Journal of Classification (2020) 37:143–157
https://doi.org/10.1007/s00357-019-9303-4

Core Clustering as a Tool for Tackling Noise in Cluster Labels

Renato Cordeiro de Amorim[1] · Vladimir Makarenkov[2] · Boris Mirkin[3,4]

Published online: 30 March 2019

Abstract

Real-world data sets often contain mislabelled entities. This can be particularly problematic if the data set is being used by a supervised classification algorithm at its learning phase. In this case, the accuracy of this classification algorithm, when applied to unlabelled data, is likely to suffer considerably. In this paper, we introduce a clustering-based method capable of reducing the number of mislabelled entities in data sets. Our method can be summarised as follows: (i) cluster the data set; (ii) select the entities that have the most potential to be assigned to correct clusters; (iii) use the entities of the previous step to define the core clusters and map them to the labels using a confusion matrix; (iv) use the core clusters and our cluster membership criterion to correct the labels of the remaining entities. We perform numerous experiments to validate our method empirically using *k*-nearest neighbour classifiers as a benchmark. We experiment with both synthetic and real-world data sets with different proportions of mislabelled entities. Our experiments demonstrate that the proposed method produces promising results. Thus, it could be used as a preprocessing data correction step of a supervised machine learning algorithm.

Keywords Label noise · Clustering · *k*-means · Core clustering · Minkowski distance

1 Introduction

The arrival of new technologies has made it quite easy to acquire vast amounts of data. However, labelling such data (i.e., assigning the class memberships to entities) is far from

✉ Renato Cordeiro de Amorim
r.amorim@essex.ac.uk.ac.uk

[1] School of Computer Science and Electronic Engineering, University of Essex, Colchester, CO4 3SQ, UK

[2] Département d'informatique, Université du Qubec à Montréal, C.P. 8888 succ. Centre-Ville, Montreal QC H3C 3P8, Canada

[3] Department of Computer Science and Information Systems, Birkbeck University of London, Malet Street, London WC1E 7HX, UK

[4] Department of Data Analysis and Machine Intelligence, National Research University Higher School of Economics, Moscow, Russian Federation

being a trivial task. More often, the labelling process is subject to human interpretation, leading to inconsistencies caused by disagreements or mistakes in human, including expert, labelling. This may have significant implications in machine learning. Mathematically, the problem can be formulated as follows. Let $\mathbf{X}$ be a data matrix composed of the entity (i.e., object) vectors $x_1, x_2, ..., x_i, ..., x_n \in \mathbb{R}^m$, where m is the number of features (i.e. variables) characterising every entity. Let us also define the label vector $\mathbf{y} = (y_1, y_2, ..., y_i, ..., y_n)$ in which y_i represents the label of x_i. Classification algorithms will attempt to determine the function $h : \mathbf{X} \rightarrow \mathbf{y}$ and apply this function to an unlabelled data matrix. Hence, the accuracy of any classification algorithm is intrinsically related to the availability and correctness of labels.

Meanwhile, mistakes and disagreements in the labelling process are not uncommon, thus leading to a certain proportion ρ of incorrect labels in y. The higher the value of ρ, the lower ability of h to correctly map $\mathbf{X}$ to $\mathbf{y}$. Since this issue affects most classification algorithms, one may be interested in reducing ρ as much as possible. One could argue that minimising the role humans play in the labelling process would reduce the value of ρ. Human experts could be asked to label a small sample of entities and a computerised method could label the remaining entities. However, methods following the active learning framework (Settles 1648) are not flawless and may also lead to high proportions ρ of mislabelled entities.

In this paper, we introduce a clustering-based method which is capable of reducing ρ. There are three main assumptions behind our method:

1. For each cluster S_k, its centre c_k does not much depend on the wrong labels.
2. For each cluster S_k, there must be an identifiable cluster core S'_k around its centre c_k $(k = 1, 2, ..., K)$.
3. For any two entities from the same core $x_i, x_j \in S'_k$, their correct labels must be the same.

Therefore, we will identify cluster centres, build the core clusters around them, use these core clusters to identify mislabelled entities, if any, and relabel them according to the core clusters and a specific cluster membership criterion.

2 Related Work

Real-world databases are estimated to contain around five percent of encoding errors, all fields taken together, when no specific measures for improving the coding errors are taken (Maletic and Marcus 2000; Redman 1998; Orr 1998; Frénay and Verleysen 2014). The exact value of this estimate may be subject to debate; however, the existence of such errors can have a considerable impact on the accuracy of classification algorithms. With this in mind, we begin by providing further details regarding label noise in the next subsection followed by a subsection on clustering.

Semi-supervised clustering is a different research area. The general idea is that some previous knowledge can be used to guide the clustering process. This previous knowledge usually has the form of pairwise must-link and may-not-link constraints. These constraints can be used to generate an initial, rather limited, set of cluster labels that are assumed to be correct. In our setting, a certain proportion of entities is mislabelled, but the investigator knows neither the proportion, nor the mislabelled entities. Neither is he nor she interested in classifying the entities. Our approach focuses on the data preprocessing stage in which we attempt to reduce the number of incorrect labels. Therefore, our paper has very little to do with semi-supervised clustering. Nonetheless, we direct interested readers to relevant surveys (Grira et al. 2004; ZHU 2006; Jain et al. 2014) and references therein.

2.1 Label Noise

A data set may have two distinguishable types of noise, feature noise and label noise (ZHU and WU 2004). Feature noise is present in the actual values of features. By consequence, this type of noise affects the data matrix **X**. Label noise affects the labels assigned to each entity, altering the actual values of **y**. Notice that label noise affects solely the observed label, **y**, not the true label of each entity. In particular, label noise does not alter the data matrix **X**.

