Item-based Classification for Robustness Against Noise*

Alexandros Nanopoulos[1]**, Apostolos N. Papadopoulos[1], Tatjana Welzer-Druzovec[2], and Yannis Manolopoulos[1]

[1]Aristotle University of Thessaloniki, Dept. of Informatics, Greece
[2]University of Maribor, Faculty of Electrical Eng. and Computer Science, Slovenia
{alex,apostol,manolopo}delab.csd.auth.gr,welzeruni-mb.si

Abstract. The existence of noise in the data significantly impacts the accuracy of classification. In this paper, we are concerned with the development of novel classification algorithms that can efficiently handle noise. To attain this, we recognize an analogy between k nearest neighbors (kNN) classification and user-based collaborative filtering algorithms, as they both find a neighborhood of similar past data and process its contents to make a prediction about new data. The recent development of item-based collaborative filtering algorithms, which are based on similarities between items instead of transactions, addresses the sensitivity of user-based methods against noise in recommender systems. For this reason we focus on the item-based paradigm to provide improved robustness, compared to kNN algorithms, against noise for the problem of classification. We propose two new item-based algorithms, which are experimentally evaluated with kNN. Our results show that, in terms of precision, the proposed methods outperform kNN classification by up to 15%, whereas compared to other methods like the C4.5 system, improvement exceeds 30%.

1 Introduction

Classification involves the construction of a model, denoted as *classifier*, for the mapping of data to a set of predefined and non-overlapping classes. The accuracy of a classifier is negatively affected by the existence of noisy data, that is, data records with incorrect attribute values. The reason is that noise confuses the learning process and causes the constructed classifier to overfit the data [6]. In this paper we are interested in developing novel classification algorithms that are robust and construct classifiers with significantly improved accuracy in the presence of noise.

The development of classification algorithms that handle noise is a worthwhile problem, due to the direct impact that noise has on the reliability of data analysis' results and, thus, on decision making. Controllable data-acquisition

* Work supported by a bilateral Greek-Slovenian project.
** A. Nanopoulos is supported by IKY Post-doctoral scholarship.

procedures can limit the amount of noise, rendering it easily resolvable with preprocessing techniques [12]. In other situations, however, the intrusion of noise cannot be restrained; for instance, data that are collected online in e-commerce sites (e.g., surveys, questionnaires, purchases, etc.). It has been reported [10] that to a large degree the visitors of e-commerce sites tend to deliberately provide false values, for reasons varying from privacy concerns to "shilling" attacks [10]. In such cases preprocessing techniques are not effective. What is needed is classification algorithms that can tackle noise.[1]

Existing classifiers can be divided into two categories [8], *eager* and *lazy*. In contrast to an eager classifier (e.g., decision tree), a lazy classifier [1] builds no general model until a new sample arrives. A k-nearest neighbor (kNN) classifier [5] is a typical example of the latter category. It works by searching the training set for the k nearest neighbors of the new sample and assigns to it the most common class among its k nearest neighbors. In general, a kNN classifier has satisfactory noise-rejection properties. Other advantages of a kNN classifier are that it (a) is analytically tractable, (b) for $k = 1$ and unlimited samples the error rate is never worse than twice the Bayes' rate, and (c) it is simple to implement.

Due to the aforementioned characteristics, kNN classifiers are very popular and find many applications. Nevertheless, although they present adequate efficiency for low amount of noise, our results (Section 4) indicate that their performance degrades for medium and high noise. Interestingly enough, an analogous conclusion has been reported [14] for the seemingly different problem of *user-based* (UB) collaborative-filtering (CF).[2] Albeit the different objectives between a kNN classifier and a UB CF algorithm, they both have a common way of working: given a new datum (record and user-transactions, respectively), they first find a neighborhood of similar past data and, then, they process its contents to make predictions about the new datum. The bottleneck in this procedure is the search for neighbors among a large population of potential neighbors [7].

The inefficiency of UB algorithms for significant amounts of noise has recently lead to the development of novel *item-based* (IB) CF [14] algorithms which, in contrast to UB ones, base their prediction on similarities between items instead of transactions. UB algorithms are outperformed by IB ones, because the latter can better isolated noise, as in each user-transaction only the items with strong similarities are considered. Therefore, based on the previously described analogy between classification and CF algorithms, the question that motivates our research is: can we involve the paradigm of IB algorithms to build lazy classifiers that are more robust against noise than kNN classifiers?

