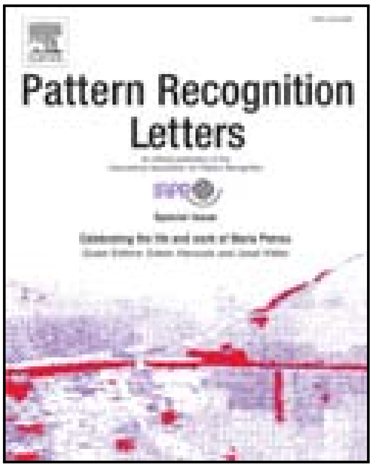
Journal Pre-proof



Unsupervised feature selection for large data sets

Renato Cordeiro de Amorim

PII: S0167-8655(18)30496-3
DOI: https://doi.org/10.1016/j.patrec.2019.08.017
Reference: PATREC 7606

To appear in: *Pattern Recognition Letters*

Received date: 22 August 2018
Revised date: 10 June 2019
Accepted date: 20 August 2019

Please cite this article as: Renato Cordeiro de Amorim, Unsupervised feature selection for large data sets, *Pattern Recognition Letters* (2019), doi: https://doi.org/10.1016/j.patrec.2019.08.017

Highlights

- We propose a novel clustering-based unsupervised feature selection algorithm
- This is possibly the first such algorithm not to require access to the whole data
- Our algorithm is particularly suitable for very large data sets

Pattern Recognition Letters
journal homepage: www.elsevier.com

Unsupervised feature selection for large data sets

Renato Cordeiro de Amorim

School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK.

ABSTRACT

The last decade saw a considerable increase in the availability of data. Unfortunately, this increase was overshadowed by various technical difficulties that arise when analysing large data sets. These include long processing times, large requirements for data storage, and other technical issues related to the analysis of high-dimensional data sets. By consequence, reducing the cardinality of data sets (with minimum information loss) has become of interest to virtually any data scientist. Many feature selection algorithms have been introduced in the literature, however, there are two main issues with these. First, the vast majority of such algorithms require labelled samples to learn from. One should note it is often too expensive to label a meaningful amount of data, particularly when dealing with large data sets. Second, these algorithms were not designed to deal with the volume of data we have nowadays. This paper introduces a novel unsupervised feature selection algorithm designed specifically to deal with large data sets. Our experiments demonstrate the superiority of our method.

1. Introduction

Recent technological advances and new government policies, such as transparency and freedom of information, have substantially increased the availability of data. This ever increasing growth rate of available data (which has exceeded that of Moore's Law (Chen and Zhang, 2014)) and the possibility of merging data sets (often originated at different sources) have had a considerable impact on the size of data sets, and by consequence their complexity. However, in the world of big data quantity does not mean quality (Kaisler et al., 2013). Thus, such data sets may very well contain irrelevant features.

The word feature refers to a measurement used to describe entities in a data set, in the literature this is sometimes called a variable. Removing irrelevant features is of interest to virtually any data analyst, particularly when dealing with large data sets. A reduction in the number of features usually leads to shorter processing times, a lower requirement in terms of memory to hold a given data set, and may help to avoid problems related to overfitting and *the curse of dimensionality*. However, direct evaluation of each possible subset of features tends to be infeasible. Given a data set X composed of N entities each described over V features there would be 2^V possible subsets of features, such evaluation becomes a NP-Hard problem as V grows.

Feature selection is certainly one of the main areas of research in machine learning, however, most of the work focuses on supervised methods (see for instance (Guyon and Elisseeff, 2003; Chandrashekar and Sahin, 2014), and references therein). These are methods that require a labelled sample of meaningful size to learn from. The validation of such algorithms often makes use of either five or ten fold cross-validation, in both cases this means there is an expectation there would be considerably more labelled samples than unlabelled. This seems to contradict the fact that data sets are growing in size and by consequence labelling a large sample of entities is becoming more and more expensive. We see unsupervised feature selection algorithms as probably the most viable option for real-world scenarios given these algorithms are data-centric and do not require a labelled sample to learn from. However, even among these there is at least one major issue. Existing unsupervised feature selection algorithms require access to the whole data set (see for instance, (Mitra et al., 2002; Cai et al., 2010; Kim et al., 2000), and references therein). This requirement can be problematic as modern data sets may occupy more memory than what is usually available.

In this paper we introduce a novel unsupervised feature selection algorithm. Our method is the first, to our knowledge, to be designed specifically for data sets that are large and may not fit in the main memory of a computer. We validate our method through various experiments and conclude that it does reduce the number of features in a data set with low (if any) information loss.