Feature selection and feature weighting can be used to address the presence of feature noise in data. These two have an older and longer body of research than that of label noise (see for instance Guyon and Elisseeff (2003), Saeys et al. (2007), De Amorim (2016), and Frénay and Verleysen (2014) and references therein). Having received much less attention, label noise remains an unsolved problem in supervised classification (Bouveyron and Girard 2009). However, the label noise is potentially more harmful than the feature noise (ZHU and WU 2004; Saáez et al. 2014). Two possible reasons for this, according to Frénay and Verleysen (2014), are as follows. First, the degree of relevance of each feature may vary, whereas any misclassified label y_i will always have a large impact in learning. In fact, a single feature may have different degrees of relevance in different clusters (De Amorim and Mirkin 2011). Second, an entity x_i is described over m features (usually $m >> 1$), but it has a single label y_i. Label noise may be less harmful than feature noise if there is a large number of features compromised by feature noise (Quinlan 1986).

There can be different potential sources of label noise (Frénay and Verleysen 2014):

1. The labeller may not have been provided with sufficient information in order to produce reliable labels (Hickey 1996).
2. The error may occur in the labeller itself. One should note that these two cases may occur even if the labeller is in fact an automated method.
3. The labelling of an entity x_i is subject to individual or group human judgement. This is a frequent case in medicine. For instance, experts in electrocardiogram analysis seldom agree on the exact boundaries of signal patterns (Hughes et al. 2004).
4. Data encoding and communication problems may also be the source of label noise (ZHU and WU 2004; Angluin and Laird 1988).

A decrease in the accuracy of classification algorithms is, arguably, the most recognised outcome of the presence of label noise in data. However, this is far from being the only negative consequence of label noise (Frénay and Verleysen 2014). Label noise may also pose a threat to feature selection and weighting. Other consequences include changes in learning requirements (i.e., an algorithm may need more labelled data on the learning stage), increase in the complexity of models to learn, distortion of the labels' frequency, etc.

There are two main approaches to address the issue of label noise (Pechenizkiy et al. 2006). One approach is to devise algorithms within the supervised learning framework that are naturally robust to the presence of label noise.

Another approach, referred to as filter approach, is to detect noisy instances and eliminate them before the data set is used for learning. In this paper, we are particularly interested in the filter approach, albeit somewhat modified. Specifically, instead of simply detecting and removing noisy instances, we are going to detect and correct them.

We have opted to develop a filter approach to address label noise before the learning stage. Therefore, there are no restrictions on classification algorithms to be applied at a later stage.

2.2 K-Means Clustering

The k-means algorithm (Macqueen and et al. 1967; Ball and Hall 1967) is arguably the most popular clustering algorithm (Jain 2010; Mirkin 2016; Steinley 2006; Bock 2008). It aims to partition the n given entities from $\mathbf{X}$ into K disjoint clusters $S = \{S_1, S_2, ..., S_K\}$ by alternatingly minimising the following quadratic objective function:

$$W(S, C) = \sum_{k=1}^{K} \sum_{i \in S_k} \sum_{v=1}^{m} (x_{iv} - c_{kv})^2, \tag{1}$$

where each cluster $S_k \in S$ is represented by a centroid $c_k \in R^m$. Equation 1 applies the squared Euclidean distance, so that the optimal c_k are within-cluster means, $c_{kv} = |S_k|^{-1} \sum_{i \in S_k} x_{iv}$ for $v = 1, 2, ..., m, k = 1, 2, , K$. The method iteratively minimises (1) by following three simple steps: (i) select at random K entities from $\mathbf{X}$ and assign them to the centroids $c_1, c_2, ..., c_K$; (ii) assign x_i to the cluster S_k represented by the nearest centroid c_k, for $i = 1, 2, ..., n$; (iii) update each centroid $c_k \in C$ as the centre of S_k. Repeat steps (ii) and (iii) until convergence.

Various implementations of this algorithm can be found in many popular software packages, such as MATLAB (MATLAB 2013), R (R Core Team 2014), Scipy (Jones et al. 2001), Clustan (Wishart 1998), etc. However, the conventional k-means algorithm has the following properties:

1. The final clustering S is highly dependent on the initial centroids.
2. The number of clusters, K, needs to be known in advance.
3. Each feature has the same contribution to the clustering criterion, regardless of its actual degree of relevance.
4. Using the squared Euclidean distance biases the results towards clusters of spherical shape, regardless of the data distribution.

In order to address some of the above weaknesses, we have previously introduced the intelligent Minkowski weighted k-means algorithm (*imwk*-means) (De Amorim and Mirkin 2011). It uses the weighted Minkowski distance as follows:

$$d^p(x_i, c_k) = \sum_{v=1}^{m} w_{kv}^p |x_{iv} - c_{kv}|^p, \tag{2}$$

and minimises the following objective function as follows:

$$W(S, C, w) = \sum_{k=1}^{K} \sum_{i \in S_k} \sum_{v=1}^{m} w_{kv}^p |x_{iv} - c_{kv}|^p, \tag{3}$$

where p is a user-defined parameter representing the Minkowski exponent, and w_{kv} is the weight of feature v at cluster S_k. Our algorithm measures the dispersion of v at cluster S_k, $D_{kv} = \sum_{i \in S_k} |x_{iv} - c_{kv}|^p$, and sets the weights w_{kv} to be inversely proportional to D_{kv}:

$$w_{kv} = \left[\sum_{u=1}^{m} [D_{kv}/D_{ku}]^{1/(p-1)} \right]^{-1}. \tag{4}$$

This is subject to weights being non-negative and totalling to 1, $\sum_{k=1}^{K} w_{kv} = 1$, for $v = 1, 2, ..., m$, as well as clusters being crisp, i.e., $S_j \cap S_t = \emptyset$ for $j \neq t$. Note that if $D_{kv_1} < D_{kv_2}$, then $w_{kv_1} > w_{kv_2}$. The above is rather intuitive. It implies that any feature v may have different degrees of relevance in different clusters of S.

