recall

In the box plots over macro-averaged recall in Figure 3, we observe that the result distribution of *k*NN, CW-kNN and *k*NN-BPP are the most stable when changing k, but the median performance of *k*NN-BPP is also overall the highest and on par with SMOTE. In the box plots showing the result distributions over the datasets in terms of G-mean score in Figure 3(e), we observe that *k*NN-BPP is the only internal modification of *k*NN that consistently has a first quartile above 0. Conventional *k*NN is the most stable algorithm but the worst performer in terms of the distribution over G-mean score. We also observe that SMOTE and *k*NN-BPP are the best performers and almost as stable as *k*NN.

Disadvantage of oversampling Although our method is on par with the more complex oversampling method SMOTE, oversampling also has clear disadvantages. Firstly, adding synthetic points to the minority class increases the number of points in the dataset, which obviously increases the runtime. Secondly, they assume that the minority class follows some compact distribution. In the case of *SMOTE* it depends on the k value chosen, a larger k makes the assumed distribution approach a Gaussian distribution. To illustrate this problem we generated one hundred simple binary classification problems consisting of a multimodal minority class and a unimodal majority class. Both classes only span 2-dimensions and both the majority class mode and the minority class modes follow Gaussian distributions in both dimensions. The minority class modes are positioned on opposite sides of the majority class mode.

On these datasets we study the performance differences of regular *k*NN, *SMOTE*, and *k*NN-BPP. The problem for *SMOTE* is that it inserts harmful SMOTE instances between the two modes [4] since the convex hull defined by the minority class covers a large area of the majority class space.

In Figure 4 we see the statistical evaluation of the performance differences of oversampling in combination with majority voting *k*NN, compared to our method *k*NN-BPP, and the regular *k*NN classifier as a baseline. In all tests, k was set equal to 10, and 10-fold stratified cross-validation was used. For the re-sampling strategies the default parameters were used which ensures a uniform class distribution. In this simple test, *k*NN-BPP is significantly better than SMOTE. The reason behind the clear win is that the number of instances in the two modes is approximately the same, which means that oversampling within one mode by inserting a point on the vector between two of the minority points is approximately as likely as inserting a harmful SMOTE instance between the two modes. This is of course dependent on which points are picked at random so if SMOTE is extremely lucky it can perform better than *k*NN-BPP. However repeating this experiment a hundred times shows that this is typically not the case, hence the significant difference.

5 Conclusion

In this paper we addressed the importance of considering the recall when tackling imbalanced classification problems. We developed an elegant and straightforward *k*NN classifier, *k*NN-BPP, that balances prior class probabilities and thus treats imbalanced classes in a fair manner. The proposed *k*NN-BPP algorithm shows performance on par with a popular oversampler applied to the datasets in combination with the conventional *k*NN-algorithm for all measured *k*-values, while having the same computational complexity as regular *k*NN. The algorithm's difference from a weighted *k*NN-algorithm has been shown. *k*NN-BPP's advantage over other recent internal modifications of *k*NN over a wide set of *k*-values has been established.

For future work it could be interesting to investigate this idea for other classifiers that are amenable to a Bayesian probabilistic interpretation, and to perform case studies in application scenarios requiring special attention to fairness and bias [21].

References

1. Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S.: KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. J. Multiple Valued Log. Soft Comput. **17**(2-3), 255–287 (2011)
2. Batista, G.E.A.P.A., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explor. **6**(1), 20–29 (2004)
3. Bellinger, C., Drummond, C., Japkowicz, N.: Beyond the boundaries of SMOTE - A framework for manifold-based synthetically oversampling. In: ECML/PKDD (1). pp. 248–263. Springer (2016)
4. Bellinger, C., Sharma, S., Japkowicz, N., Zaïane, O.R.: Framework for extreme imbalance classification: SWIM - sampling with the majority class. Knowl. Inf. Syst. **62**(3), 841–866 (2020)
5. Chapelle, O., Schölkopf, B., Zien, A.: Introduction to semi-supervised learning. In: Semi-Supervised Learning, pp. 1–12. The MIT Press (2006)
6. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
7. Cover, T.M., Hart, P.E.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**(1), 21–27 (1967)
8. Davis, J., Goadrich, M.: The relationship between precision-recall and ROC curves. In: Cohen, W.W., Moore, A.W. (eds.) Proc. ICML. pp. 233–240 (2006). https://doi.org/10.1145/1143844.1143874