[**]Corresponding author: Tel.: +44 (0)1206 872895;
e-mail: r.amorim@essex.ac.uk (Renato Cordeiro de Amorim)

2. Related work

There are two main areas of research related to the work we present in this paper, (i) unsupervised learning algorithms in general and (ii) unsupervised feature selection. Section 2.1 provides an overview of related clustering algorithms, these are unsupervised algorithms directly related to our contribution (for details see Section 3). Section 2.2 provides an overview of unsupervised feature selection algorithms.

2.1. Clustering algorithms

The literature in data clustering presents a number of algorithms, among these k-means (Ball and Hall, 1967; MacQueen, 1967) is arguably the most popular (Hans-Hermann, 2008; Jain, 2010; Mirkin, 2012; Steinley, 2006). K-means groups the N entities in a data set X into K homogeneous clusters $S = \{S_1, S_2, ..., S_K\}$ so that $S_k \cap S_j = \emptyset$ for $k, j = 1, 2, ..., K$ and $k \neq j$, leading to $| \cup_{k=1}^{K} S_k| = N$. Each cluster $S_k \in S$ is represented by a centroid $z_k \in Z$, often called the prototype of S_k. This algorithm aims to locate homogeneous clusters by minimising the within-cluster sum of squares given by

$$P(S, Z) = \sum_{k=1}^{K} \sum_{x_i \in S_k} \sum_{v=1}^{V} (x_{iv} - z_{kv})^2, \tag{1}$$

where V represents the number of features describing each entity $x_i \in X$. The minimisation of (1) follows three simple steps: (i) select K entities from X at random and copy their values to $z_1, z_2, ..., z_K$; (ii) assign each entity $x_i \in X$ to the cluster S_k represented by the centroid z_k which is the nearest to x_i; (iii) update each centroid $z_k \in Z$ to the centre of its cluster $S_k \in S$. Since (1) applies the squared Euclidean distance, the centre of S_k is given by $z_{kv} = |S_k|^{-1} \sum_{x_i \in S_k} x_{iv}$ for $v = 1, 2, ..., V$.

K-means is a rather popular algorithm implemented in a number of software packages often used for data analysis, such as: Clustan, Scipy, R, and MATLAB (Wishart, 1998; Jones et al., 2001; R Core Team, 2014; MATLAB, 2013). However popular k-means does have known weaknesses, for instance: (i) it requires the number of clusters, K, to be known beforehand; (ii) it may get trapped in local minima; (iii) it assumes the degree of relevance of each feature is the same; (iv) it cannot cluster large data sets in an acceptable speed. In this paper, we are particularly interested in adapting a clustering algorithm so that it can be used for unsupervised feature selection on large data sets, which is related to weaknesses (iii) and (iv).

The availability of data has increased considerably in the last decade, and surely the internet has been one of the major forces behind this. The impact of this increase is so high that it has lead to the popularity of the term big data. The availability of data is indeed increasing and so is the expectation of users in terms of processing time. Users tend to expect very low latencies, meaning a given application is expected to process large amounts of data in less and less time. Web-Scale k-means (WSk-means) (Sculley, 2010) was introduced to address these extreme requirements of latency and scalability. This algorithm addresses these requirements by applying a sampling approach to clustering taking only a couple of extra parameters in relation to k-means: the quantity of uniformly random samples (T), and the cardinality of each sample (b).

Web-Scale *k*-means

```
1. Select $K$ entities from $X$ at random, and copy their values to $z_1, z_2, ..., z_K$.
2. Set each counter $c_1, c_2, ..., c_K$ to zero.
3. Repeat $T$ times:
   3..1 $X_t \leftarrow b$ entities selected at random from $X$.
   3..2 For each $x_i \in X_t$
        i. Find $z_k$, the closest centroid to $x_i$.
        ii. Set $c_k = c_k + 1$.
        iii. Set $\eta = c_k^{-1}$.
        iv. Set $z_{kv} = (1 - \eta)z_{kv} + \eta x_{iv}$ for $v = 1, 2, ..., V$.
```