We first describe the Minkowski weighted k-means (*mwk*-means) as follows:

1. *Initial setting*. Set K and p to their user-defined values. Set $S \leftarrow \emptyset$, and $w_{kv} = 1/m$ for $k = 1, 2, ..., K$ and $v = 1, 2, ..., m$.
2. *Getting the initial centroids*. Select K entities from $\mathbf{X}$ at random and assign their values to the centroids $c_1, c_2, ..., c_K$.
3. *Cluster update*. Assign each entity $x_1, x_2, ..., x_n$ to the cluster S_k^* represented by the nearest c_k following (2). This generates the clustering $S^* = \{S_1^*, S_2^*, ..., S_K^*\}$. If $S = S^*$, go to step 6.
4. *Centroid update*. Update each $c_k \in C$ to the component-wise Minkowski centre of $x_i \in S_k$.
5. *Weight update*. Update each w_{kv} following Eq. 4. Set $S \leftarrow S^*$, then go to step 3.
6. *Output*. Output $S = \{S_1, S_2, ..., S_K\}$, $C = \{c_1, c_2, ..., c_K\}$, and w.

For a given value of the exponent p, the (*mwk*-means) algorithm should be carried out a number of times with random initialisations and that resulting clustering that minimises the objective function (3) should be selected. The *imwk*-means algorithm applies the concept of anomalous pattern (Mirkin 2016) to find good initial centroids and weights, and thus reduce the time complexity of (*mwk*-means), as described as below:

1. *Initial setting*. Set K and p to their user-defined values. Set c_c to the component-wise Minkowski centre of $x_1, x_2, ..., x_n$. Set $C_t \leftarrow \emptyset$, and each $w_{kv} = 1/m$.
2. *Get a tentative centroid*. Find the entity in $\mathbf{X}$ which is the farthest from c_c, as per (2), and assign its component-wise values to the tentative centroid c_t.
3. *Cluster*. Apply *mwk*-means using two centroids, c_t and c_c, generating the clustering $S = \{S_t, S_c\}$. During this clustering, do not allow c_c to move during the centroid update.
4. *Remove cluster*. Add c_t to C_t and respective weights to w. Remove each $x_i \in S_t$ from $\mathbf{X}$.
5. If $n > 0$, go to step 2.
6. Remove from C_t and w all entries other than those related to the K clusters with the highest cardinality.
7. Run *mwk*-means initialised with the centroids in C_t and weights in w.

The algorithms above require the calculation of the component-wise Minkowski centre. Let r be a column vector of the data matrix formed by $x_i \in S_k$. The Minkowski centre of r is the value μ minimising $\gamma_r(\mu) = \sum_j |r_j - \mu|^p$. Since we only consider $p > 1$, γ_r is a U-shaped curve with a minimum in the interval $[\min(r), \max(r)]$ (De Amorim and Mirkin 2011). We can find μ using standard methods for convex optimisation. Here, we initialise μ to the mean of r and improve it stepwise. At each step, we move μ by a fixed step of, say, 0.001, to the side in which γ_r decreases. No special method is needed if p is equal to one or two. In these cases, the Minkowski centre is equal to the median and the mean of r, respectively.

In fact, the choice of a clustering algorithm is not important in our context, as described in Section 3. We use the two algorithms described above because our experiments proved to be superior versions of the popular k-means algorithm. We direct interested readers to the numerous sources in the field, such as Jain (2010), Kaufman and Rousseeuw (1990), De Amorim and Makarenkov (2016), and Mirkin (2016), and references therein.

3 Method for Core Clustering

Consider a data matrix $\mathbf{X} \in \mathbb{R}^{nm}$ and respective labels $\mathbf{y} = (y_1, y_2, ..., y_n)$ containing a proportion ρ of mislabelled entities. Our main aim is to develop a method capable of reducing ρ. Since the exact value of ρ is unknown, we should not design a method based on the supervised learning framework. A method using the supervised learning framework would be unlikely to work if ρ was over 50%, precisely when we need to reduce ρ the most.

A clustering-based method will not be affected by a very high ρ. Clustering algorithms, such as those described in Section 2.2, are data-centric and can generate a clustering $S = \{S_1, S_2, ..., S_K\}$ without requiring labelled entities. If we could generate a perfect clustering S, then $\{x_i, x_j\} \subseteq S_k$ would mean that $y_i = y_j$. Unfortunately, there is no clustering algorithm that can find a perfect S in all cases. We can, however, produce a partial clustering $S' = \{S'_1, S'_2, ..., S'_K\}$ referring to only such entities $x_i \in S_k$ that are clustered most reliably. In our method, $S'_k \subseteq S_k$ is a core cluster. In this case, $S'_k \subseteq S_k$ for $k = 1, 2, ..., K$, leading to $|\bigcup_{k=1}^{K} S'_k| \leq n$.

The key, as one would expect, is to define a rule to decide whether an entity $x_i \in S_k$ should or should not be assigned to S'_k. Here, we introduce such a rule based on both cluster cohesion and separation.