---

[1] A racy account of the people's intention to provide false data when being asked, is described by Witten et al. [16]: when the authors ask their students to give their birthdate *after* they have explained to them that they want to demonstrate the "birthday paradox", it takes 366 tries to find two students with the same birthdate!

[2] Note that recommender systems and CF algorithms, in particular, find many and significant applications [13].

Our contributions are summarized as follows. First, we demonstrate that the IB paradigm can be involved to develop robust classifiers. We expound that the straightforward adoption of IB methods leads to less accurate classification. Therefore, we develop a novel algorithm, which addresses the deficiencies of straightforward adoption. Finally, we recognize a trade-off between the robustness of the new IB algorithm against noise and the high precision of kNN classification in the absence of noise. For this reason, we develop a hybrid algorithm that combines the advantages of both approaches. Our experimental results illustrate that: (i) in the presence of noise the proposed IB algorithm improves the precision of kNN classification up to 15%; (ii) the hybrid algorithm is a good amalgamation, because it attains precision analogous to a kNN classifier, when no or low amount of noise is added, and analogous to an IB algorithm, when medium or high amount of noise is added.

The rest of this paper is organized as follows. Section 2 describes related work. In Section 3 we develop the proposed method, whereas in Section 4 we present the experimental results. We conclude this paper in Section 5.

2 Related work

Due to its simplicity and good performance, kNN classification has been studied thoroughly [5]. Several variations were developed [3], like the distance-weighted kNN, which puts emphasis on nearer neighbors, and the locally-weighted averaging, which uses kernel width to controls the size of neighborhood that has large effect. Also, kNN classification has been combined with other methods and, instead of predicting a class with simple voting, prediction is done by another machine learner (e.g., neural-network) [4]. In our research we are interested in examining the essential characteristics of kNN and item-based classification, that is, to compare classification based on similarities between samples (kNN) in contrast to similarities between attribute values (item-based). For this reason, to make comparison clearer, we did not examine techniques like the aforementioned ones. Moreover, such techniques (e.g., combination with other machine learners) can be applied for both types of classifiers.

Another category of research that is related to ours is the integration of association-rules mining with classification [9]. Integration is done by focusing on mining *class association rules* (CARs), i.e., association rules between items and classes. CARs are used to build an eager classifier, which was experimentally shown to achieve better precision compared to the C4.5 classification system. Basically, CARs correspond to multi-order dependencies between items and class labels. Since the number of CARs may increase rapidly and impact the accuracy of classification, their number is controlled with parameters for minimum support and confidence, and with heuristics for grouping rules. In contrast, we focus on up to second-order dependencies to develop item-based classifiers. Second-order dependencies are effective in capturing correlations between attributes and classes. Additionally, they make our approach much more simple, because it does not need the tuning of parameters like minimum support and confidence, which,

in general, is a difficult task [6]. The performance of CARs has not been examined with respect to noise. It is interesting to examine, in our future work, if multi-order dependencies help in the presence of noise. Other uses of association rules include the mining of partial classifications [2]. Partial classification does not cover all classes and all examples of any given class, and is used when complete classification is infeasible or undesirable.

In the recent years the field of CF has found many important applications in real-world cases [13]. Although UB CF algorithms are simple and quite effective, IB methods [14] have been proposed to address their limitations, for instance, sensitivity to noise. Nevertheless, to our knowledge, no work has studied the involvement of the IB paradigm to the problem of classification.

3 Developing item-based classifiers

We first summarize the IB paradigm and describe a simple classifier that is based on its direct implementation. Next, we elaborate further and propose extensions for more effective classification.

3.1 Direct implementation of the IB paradigm

Let I be a domain of items, D the collection of past transactions, and $T = \{i_1, \ldots, i_m\}$ the transaction of the target user. An IB collaborative-filtering algorithm finds for each item $i_j \in T$ $(1 \leq j \leq m)$ the set $N(i_j)$ of k items with which i_j is most correlated within transactions of D. The items which belong to $\bigcup_{j=1}^{m} N(i_j) - T$ (i.e., excluding those contained in T) are recommended to the user. Correlation between two items i_1 and i_2 can be measured in several ways [6]: (a) for items with numerical values the *adjusted cosine similarity* measure has been proposed, (b) for nominal items a commonly used measure is *confidence*. The confidence is an estimator for the probability $P(i_2|i_1)$ and is computed as: $\text{conf}(i_1, i_2) = \frac{\text{supp}(i_1, i_2)}{\text{supp}(i_1)}$, where $\text{supp}(i_1, i_2)$ and $\text{supp}(i_1)$ denote the *support* of itemsets $\{i_1, i_2\}$ and $\{i_1\}$, respectively.