WSk-means allows one to balance the trade-off between scalability and latency by tuning the parameters T and b. This is a popular and effective algorithm, but it considers all features regardless of their actual relevance. Generally speaking, it is fair to assume that large data sets are likely to contain features with different degrees of relevancy. There has been considerable research in feature relevance in clustering (see for instance (de Amorim, 2016; Huang et al., 2005; Chan et al., 2004), and references therein), and we have had considerable success with the Minkowski weighted k-means (MWk-means) (de Amorim and Mirkin, 2012). This algorithm applies a weighted version of the Minkowski distance

$$d_p(x_i, z_k) = \sum_{v=1}^{V} w_{kv}^p |x_{iv} - z_{kv}|^p, \tag{2}$$

where the Minkowski exponent p is a user-defined parameter, and w_{kv} represents the weight (ie. degree of relevance) of feature v at cluster S_k. The use of the Minkowski distance allows the distance bias to be adjusted with the help of p. At $p = 1$, $p = 2$, and $p \rightarrow \infty$ the cluster shape will be biased towards rhombus, circle, and square, respectively. Clearly, interpolations of these shapes can be reached with intermediate values of p. The distance (2) leads to the MWk-means criterion

$$P(S, Z, w) = \sum_{k=1}^{K} \sum_{x_i \in S_k} \sum_{v=1}^{V} w_{kv}^p |x_{iv} - z_{kv}|^p, \tag{3}$$

subject to

$$\begin{cases} S_k \cap S_j = \emptyset \text{ for } k, j = 1, 2, ..., K \text{ and } k \neq j, \\ w_{kv} \geq 0 \text{ for } k = 1, 2, ..., K \text{ and } v = 1, 2, ..., V, \\ \sum_{v=1}^{V} w_{kv} = 1 \text{ for } k = 1, 2, ..., K, \\ p \geq 1. \end{cases} \tag{4}$$

and minimised iff

$$w_{kv} = \left(\sum_{u=1}^{V} [D_{kv}/D_{ku}]^{1/(p-1)} \right)^{-1}, \tag{5}$$

where the dispersion of a given feature v at a cluster S_k is given by $D_{kv} = \sum_{x_i \in S_k} |x_{iv} - z_{kv}|^p$. To avoid divisions by zero one can add the average over all dispersions to each D_{kv}. We formalise MWk-means as follows:

Minkowski weighted *k*-means

```
1. Select $K$ entities from $X$ at random, and copy their values
   to $z_1, z_2, ..., z_K$. Set each $w_{kv}$ to $V^{-1}$ and select a value for $p$.
2. Assign each entity $x_i \in X$ to the cluster $S_k \in S$ whose
   centroid $z_k$ is the nearest to $x_i$, as per (2).
3. Update each centroid $z_i \in Z$ to the centre of its cluster $S_k$.
4. Update each weight $w_{kv}$ as per (5).
```

MW*k*-means applies the Minkowski distance, by consequence the centroid z_k of cluster $S_k \in S$ is only given by the component-wise mean of $x_i \in S_k$ if $p = 2$. At $p = 1$ and $p \rightarrow \infty$ the centre is given by the median and mid-range, respectively. In all other cases one can apply a gradient descent method. At $p \geq 1$ we have a U-shaped curve $\gamma_v(\mu) = \sum_{x_i \in S_k} |x_{iv} - \mu|^p$, with a minimum in the interval $[min(x_v), max(x_v)]$ (de Amorim and Mirkin, 2012; de Amorim and Hennig, 2015). The minimum μ for a feature v at cluster S_k can be found by first setting $\mu_{kv} = |S_k|^{-1} \sum_{x_i \in S_k} x_{iv}$, and then improve it stepwise by a fixed amount (say, 0.001) per step to the side in which γ_v is reduced. In the MW*k*-means model the weight w_{kv} represents the degree of relevance of a feature v at cluster $S_k \in S$, thus supporting two intuitive and related ideas. First, a given feature v may have different degrees of relevance at each cluster $S_k \in S$. Second, even among relevant features there may be different degrees of feature relevance. The literature has shown that applying feature weights does increase cluster recovery (de Amorim and Mirkin, 2012; de Amorim, 2016; Chan et al., 2004; Huang et al., 2005) and if these weights are used as feature rescaling factors it increases the likelihood of clustering validity indexes to return the true number of clusters in a data set (de Amorim and Hennig, 2015).

2.2. *Feature selection*

Modelling the degree of relevance of features using feature weighting can certainly improve cluster recovery. However, in some scenarios (particularly those related to large data sets) it may be better to simply remove some of the features from a data set. For instance, unsupervised feature selection may make it possible to analyse a data set that was previously thought to be too large to be analysed. Of course, for this to happen we need an unsupervised feature selection algorithm capable of dealing with large data sets.