Cohesion measures the homogeneity of entities within a cluster. Any k-means-based algorithm improves this measure by minimising the within-cluster distance between entities in some way. The criteria (1) and (3) are examples of that. Separation measures how distinct clusters are far away from each other. This measure is not directly improved by k-means, and there is a room for argument about whether improving cohesion will always lead to an improved separation.

Let us define first $a(x_i) = (|S_k| - 1)^{-1} \sum_{j \in \{S_k \setminus i\}} d(x_i, x_j)$. This is the average distance between x_i and all $x_j \in S_k$, $i \neq j$, measuring the cluster cohesion. Second, we define $b(x_i) = \min\{u_t \mid u_t = \sum_{j \in S_t} |S_t|^{-1} d(x_i, x_j), \forall S_t \in S \setminus S_k\}$. The cluster separation $b(x_i)$ is the lowest average distance of $x_i \in S_k$ to the entities of any other cluster.

Ideally, $x_i \in S_k$ will have a low value of $a(x_i)$, meaning that x_i is well matched to S_k, and a high value of $b(x_i)$, meaning that x_i is badly matched to any other cluster. We can now define as follows the Silhouette width index (Kaufman and Rousseeuw 1990):

$$\text{sil}(x_i) = \frac{b(x_i) - a(x_i)}{\max\{a(x_i), b(x_i)\}}. \tag{5}$$

Note that $-1 \leq \text{sil}(x_i) \leq 1$; $\text{sil}(x_i)$ equal to -1 implies $a(x_i) >> b(x_i)$, suggesting that x_i should be assigned to a different cluster, whereas $\text{sil}(x_i) = 1$ implies exactly the opposite. In our method, $x_i \in S_k$ is assigned to S'_k iff $\text{sil}(x_i) \geq \theta$, where θ is a preselected threshold.

Algorithm for Core Clustering and Relabelling

```
1. Obtain a clustering $S = \{S_1, S_2, ..., S_K\}$ and respective centroids $C = \{c_1, c_2, ..., c_K\}$
   by applying a clustering algorithm. Here, we experiment with k-means and imwk-means.
2. Set the threshold parameter $\theta = 0.5$ and $\alpha = 0.05$.
3. For each entity $x_i \in S_k$, assign $x_i$ to $S'_k$ iff $sil(x_i) \geq \theta, k = 1, 2, ..., K$.
4. If $|S| \neq |S'|$, set $\theta = \theta - \alpha \cdot n^{-1} \sum_{i=1}^{n} |sil(x_i)|$, and go to step 3.
5. Map each core $S'_k \in S'$ to one of the original labels in y using a confusion matrix, for
   $k = 1, 2, ..., K$.
6. Relabel all labels according to this mapping.
```

The Silhouette width index (5) is a cluster validity index that tends to perform well in comparative studies (see for instance Arbelaitz et al. (2013) and references therein). However, there is no index that is clearly superior to all others in all cases. For example, the total sum of the within-cluster distances between entities and centroids, given by Eq. 1 in the case of k-means, can be also used to select clustering solutions. With this in mind, we also experimented with the two other rules described as follows.

Distance to the Centre Each cluster $S_k \in S$ is represented by a centroid $c_k \in C$. An entity x_i is assigned to the cluster S_k whose centroid c_k is the nearest. One could claim that the closer $x_i \in S_k$ to c_k, the stronger the membership of x_i to S_k is. Under this principle if $\{x_i, x_j\} \subseteq S_k$ and $d(x_i, c_k) < d(x_j, c_k)$, we can say the degree of membership of x_i to S_k is higher than that of x_j for a distance function d. We can then assign a given entity $x_i \in S_k$ to S'_k iff $d(x_i, c_k) \leq n^{-1} \sum_{k=1}^{K} \sum_{i \in S_k} d(x_i, c_k)$. Unfortunately, this rule produced results below our expectations so we do not report them here.

Quartile Rule The distribution of distances between the entities and their respective cluster centroids may be the reason why the previous rule failed. This is likely to be the case if such a distribution is skewed. With this in mind, we decided to assign $x_i \in S_k$ to S'_k iff $d(x_i, c_k)$ was in the first quartile of this distribution. This rule is less sensitive to wide variations in the distances, but more restrictive as in the best-case scenario the method corrects a maximum of $0.25 \cdot n$ labels. Nevertheless, this rule also failed to produce decent classification results, so we do not report them here either.

It is worth noting that these two rules are based solely on cluster cohesion. This is very much the case of k-means itself, making us wonder if the fast and popular approach of minimising (1) hoping that it also increases (5) is the best way to cluster a data set.

4 Setting of Experiments

We studied the effectiveness of our method using synthetic and real-world data sets. Each cluster of the synthetic data sets originates from a Gaussian distribution whose covariance matrix is a diagonal matrix with the same random diagonal value σ^2 in the interval [0.5, 1.5]. The cardinality of each cluster is selected from a uniform distribution, with the constraint that no cluster could have less than 20 entities. Thus, our synthetic data sets correspond to the sets of spherical clusters with different cardinalities and spreads. Each of the centre's components is generated independently using the standard normal distribution $N(0, 1)$.

We generated 50 data sets for each of the following parameter configurations: (i) 1000x6-3, 1000 entities over 6 features partitioned into three clusters; (ii) 1000x12-6, 1000 entities over 12 features partitioned into 6 clusters; (iii) 1000x20-10, 1000 entities over 20 features partitioned into 10 clusters.