A classifier that directly follows the IB paradigm has to take into account the correlations between attribute values in the training samples. Each such sample $S = \{v_1, \ldots, v_m, c\}$, where v_i is the value of i-th attribute and c is the class label, can be considered as a multidimensional transaction. For simplicity we henceforth assume that the attributes are either nominal or have been discretized.[3] Using the confidence measure, we find, for each attribute value v_i, the set $N(v_i)$ of the k attribute values that are most correlated with v_i in the training samples (we allow the inclusion of v_i itself in $N(v_i)$). This way, for a new sample $X = \{v_1, \ldots, v_m\}$, we form the set $U(X) = \bigcup_{v_i \in X} N(v_i)$. To predict the class of X, we also have to find for each $v_j \in U(X)$ the set $N'(v_j)$ of the k' class labels

[3] Extension to continuous attributes is possible with works [15] that find correlations between quantitative items.

that are most correlated with v_j. (In our implementation we set $k' = 1$.) Correlation in this case reflects $P(c|v_j)$ and is measured as: $\text{conf}(v_j, c) = \frac{\text{supp}(v_j,c)}{\text{supp}(v_j)}$. For each class label c, we compute sum the $\sum_{v_j \in U(X)} \text{conf}(v_j, c)$. Finally, we select for prediction the class with the greatest sum.

The direct algorithm is denoted as *Item-based* classifier (IBC). To understand the intuition behind IBC we consider that: even when several attributes values in the training set have noise, IBC focuses only on the attributes values that (a) are correlated with those in the new sample, and (b) can confidently indicate a class. Thus, IBC more effectively avoids noisy attributes values, because they tend to form weaker correlations and are less frequently included in the $U(X)$ set. In contrast, when kNN algorithm finds the nearest neighbors, it takes into account the noise in the attribute values while computing the similarities. Nevertheless, IBC has two problems:

1. It is based on *first-order* correlations between attribute values and class labels. This is restrictive, since higher-order dependencies may improve the accuracy of classification.
2. $U(X)$ has the propensity of getting large. This may be desirable for IB CF algorithms (for higher recall). However, it can hinder the accuracy of IBC, especially when combined with the 1st problem, because the more focused $U(X)$ is, the better the prediction becomes.

In the following of this section we develop solutions to the aforementioned problems. Regarding the execution time, IBC needs to maintain the support values that are required to compute the confidence values when a new sample arrives. This can be done easily with two-dimensional arrays or with trie structures [6], which can be easily updated with insertions/deletions of samples. Therefore, IBC classifies a new sample in lazy manner and the pre-computed support values play a role analogous to that of an index structure that speeds-up the finding of nearest neighbors [11] in kNN classifiers.

3.2 Pairwise-based classifier

Initially we consider the first problem of IBC. As described, CAR classifiers [9] detect dependencies of various orders between the attribute values and the classes labels, but require careful tuning of several parameters. As a conciliation between first-order dependencies in IBC and multi-order dependencies in CAR-like methods, we choose to consider up to second-order dependencies, which: (a) can effectively capture many associations between attribute variables and class labels; (b) are simple and do not require parameter tuning; and (c) do not present significant storage overhead, in contrast to storing much more many support values that would be required to calculate multi-order dependencies. Because we want to estimate probabilities of the form $P(c|\{v_i, v_j\})$ ($\{v_i, v_j\}$ corresponds to a 2-itemset) through $\text{conf}(\{v_i, v_j\}, c)$ values, we find and store supports of the form $\text{supp}(\{v_i, v_j, c\})$ from the training samples.

With respect to the second problem of IBC, given a new sample $X = \{v_1, \ldots, v_m\}$, we propose the following method for the formation of $U(X)$. For

each $v_i \in X$, we include in $U(X)$ only those $v_j \in N(v_i)$ for which it also holds that $v_j \in X$, that is, we concentrate on pairs from the new sample X. The latter condition restricts the size of $U(X)$ compared to the case of IBC. Differently from IBC, we keep for each $v_j \in U(X)$ the associated v_i. This way, $U(X)$ contains pairs of the form (v_i, v_j), that are tested against each class label c by computing $\text{conf}(\{v_i, v_j\}, c)$ as described previously.[4] The confidence measures for each class are summed and the class with the greatest sum is selected for prediction. The resulting algorithm is denoted as *Pairwise-based* classifier (PBC).