Feature Selecting using Feature Similarity (FSFS) (Mitra et al., 2002) is arguably the most popular unsupervised feature selection algorithm (and certainly the most cited). This algorithm applies *k*–nearest neighbours (Altman, 1992) and a feature similarity measure given by

$$2\lambda_2(v_1, v_2) = (var(v_1) + var(v_2) - \sqrt{(var(v_1) + var(v_2))^2 - 4var(v_1)var(v_2)(1 - \rho(v_1, v_2)^2)}, \quad (6)$$

where $var()$ represents the variance of a feature, $\rho(v_1, v_2) = \frac{cov(v_1, v_2)}{\sqrt{var(x_1)var(v_2)}}$, and $cov()$ is the covariance. $\lambda_2(v_1, v_2)$ is zero if the features v_1 and v_2 are linearly independent, $\lambda_2(v_1, v_2)$ increases as the dependency between v_1 and v_2 decreases. FSFS determines the set of maximally independent features by discarding those considered to be redundant. This algorithm has a user-defined parameter which is approximately equal to the number of features removed by the algorithm.

Traditionally, unsupervised feature selection algorithms compute feature specific scores. They then use these scores to decided which features should be kept and which should be removed from a given data set. Such approaches neglect the possible correlation between different features. With this in mind Cai, Zhang and He proposed the Multi-Cluster Feature Selection (MCFS) (Cai et al., 2010). Their method has been inspired by developments on manifold learning and L1-regularized models, and attempts to select those features such that the multi-cluster structure of the data can be best preserved with a single user-defined parameter. This algorithm has two parameters that can be fine-tuned by the user (the number of eigenfunctions used, and the number of neighbours for a *k*-NN graph) as well as a parameter that must be defined: the number of features to be selected.

Previously, we introduced Feature Selection via Feature Weighting (Panday et al., 2018). This method took into account that under the MW*k*-Means framework (see Section 2.1) a very low $\sum_{k=1}^{K} w_{kv}$ essentially means feature v will not contribute to the clustering. However, in this case v would still be used in computations (consuming CPU time) and occupy memory. FSFW removes any feature v for which $K^{-1} \sum_{k=1}^{K} w_{kv} < V^{-1}$, or, $\max\limits_{1 \leq k \leq K} w_{kv} < V^{-1}$.

Unfortunately, FSFS, MCFS and FSFW follow a common pattern in unsupervised feature selection: current algorithms require access to the whole data set (for other examples see (Guyon and Elisseeff, 2003; Chandrashekar and Sahin, 2014; Kim et al., 2000), and references therein). Thus, if a data set is large enough not to fit in the memory of a computer these algorithms cannot be used.

3. Web-Scale Minkowski weighted *k*-means and feature selection

In this section we introduce the Web-Scale Minkowski weighted *k*-means (WSMW*k*-means). This is, to our knowledge, the first clustering algorithm capable of producing and applying feature weights for large data sets. These feature weights model the degree of relevance of each feature at each cluster, and we show that these weights can be used for unsupervised feature selection in large data sets. WSMW*k*-means minimises the MW*k*-means criterion (3) following a framework inspired by that of WS*k*-means (for details see Section 2).

Web-Scale Minkowski weighted *k*-means ($p = 2$)

```
1. Select $K$ entities from $X$ at random, and copy their values
   to $z_1, z_2, ..., z_K$. Set each $w_{kv}$ to $V^{-1}$.
2. Set each counter $c_1, c_2, ..., c_K$ to zero.
3. Repeat from $t = 1$ to $t = T$:
   3..1 $X_t \leftarrow b$ entities selected at random from $X$.
   3..2 for each $x_i \in X_t$
          i. Find $z_k$, the closest centroid to $x_i$ using (2).
         ii. Assign $x_i$ to $S_k$, the cluster represented by $z_k$.
        iii. Set $c_k = c_k + 1$.
```

```
            iv. Set $\eta = c_k^{-1}$.
            v. $z_{kv} = (1-\eta)z_{kv} + \eta x_{iv}$ for $v = 1, 2, ..., V$.
      3..3 Set $\lambda = t^{-1}$.
      3..4 Given $X_t$, $S$, and $Z$, generate a new set of weights $w'$ following Equation (5).
      3..5 Set each weight $w_{kv} = (1-\lambda)w_{kv} + \lambda w'_{kv}$
   4. Assign each $x_i \in X$ to the cluster $S_k$ represented by the centroid $z_k$ which is the nearest to $x_i$ as per Equation (2),
```