We also experimented with eight popular data sets from the UCI machine learning repository (Lichman 2013) (see Table 1). If a data set contains categorical features, we replace them by dummy zero-one features corresponding to individual categories. For a given categorical feature, v, only one of the generated L binary features has the value of one, i.e., that representing the original category of v. Afterwards, we standardise each dummy feature like any quantitiative feature (see formula 6 further on), by subtracting its average over the whole data set, that is, the relative frequency of the corresponding category. With this, frequent categories will have a lower value, and contribution to the clustering, than those that are infrequent.

Table 1 Real-world data sets tested in our simulations. These data sets were obtained from the UCI machine learning repository (Lichman 2013)

Data set	Number of features (m)		Number of labels (clusters)
	Before standardisation	After standardisation	
Australian credit approval	14	42	2
Breast cancer	9	9	2
Heart	13	25	2
Iris	4	4	3
Pima	8	8	2
Soya	35	58	4
Wine	13	13	3
Zoo	16	16	7

The method of enveloping the categories of categorical features clearly affects the dimensionality of data sets containing such features, as clearly demonstrated in Table 1.

We have standardised the numerical features as follows:

$$x_{iv} = \frac{x_{iv} - \overline{x_v}}{\max(x_v) - \min(x_v)}, \tag{6}$$

where $\overline{x_v} = n^{-1} \sum_{i=1}^{n} x_{iv}$, the grand mean. We chose to use (6) rather than the popular z-scoring because the latter standardisation favours unimodal distributions.

Unimodal features tend to have lower standard deviations than those that are multimodal. This leads the former to have a higher z-score, and contribution to the clustering, than the latter. For clustering, this is counter-intuitive as we are usually interested in the information contained in multimodal features.

In this paper, we conduct three sets of experiments, one for each of three values of the percentage of perturbed labels: 2.5, 5, and 10%. For each of these perturbation percentages, we consider the original set of labels and change that percentage of labels, rounded towards positive infinity, to an incorrect value. This incorrect value is chosen, uniformly random, from the existing set of labels for a given data set.

We do the above once for each of the synthetic data sets, and 20 times for each of the eight real-world data sets in Table 1. This leads to a total of 150 synthetic data sets and 160 real-world data sets, per percentage of perturbed labels, in our experiments.

We have run k-means 100 times and selected the final clustering as that minimising the objective function (1). In the case of *imwk*-means, we experimented with the values of p ranging from 1.1 to 5.0 with the step of 0.1, selecting the final clustering as that with the highest average silhouette over all entities in $\mathbf{X}$.

For comparison, we have also run the popular k-nearest neighbours (k-NN) (Friedman et al. 1977) algorithm for supervised learning. It assigns to y_i the mode of the labels of the k-nearest entities to x_i. We carried out this algorithm for $i = 1, 2, ..., n$ and $k \in \{5, 10, 15\}$.

Table 2 Average baseline ARI, and its standard deviation, between the corrupted labels and the correct labels for synthetic data sets

	ρ = 2.5%		ρ = 5%		ρ = 10%	
	ARI	Std	ARI	Std	ARI	Std
1000x6-3	0.9246	0.008	0.8511	0.014	0.7173	0.022
1000x12-6	0.9467	0.004	0.8938	0.005	0.7920	0.008
1000x20-10	0.9522	0.004	0.9054	0.006	0.8137	0.010

5 Results and Discussion

In our experiments, there are in fact three label vectors **y** for a given data matrix **X**. First, we have the correct labels **y** such that y_i represents the true label of x_i for $i = 1, 2, ..., n$. Second, we have the corrupted **y** containing a proportion ρ of mislabelled entities, so that there is a probability ρ for y_i to represent a wrong label for x_i, $i = 1, 2, ..., n$. Third, we have the updated **y** so that y_i was obtained by our core clustering method described in Section 4, fed in with **X** and the corrupted labels.

We are interested in measuring the similarity between the correct and corrupted labels, as well as between the correct and updated labels. We aim to see whether the similarity given by the latter pair is higher than that given by the former pair. We measure the similarity using the adjusted Rand index (ARI) (Hubert and Arabie 1985), a popular choice for comparing partitions.

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}, \tag{7}$$

where $n_{ij} = |S_i \cap S_j|$, $a_i = \sum_{j=1}^{K} |S_i \cap S_j|$ and $b_j = \sum_{i=1}^{K} |S_i \cap S_j|$.

Let us analyse first the results obtained when experimenting with the synthetic data. Table 2 presents the average values of ARI, and the corresponding standard deviations, between the corrupted and correct labels. These values represent our baseline that needs to be improved.

Table 3 reports the average change in ARI, and the standard deviation of this change, between the updated and correct labels. A positive value means an improvement on the baseline, whereas a negative value means the opposite. At ρ = 2.5%, we can see that *k*-NN produces very poor results, in some cases reducing the average baseline ARI by 0.2 or more. In this scenario, the improvement generated by our method is modest, reaching an average value of 0.0175 in the data sets with the highest number of features, when the clustering is done with *imwk*-means. At ρ = 5%, the average change in ARI produced by *k*-NN is still negative for all data sets, meaning this method tends to be less satisfactory at increased ρ. In contrast, our core clustering method leads to positive changes in ARI in all the cases. In the data sets with the highest number of features, this average improvement reaches 0.0422 when using *imwk*-means for clustering, resulting in a total average ARI of 0.9476 (after adding the values of Table 3 to those of Table 2). At ρ = 10%, the *k*-NN algorithm delivers a considerable improvement in two data set configurations, but this improvement is much lower than the improvement provided by our core clustering method.