The incentive to the design of the PBC algorithm is described as follows. We want to base our prediction on second-order dependencies between attribute values and class labels. Therefore, within a record we have to consider the confidence measure between each pair of attribute values and all class labels. To avoid the impact of noise, we take into account only the pairs of attribute values for which there is a strong correlation between them. To achieve this, for each attribute value in a record, we select between the remaining ones only those that belong to its neighborhood. This way, as described, we manage to both consider second-order dependencies and to restrain the length of $U(X)$, by considering pairs with strongly correlated items.

An example of PBC is depicted in Figure 1. The data table contains five training samples over three attributes (A, B, C) and one class label. The new sample is $X = \{a_1, b_2, c_2\}$, depicted in the last row. Assuming that $k = 2$, the neighborhood sets for each attribute values of X are depicted in the upper-right part of the figure. For instance, $N(a_1) = \{a_1, b_1\}$, because $\text{conf}(a_1, a_1) = 1$ and $\text{conf}(a_1, b_1) = 3/4$ are the highest two for a_1. The resulting $U(X)$ set is depicted bellow. For instance, associated with a_1 in $U(X)$ is only a_1 itself (forming the pair (a_1, a_1)). Although $b_1 \in N(a_1)$, it does not belong in X and is excluded from $U(X)$. The same applies for c_1 with respect to $N(b_2)$, whereas both c_2 and a_1, which belong in $N(c_2)$, are included in $U(X)$ associated with c_2. For each member of $U(X)$, PBC examines the confidence values against the two class labels l_1 and l_2. For instance, for (a_1, a_1) we compute a $\text{conf}(a_1, l_1) = \frac{\text{supp}(\{a_1, l_1\})}{\text{supp}(a_1)} = 3/4$, which checks a first-order dependency between a_1 and l_1. In contrast, for pair (c_2, a_1) we check a second-order dependency by computing $\text{conf}(\{c_2, a_1\}, l_1) = \frac{\text{supp}(\{\{a_1, c_2, l_1\})}{\text{supp}(\{a_1, c_2\})} = 2/2 = 1$. Taking the sum of confidence values for each class, we get $\text{Sum}(l_1) = 2.75$ and $\text{Sum}(l_2) = 1.25$, thus we assign the new sample to class l_1.

3.3 Extensions of the pairwise-based classifier

The assigning of class labels based on the greatest sum value, a method that is denoted as majority voting, can impact the accuracy of PBC in the presence of noise, when some classes have much more samples in the training set than the others (skewed class distribution). Since larger classes get a larger sum value

[4] Similarly to IBC, we allow v_i to be included in $N(v_i)$ as (v_i, v_i). When it is tested against a class c, we compute $\text{conf}(v_i, c)$, which corresponds to first-order dependency.

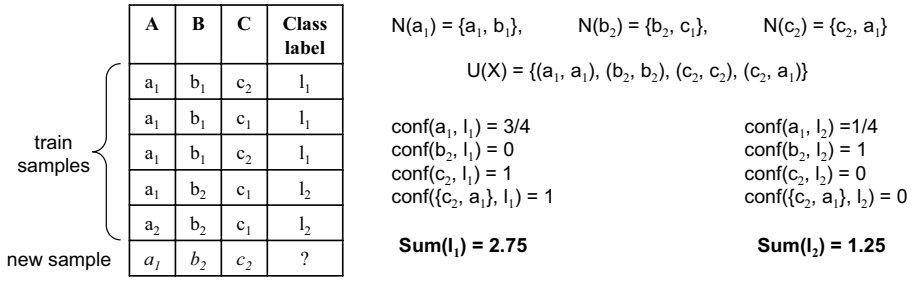


Fig. 1. Example of PBC.

more easily, PBC tends to bias its prediction towards them, thus the accuracy for smaller classes reduces. To overcome this problem we use a simple variation of the majority voting: Instead of selecting the class with the greater sum of confidence values, we select the one with the largest deviation from the expected value of this sum. In particular, for each class c, we take into account two sums, $S_1(c)$ and $S_2(c)$. $S_1(c)$ is computed as described previously. For $S_2(c)$, each time a confidence for class c is computed, we add to $S_2(c)$ the value $\frac{1}{supp(c)}$. In the example of Figure 1, when we compute $\text{conf}(a_1, l_1)$, we add to S_1 the value $3/4$, and to S_2 the value $1/\text{supp}(l_1) = 1/3$. In the end, we select the class with the largest difference $S_1(c) - \alpha S_2(c)$. The α parameter helps to to control the bias: e.g., with $\alpha = 0$ (or even $\alpha < 0$) the larger classes are favored, whereas with $\alpha > 0$ the smaller classes are favored. Its tuning will be explained in Section 4.