9. Domingos, P.M.: Metacost: A general method for making classifiers cost-sensitive. In: KDD. pp. 155–164. ACM (1999)
10. Dubey, H., Pudi, V.: Class based weighted k-nearest neighbor over imbalance dataset. In: PAKDD (2). pp. 305–316. Springer (2013)
11. Elkan, C.: The foundations of cost-sensitive learning. In: IJCAI. pp. 973–978. Morgan Kaufmann (2001)
12. Han, H., Wang, W., Mao, B.: Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. In: ICIC (1). pp. 878–887. Springer (2005)
13. Hand, D.J.: Measuring classifier performance: a coherent alternative to the area under the ROC curve. Mach. Learn. **77**(1), 103–123 (2009). https://doi.org/10.1007/s10994-009-5119-5
14. He, H., Bai, Y., Garcia, E.A., Li, S.: ADASYN: adaptive synthetic sampling approach for imbalanced learning. In: IJCNN. pp. 1322–1328. IEEE (2008)
15. Japkowicz, N.: The class imbalance problem: Significance and strategies. In: Proc. of the Int'l Conf. on Artificial Intelligence. vol. 56 (2000)
16. Japkowicz, N.: Assessment metrics for imbalanced learning. In: He, H., Ma, Y. (eds.) Imbalanced Learning: Foundations, algorithms, and applications, chap. 8, pp. 187–206. John Wiley & Sons (2013)
17. Kriminger, E., Príncipe, J.C., Lakshminarayan, C.: Nearest neighbor distributions for imbalanced classification. In: IJCNN. pp. 1–5. IEEE (2012)
18. Kubat, M., Matwin, S.: Addressing the curse of imbalanced training sets: One-sided selection. In: ICML. pp. 179–186. Morgan Kaufmann (1997)
19. Lemaitre, G., Nogueira, F., Aridas, C.K.: Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. J. Mach. Learn. Res. **18**, 17:1–17:5 (2017)
20. Liu, W., Chawla, S.: Class confidence weighted $k$nn algorithms for imbalanced data sets. In: PAKDD (2). pp. 345–356. Springer (2011)
21. Ntoutsi, E., Fafalios, P., Gadiraju, U., Iosifidis, V., Nejdl, W., Vidal, M., Ruggieri, S., Turini, F., Papadopoulos, S., Krasanakis, E., Kompatsiaris, I., Kinder-Kurlanda, K., Wagner, C., Karimi, F., Fernández, M., Alani, H., Berendt, B., Kruegel, T., Heinze, C., Broelemann, K., Kasneci, G., Tiropanis, T., Staab, S.: Bias in data-driven artificial intelligence systems - an introductory survey. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. **10**(3) (2020)
22. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., VanderPlas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
23. Qin, Z., Wang, A.T., Zhang, C., Zhang, S.: Cost-sensitive classification with k-nearest neighbors. In: KSEM. pp. 112–131. Springer (2013)
24. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
25. Rodieck, R.W.: The density recovery profile: a method for the analysis of points in the plane applicable to retinal studies. Visual neuroscience **6 2**, 95–111 (1991)
26. Siddappa, N.G., Kampalappa, T.: Imbalance data classification using local mahalanobis distance learning based on nearest neighbor. SN Comput. Sci. **1**(2), 76 (2020)
27. Song, Y., Huang, J., Zhou, D., Zha, H., Giles, C.L.: IKNN: informative k-nearest neighbor pattern classification. In: PKDD. pp. 248–264. Springer (2007)
28. Zhang, S.: Cost-sensitive KNN classification. Neurocomputing **391**, 234–242 (2020)