Table 3 Average change in ARI between the updated labels and correct labels, as well as the standard deviation of ARI values. A positive value indicates an improvement on the baseline, while a negative value indicates the opposite

	ρ = 2.5%		ρ = 5%		ρ = 10%	
k-NN	ARI	Std	ARI	Std	ARI	Std
k = 5						
1000x6-3	– 0.2190	0.155	– 0.1536	0.157	– 0.0433	0.158
1000x12-6	– 0.1343	0.096	– 0.0856	0.096	0.0049	0.099
1000x20-10	– 0.0667	0.049	– 0.0223	0.048	0.0632	0.050
k = 10						
1000x6-3	– 0.2056	0.144	– 0.1346	0.146	– 0.0116	0.151
1000x12-6	– 0.1284	0.090	– 0.0763	0.090	0.0217	0.093
1000x20-10	– 0.0647	0.049	– 0.0200	0.048	0.0718	0.049
k = 15						
1000x6-3	– 0.2007	0.143	– 0.1301	0.144	– 0.0012	0.146
1000x12-6	– 0.1290	0.091	– 0.0763	0.092	0.0213	0.094
1000x20-10	– 0.0692	0.052	– 0.0227	0.050	0.0681	0.051
Our method						
k-means						
1000x6-3	0.0041	0.024	0.0240	0.034	0.0549	0.058
1000x12-6	0.0091	0.014	0.0247	0.023	0.0556	0.041
1000x20-10	0.0168	0.013	0.0397	0.018	0.0841	0.035
imwk-means						
1000x6-3	0.0047	0.025	0.0260	0.035	0.0595	0.059
1000x12-6	0.0015	0.034	0.0214	0.039	0.0591	0.052
1000x20-10	0.0175	0.014	0.0422	0.019	0.0941	0.034

Table 4 presents the average ARI, and standard deviation, between the corrupted and correct labels for the real-world data sets. The low values of the standard deviation were expected here. We corrupt the proportion ρ of labels 20 times, so the standard deviation of ARI is proportional to the number of labels in each data set. If the data set has only two labels, the standard deviation is zero.

Table 5 shows the average change in ARI, as well as the standard deviation of this change, for the real-world data sets. At ρ = 2.5%, we can see that the k-NN method fails to improve the average ARI in six to seven of the eight data sets (depending on the value of k). Our method produces better results, failing to produce improvements in only three data sets. More importantly, the worst-case scenario for our core clustering method produces an average reduction in ARI of 0.2088 (for Pima Indians data set, clustered with *k*-means), while the worst-case scenario for *k*-NN produces a reduction of 0.7062 for the same data set (at $k = 10$).

At ρ = 5%, the k-NN method fails to improve the average ARI in three to four of the eight data sets (depending on the value of k). Again our method produces better results, failing to produce improvements in two to three data sets (depending on the clustering algorithm).

Table 4 Average ARI and standard deviation between the corrupted labels and the correct labels on real-world data sets. A positive value indicates an improvement on the baseline, while a negative value indicates the opposite

	$\rho = 2.5\%$		$\rho = 5\%$		$\rho = 10\%$	
	ARI	Std	ARI	Std	ARI	Std
AustraCC	0.8982	0.000	0.8071	0.000	0.6395	0.000
Breast cancer	0.8987	0.000	0.8081	0.000	0.6370	0.000
Heart	0.8986	0.000	0.8026	0.000	0.6386	0.000
Iris	0.9210	0.000	0.8449	0.001	0.7204	0.002
Pima	0.8977	0.000	0.8056	0.000	0.6368	0.000
Soya	0.8860	0.011	0.8391	0.016	0.7370	0.021
Wine	0.9168	0.003	0.8540	0.004	0.7189	0.005
Zoo	0.9403	0.019	0.8818	0.018	0.7986	0.033

The worst-case scenario for both methods was again the Pima Indians data set. While k-NN produced an average reduction in ARI from 0.5808 to 0.6196 (depending on the value of k), our method produced an average reduction from 0.1626 to 0.1681 (depending on the clustering algorithm), only. At $\rho = 10\%$, again, the core clustering method was capable of improving the average ARI in more data sets than k-NN. Even when our method does fail to produce an improvement, its worst-case scenario is not as poor as that of k-NN.

The numerous experiments in this section do show that our core clustering method clearly outperforms the popular k-NN algorithm, especially when using *imwk*-means. Of course, there are some reasons for this rather positive performance. The k-NN algorithm assigns to x_i, the label of the majority of its k-neighbours, which takes solely (Euclidean) cohesion into account. Our core clustering method makes use of the silhouette width index (SW), with a suggested threshold of 0.5. Given an entity $x_i \in S_k$, the SW measures how similar x_i is to all other entities in S_k compared to the entities in other clusters. Hence, taking cohesion and separation into account. Also, our suggested threshold value ensures the clustering has a reasonable structure (Struyf et al. 1997). This becomes somewhat clearer when one notices that the SW index (Equation (5)) is only strictly positive if $b(x_i) > a(x_i)$, so that

$$\frac{b(x_i) - a(x_i)}{\max\{a(x_i), b(x_i)\}} \geq 0.5$$

implies $\max\{a(x_i), b(x_i)\} = b(x_i)$, leading to

$$a(x_i) \leq \frac{1}{2} b(x_i).$$

The average distance between x_i over all entities in its cluster, is at most half of the lowest average distance between x_i to the entities in any other cluster.

The use of *imwk*-means has two important implications. First, it removes the restrictive bias towards Gaussian clusters (or in the case of k-NN, Gaussian neighbours). Second, *imwk*-means (and by consequence our core clustering method) takes into account that even among relevant features there may be different degrees of relevance, which may be cluster-dependent. The superiority granted by feature weighting in clustering has already been extensively demonstrated in the literature (for details see the recent survey (De Amorim 2016), and references therein).