The characteristics of PBC make it more robust against noise, compared to IBC and kNN classifiers. Nevertheless, by careful experimentation we have recognized a tradeoff between the good performance of kNN classifiers when no additional noise is added and the robustness of PBC when the amount of noise is significant. For this reason we propose a hybrid approach. When a new sample X arrives, we first apply kNN classification. If all k nearest neighbors belong to the same class, then we assign X to this class. Otherwise, at the moment that a neighbor with different class has been found, we stop the kNN classifier and apply PBC. The resulting algorithm is denoted as *Hybrid-based* classifier (HBC). The intuition behind HBC is that when kNN reaches a unanimous decision, then low or no noise exists (noise tends to limit such unanimous decisions). For this reason, the prediction of kNN classifier is preferred in these cases. Otherwise, PBC is preferred, because it offers better robustness against noise. This way, HBC combines the advantage of both approaches.

4 Performance results

We examined experimentally the performance of the described classifiers: kNN, IBC, RBC, and HBC, which we implemented in C++. For comparison purposes, we also examined classification with C4.5 decision-tree, which is denoted as DT.

We selected the precision of prediction as our performance measure. For kNN classification we used the Jaccard similarity measure (nominal attributes). To tune α for RBC and HBC, we set it with values between -1 and 0 for low noise values (less than 20%; see below). For higher noise values, we set it to 1 to reduce bias against smaller classes, as described in Section 3.3. We tested several values for the k parameter (for all methods besides DT). When noise is added, in some cases precision increases with larger k values, whereas in some others it decreases. For this reason in each measurement we selected the best results for k between 3 and 10 (larger values leaded to non stable precision).

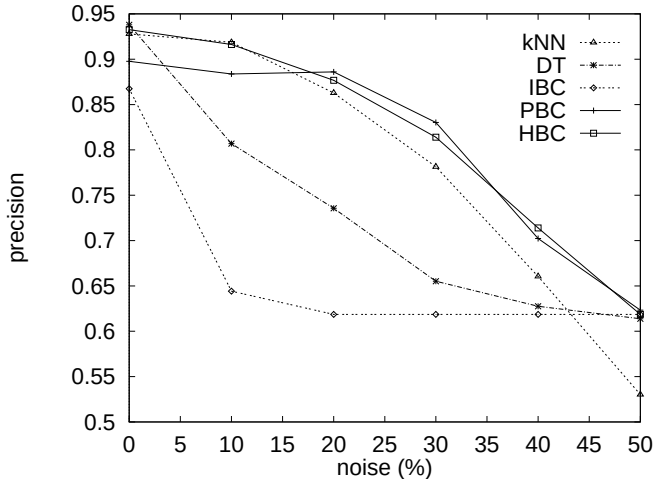
We used the following real data sets from the UCI Machine Learning Repository[5]: Congressional Voting Records Database (denoted as Voting), Mushrooms Database (denoted as Mushrooms), Nursery Database (denoted as Nursery), and Pittsburgh Bridges Database (denoted as Bridges). All the data sets contain nominal attributes. The smallest, in number of rows, set (Bridges) contains 108 samples, whereas the the largest (Nursery) 12,960. The distribution of classes was more skewed in the Bridges data set than in the others.

Noise is added in two ways: (1) We modify each attribute value (not the class labels) with probability denoted as *noise*, which is a parameter. We focused on random noise and when a value is replaced, we select one of the remaining ones with uniform probability. (2) We modify the class labels (not the attribute values) with probability equal to *noise*. Again, when we replace a class label, we select one of the others with uniform probability. The two types of noise, which are denoted as attribute noise and class noise, respectively, are different and we want to examine both of them.