Table 5 Average change in ARI between the updated labels and correct labels, as well as the standard deviation

	$\rho = 2.5\%$		$\rho = 5\%$		$\rho = 10\%$	
k-NN	ARI	Std	ARI	Std	ARI	Std
k=5						
AustraCC	− 0.3830	0.014	− 0.2907	0.020	− 0.1680	0.029
Breast cancer	−0.0169	0.010	0.0644	0.012	0.1965	0.028
Heart	− 0.5407	0.021	−0.4573	0.028	−0.3232	0.031
Iris	− 0.0563	0.012	0.0196	0.015	0.1254	0.041
Pima	− 0.6857	0.011	− 0.5973	0.013	− 0.4582	0.015
Soya	0.1140	0.011	0.1565	0.028	0.2485	0.041
Wine	− 0.0653	0.014	− 0.0087	0.019	0.1187	0.034
Zoo	0.0112	0.020	0.0685	0.021	0.1460	0.041
k = 10						
AustraCC	− 0.3599	0.008	− 0.2747	0.016	− 0.1339	0.024
BreastCancer	− 0.0281	0.005	0.0565	0.008	0.2137	0.019
Heart	− 0.4945	0.015	− 0.3946	0.020	− 0.2498	0.027
Iris	− 0.0476	0.012	0.0286	0.019	0.1478	0.014
Pima	− 0.7062	0.009	− 0.6196	0.013	− 0.4617	0.018
Soya	0.1140	0.011	0.1609	0.016	0.2584	0.030
Wine	−0.0278	0.014	0.0329	0.016	0.1552	0.023
Zoo	−0.0130	0.021	0.0436	0.022	0.1210	0.042
k=15						
AustraCC	− 0.3702	0.010	− 0.2780	0.012	− 0.1223	0.019
Breast cancer	− 0.0281	0.005	0.0563	0.011	0.2263	0.010
Heart	−0.5118	0.011	− 0.4190	0.017	− 0.2636	0.025
Iris	− 0.0326	0.009	0.0509	0.017	0.1726	0.012
Pima	− 0.6753	0.009	− 0.5808	0.013	− 0.4248	0.015
Soya	0.0986	0.044	0.1375	0.059	0.2135	0.092
Wine	− 0.0003	0.016	0.0680	0.020	0.1792	0.029
Zoo	− 0.0450	0.016	0.0092	0.028	0.0941	0.046
Our method						
k-means						
AustraCC	− 0.0162	0.006	− 0.0047	0.010	0.0110	0.009
Breast cancer	0.0065	0.008	0.0850	0.012	0.2196	0.013
Heart	0.0043	0.015	0.0189	0.020	0.0241	0.015
Iris	0.0115	0.012	0.0668	0.015	0.1705	0.025
Pima	− 0.2088	0.010	− 0.1681	0.015	− 0.0891	0.012
Soya	0.0550	0.042	0.0963	0.042	0.1579	0.052
Wine	0.0404	0.020	0.0911	0.024	0.1490	0.023
Zoo	− 0.1672	0.061	− 0.1141	0.056	− 0.0755	0.052

Table 5 (continued)

	$\rho = 2.5\%$		$\rho = 5\%$		$\rho = 10\%$	
k-NN	ARI	Std	ARI	Std	ARI	Std
imwk-means						
AustraCC	– 0.0162	0.006	– 0.0047	0.010	0.0110	0.009
Breast cancer	0.0068	0.008	0.0839	0.013	0.2201	0.013
Heart	0.0170	0.010	0.0310	0.019	0.0314	0.017
Iris	0.0276	0.012	0.0825	0.012	0.1824	0.031
Pima	– 0.1978	0.011	– 0.1626	0.014	– 0.0892	0.015
Soya	0.0550	0.042	0.0963	0.042	0.1579	0.052
Wine	0.0396	0.020	0.0896	0.023	0.1459	0.024
Zoo	– 0.0051	0.019	0.0490	0.017	0.1355	0.033

Our core clustering method should produce positive results in data sets containing incorrect labels, and for which SW is a sound clustering validity index (CVI). A recent and comprehensive study (Arbelaitz et al. 2013) demonstrated there is no CVI having a clear advantage over all others in all scenarios; however, SW obtained the best results in many of them.

6 Conclusion

Supervised machine learning algorithms infer a function from data to respective labels. They then use this function to map labels to unlabelled entities. By consequence, the accuracy of such algorithms on unlabelled data depends heavily on the availability and correctness of labels.

Labelling entities manually is a laborious task. This is particularly true nowadays as the availability of data for most real-world problems has grown considerably. Humans do not always excel at repetitive tasks so such labelling is prone to errors. Disagreements are another source of mislabelling.

In this paper, we introduce a method capable of reducing the proportion of mislabelled entities. Our core clustering method builds upon the assumption that given a perfect clustering, two entities assigned to the same cluster should have the same label. Unfortunately, producing such a clustering is a non-trivial task in itself, one that has been subject to intensive research for many decades. Instead of trying to produce a full clustering in which each entity belongs to a cluster, our method produces a partial clustering of cluster cores. These cluster cores contain only the entities that have the most potential to be labelled correctly. We then use them to correct mislabelled entities.

We have run a number of computational experiments to validate our method empirically. These experiments include both synthetic and real-world data with different proportions of mislabelled entities. The obtained results suggest that our method does indeed reduce the proportion of mislabelled entities in data sets and thus could be used as a preprocessing data correction step of a supervised machine learning algorithm.