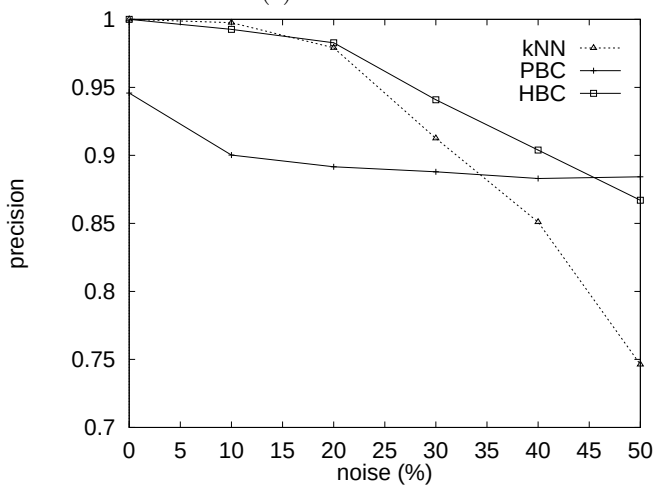
First, we tested the Voting data set against attribute noise. We measured precision with 10-cross validation. The results with respect to the *noise* parameter are given in Figure 2a. IBC, due to the two problems that are described in Section 3.1, has the lower precision. As expected, DT performs well when *noise*$= 0$. However, its precision is reduced significantly as *noise* increases. The comparison between kNN and PBC illustrates the tradeoff that we described in the previous section: for low *noise* values kNN performs better. But as *noise* increases, RBC due to its robustness becomes better; their difference in precision reaches 9% for large *noise* values. HBC tracks each time the best of the latter two algorithms. When *noise* is low, the precision of HBC is analogous to that of kNN. As *noise* increases, HBC behaves similarly to RBC. This explains the motivation for the development of the hybrid approach. For larger values of *noise*, the precision of all methods besides kNN converges to the same point, whereas kNN presents the worst precision for large *noise* values.

Next, we examined the Mushrooms data set, which contains classes that are, in general, easily detectable. We added attribute noise. To make the comparison more challenging, we selected as evaluation set a 10% of samples whose majority belongs to the less frequent between the two classes (non-edible) and we used the other 90% for training. The results are depicted in Figure 2b. For clarity, we omit the results for DT and IBC, because they consistently present the worst

[5] http://www.ics.uci.edu/∼mlearn/MLRepository.html



(a)

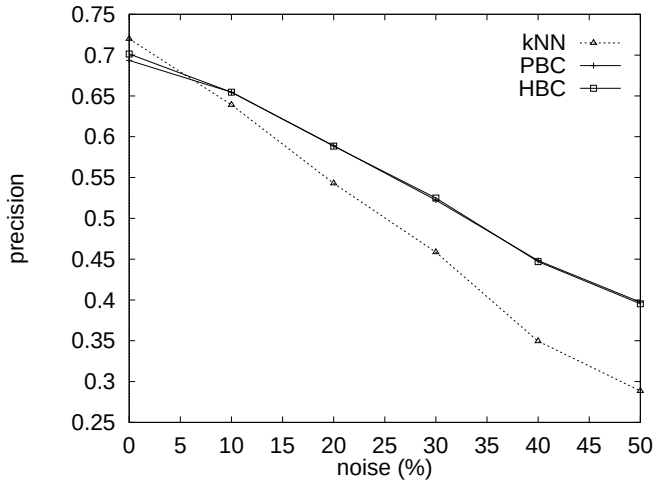


(b)

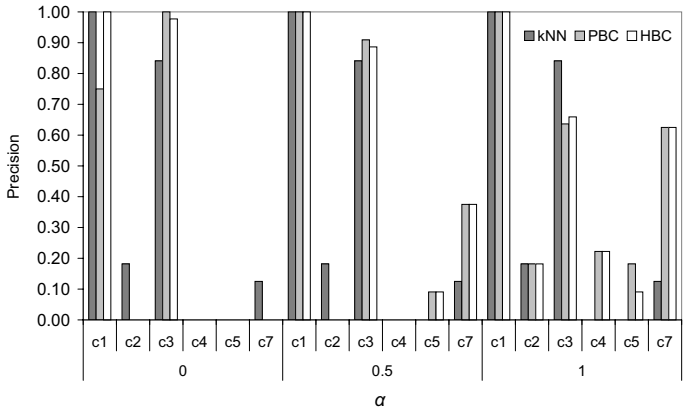
Fig. 2. Precision w.r.t. *noise* for: (a) Voting, (b) Mushrooms data set.

performance, compared to the other methods, when noise is added. Similarly to the previous experiment, the same tradeoff is observed. For lower *noise* values kNN outperforms RBC, whereas the latter becomes better when *noise* increases. HBC again tracks the best of the two algorithms. For low *noise* it outperforms RBC by 5.5% and for larger *noise* its difference from kNN reaches 12% (in the same case, the difference between RBC and UB is more than 14%).