Acknowledgments BM thanks the Laboratory for Decision Choice and Analysis of the National Research University Higher School of Economics Moscow RF for partially supporting his work in the framework of the HSE University Basic Research Program funded by the Russian Academic Excellence Project '5-100'.

References

Angluin, D., & Laird, P. (1988). Learning from noisy examples. *Machine Learning*, *2*(4), 343–370.

Arbelaitz, O., Gurrutxaga, I., Muguerza, J., Pérez, J.M., Perona, I. (2013). An extensive comparative study of cluster validity indices. *Pattern Recognition*, *46*(1), 243–256.

Ball, G.H., & Hall, D.J. (1967). A clustering technique for summarizing multivariate data. *Behavioral Science*, *12*(2), 153–155.

Bock, H.-H. (2008). Origins and extensions of the k-means algorithm in cluster analysis. *Journal Electronique d'Histoire des Probabilités et de la Statistique (Electronic Journal for History of Probability and Statistics)*, *4*, 2.

Bouveyron, C., & Girard, S. (2009). Robust supervised classification with mixture models: learning from data with uncertain labels. *Pattern Recognition*, *42*(11), 2649–2658.

De Amorim, R.C. (2016). A survey on feature weighting based K-Means algorithms. *Journal of Classification*, *33*(2), 210–242. https://doi.org/10.1007/s00357-016-9208-4.

De Amorim, R.C., & Makarenkov, V. (2016). Applying subclustering and Lp distance in Weighted K-Means with distributed centroids. *Neurocomputing*, *173*, 700–707.

De Amorim, R.C., & Mirkin, B. (2011). Minkowski metric, feature weighting and anomalous cluster initializing in k-means clustering. *Pattern Recognition*, *45*, 3.

Frénay, B., & Verleysen, M. (2014). Classification in the presence of label noise: a survey. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(5), 845–869.

Friedman, J.H., Bentley, J.L., Finkel, R.A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, *3*(3), 209–226.

Grira, N., Crucianu, M., Boujemaa, N. (2004). Unsupervised and semisupervised clustering: a brief survey. A review of machine learning techniques for processing multimedia content, Report of the MUSCLE European Network of Excellence (FP6), pp. 1001–1030.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Hickey, R.J. (1996). Noise modelling and evaluating learning from examples. *Artificial Intelligence*, *82*(1), 157–179.

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, *2*(2), 193–218.

Hughes, N.P., Roberts, S.J., Tarassenko, L. (2004). Semi-supervised learning of probabilistic models for ECG segmentation. In: Engineering in Medicine and Biology Society, 2004. IEMBS'04. 26th Annual International Conference of the IEEE. Vol. 1. IEEE, pp. 434–437.

Jain, A., Jin, R., Chitta, R. (2014). Semi-supervised clustering. Handbook of Cluster Analysis, pp. 1–35.

Jain, A.K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, *31*(8), 651–666.

Jones, E., Oliphant, T., Peterson, P., et al. (2001). SciPy: Open source scientific tools for Python. [Online; accessed 2016-11-28]. http://www.scipy.org/.

Kaufman, L., & Rousseeuw, P.J. (1990). Finding groups in data: an introduction to cluster analysis. Vol. 39. Wiley Online Library.

Lichman, M. (2013). UCI Machine Learning Repository. http://archive.ics.uci.edu/ml.

Macqueen, J., et al. (1967). Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. vol. 1. 281–297. California, USA, pp. 14.

Maletic, J.I., & Marcus, A. (2000). Data cleansing: beyond integrity analysis. In: IQ. Citeseer, pp. 200–209.

MATLAB. (2013). *version 8.10.0 (R2013a). Natick*. Massachusetts: The MathWorks Inc.

Mirkin, B.G. (2016). *Clustering for data mining: a data recovery approach* Vol. 3. Boca Raton: CRC Press.

Orr, K. (1998). Data quality and systems theory. *Communications of the ACM*, *41*(2), 66–71.

Pechenizkiy, M., Tsymbal, A., Puuronen, S., Pechenizkiy, O. (2006). Class noise and supervised learning in medical domains: the effect of feature extraction. In: 19th IEEE symposium on computer-based medical systems (CBMS'06). IEEE, pp. 708–713.

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106.

R Core Team. (2014). *R: A language and environment for statistical computing*. Vienna: R Foundation for Statistical Computing. http://www.R-project.org.

Redman, T.C. (1998). The impact of poor data quality on the typical enterprise. *Communications of the ACM*, *41*(2), 79–82.

Saeys, Y., Inza, I., Larrañaga, P. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics*, *23*(19), 2507–2517.

Saáez, J.A., Galar, M., Luengo, J., Herrera, F. (2014). Analyzing the presence of noise in multi-class problems: alleviating its influence with the Onevs- One decomposition. *Knowledge and Information Systems*, *38*(1), 179–206.

Settles, B. (1648). *Active Learning Literature Survey*. Computer Sciences Technical Report: University of WisconsinMadison.

Steinley, D. (2006). K-means clustering: a half-century synthesis. *British Journal of Mathematical and Statistical Psychology*, *59*(1), 1–34.

Struyf, A., Hubert, M., Rousseeuw, P., et al. (1997). Clustering in an object-oriented environment. *Journal of Statistical Software*, *1*(4), 1–30.

Wishart, D. (1998). Clustan. http://www.clustan.com/ (visited on 11/28/2016).

ZHU, X. (2006). Semi-supervised learning literature survey. Computer Science. *University of Wisconsin-Madison*, *2*(3), 4.

ZHU, X., & WU, X. (2004). Class noise vs. attribute noise: a quantitative study. *Artificial Intelligence Review*, *22*(3), 177–210.