Also, we examined the Nursery data set for class noise. The results (using 10-cross validation) are illustrated in Figure 3a, which are similar to the previous ones in terms of relative performance.



(a)



(b)

Fig. 3. (a) Precision w.r.t. *noise* for Nursery data set. (b) Precision for each class in the Bridges data set w.r.t α.

Finally, we tested the impact of α for skewed class distribution. As mentioned, when $\alpha = 0$, prediction is biased towards the largest classes. As α increases, the bias against less frequent classes reduces. The Bridges data set contains 7 class labels, but we omit the sixth label (it was 'NIL' and was contained by only one sample). Compared to the number of samples of the smallest class (c_7), the largest class (c_3) contains 5 times more samples. The second largest (c_1) contains twice more samples. We added no noise in order not to change the existing skew in class distribution. The precision with respect to α is depicted, separately for each class, in Figure 3b. When $\alpha = 0$, the largest classes (c_1 and c_3) are predicted with much larger precision than the others. The kNN classifier achieves a small, but larger non-negligible precision for two of the smaller classes (c_2 and c_7). For the remaining classes (c_4 and c_5), all methods result to zero precision. As α increases, PBC and HBC are not less biased towards the large classes. For $\alpha = 1$, the precision of PBC and HBC for c_2 is analogous to that of kNN, whereas they outperform kNN for the smallest class c_7. Moreover, PBC and HBC have a small, but non-negligible precision for classes c_4 and c_5, for which kNN presents zero precision. On the other hand, the precision of PBC and HBC for the largest class c_3 reduces when $\alpha = 1$. Thus, the reduction of bias results to a trade-off between the precision for smaller and larger classes.

5 Conclusions

We propose new algorithms for lazy classification, which follow the item-based paradigm. In contrast to kNN classification that predicts according to similarities between samples in the training set, item-based classification predicts according to similarities between attribute values. This characteristic makes item-based classification more robust against noise.

Our first algorithm, PBC, resolves the problems of the straightforward adoption of the IB paradigm. Our second algorithm, HBC, is a hybrid approach that combines the advantages of kNN and PBC algorithms. Our experimental results show that, in the presence of attribute or class noise, the proposed algorithms outperform kNN classification in terms of precision by up to 15%. The improvement against the C4.5 classification system is more than 30%.

In our future work we will extend item-based classification algorithms to consider distance weighting (between attribute values), we will examine mixtures of nominal and continuous attributes, and complex voting methods (e.g., integration with neural networks).

References

1. D. W. Aha. Editorial. *Artificial Intelligence Review (Special Issue on Lazy Learning)*, 11(1-5):1–6, 1997.
2. K. Ali, S. Manganaris, and R. Srikant. Partial classification using association rules. In *Proc. Knowledge Discovery and Data Mining Conf.*, pages 115–118, 1997.

3. C. Atkeson, A. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5).
4. C. Atkeson and S. Schaal. Memory-based neural networks for robot learning. *Neurocomputing*, 9.
5. B.V. Dasarathy. *Nearest Neighbor Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, 1991.
6. J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.
7. J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proc. ACM Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 230–237, 1999.
8. M. James. *Classification Algorithms*. John Wiley & Sons, 1985.
9. B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proc. Knowledge Discovery and Data Mining Conf.*, pages 80–86, 1998.
10. M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre. Collaborative recommendation: A robustness analysis. *ACM Transactions on Internet Technology*, 4(4):334–377, 2004.
11. A. Papadopoulos and Y. Manolopoulos. *Nearest Neighbor Search: A Database Perspective*. Springer (Series in Computer Science), 2005.
12. D. Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann, 1999.
13. P. Resnick and H. R. Varian. Recommender systems (special issue). *Communications of the ACM*, 40(3), 1997.
14. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. World Wide Web Conf.*, pages 285–295, 2001.
15. R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proc. ACM Conf. on Management of Data*, pages 1–12, 1996.
16. I. Witten, A. Moffat, and T. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images (2nd Edition)*. Morgan Kaufmann, 1999.