Feature weighting can be seen as a generalisation of feature selection. This is straightforward if one interprets these as allowing w_v (the degree of relevance of feature v) to be $0 \leq w_v \leq 1$ or $w_v \in \{0, 1\}$, respectively. Clearly, it is possible to move from feature weighting to feature selection if one is able to find a suitable threshold θ. WSMWk-means was designed to clusters data sets in which a feature v may be relevant to a single cluster. It does so by allowing v to have K weights $w_{1v}, w_{2v}, ...w_{Kv}$. With this in mind we have decided to set $\theta = V^{-1}$, and compare it against $\max\limits_{1\leq k\leq K} w_{kv}$, so that

$$w_v = \begin{cases} 1, \max\limits_{1\leq k\leq K} w_{kv} \geq V^{-1} \\ 0, \max\limits_{1\leq k\leq K} w_{kv} < V^{-1}. \end{cases} \tag{7}$$

The rule above deselects v iff $\max\limits_{1\leq k\leq K} w_{kv} < V^{-1}$, that is

$$\max_{1\leq k\leq K} \left(\sum_{u=1}^{V} [D_{kv}/D_{ku}]^{1/(p-1)} \right)^{-1} < V^{-1}$$

If we consider $p = 2$ we have

$$\max_{1\leq k\leq K} \sum_{u=1}^{V} \frac{D_{kv}}{D_{ku}} > V,$$

leading to

$$\max_{1\leq k\leq K} \sum_{u=1}^{V} \frac{D_{kv} - D_{ku}}{D_{ku}} > 0.$$

The above can only be true if $D_{kv} > V^{-1} \sum_{u=1}^{V} D_{ku}$ for at least one $S_k \in S$. This implies $D_{kv} > D_{ku}$ for at least one feature u in at least one cluster $S_k \in S$. That is, at least once the dispersion of v is higher than that of u in the same cluster S_k, implying the degree of relevance of feature u is higher than that of v ($w_{ku} > w_{kv}$). Hence, the situation $\max\limits_{1\leq k\leq K} w_{kv} < V^{-1}$ only happens if there is a more relevant feature than v for each cluster $S_k \in S$.

In this section we introduced WSMWk−means, an algorithm that minimises the MWk-means criterion (3) following a framework inspired by that of WSk-means. We find this to be a particularly interesting route to design an unsupervised feature selection algorithm because both WSk-means and MWk-means have considerable strengths that complement each other. WSk-means is arguably the most popular algorithm for clustering very large data sets. However, it considers all features in a data set regardless of their actual relevance. This means that features are taken into account even if they are not relevant at all, which is counter-productive when dealing with large data sets. On the other hand, assigning a within-cluster degree of relevance to any given feature (w_{kv}) is precisely the strength of MWk-means. By combining and extending these two algorithms we were able to calculate the relevance of features in data sets that were prohibitively large, culminating in a novel unsupervised feature selection algorithm designed specifically for large data sets.

4. Set up of experiments

Our method has a mathematically sensible approach for unsupervised feature selection in large data sets (for details see section 3). Of course, we also provide an empirical validation and describe the set up of our experiments in this section. The data sets we experiment with come from the popular UCI machine learning repository (Dheeru and Karra Taniskidou, 2017). In terms of data pre-processing we first removed any feature with a range of zero. We then replaced any missing value x_{iv} by the average of v over the whole data set, $\bar{x}_v$. This was possible because we had missing values solely in numerical features. The Poker hand data set contains five categorical features. We dealt with these by replacing each feature v with V' categories by V' binary features, in which only the binary feature related to the original category was set to one. Finally, we standardised each feature (including those that were binary) of a given data set using

$$x_{iv} = \frac{x_{iv} - \bar{x}_v}{range(x_v)}. \tag{8}$$

We chose (8) rather than the z-score because the latter favours features under a unimodal distribution. Such features tend to have a lower standard deviation than those that are multimodal. By consequence their z-score is higher, leading to a higher contribution to the clustering from unimodal features than from multimodal features. However, multimodal features are those that are usually of particular interest during clustering. The last adjustment on our original data sets was to create labels for the Online News popularity data set as these were not available. Rather arbitrarily we decided to assign six clusters to this data set, based on the number of shares (the target for this data set). To do so we used bins of 1,000, and assigned entities with 5,000 or more shares to cluster six.

Unfortunately, we do not know which features are relevant for each of the data sets we experiment with. Our experiments address this issue in two ways. First, we added noise features to each of our original data sets (see details in Table 1). Here, a noise feature is one composed entirely of within-domain uniformly random values. For each original data set we generated two other data sets by adding $\lceil V \times 0.1 \rceil$ and $\lceil V \times 0.2 \rceil$ noise features, respectively. We opened an exception for the Skin Segmentation data set and added one and two noise features respectively, otherwise we would be adding a single noise feature in both cases. If a feature selection method removes one of these noise features, we know it is removing a feature that should be removed. Second, if a method removes an original feature (that is, a feature that originally belonged to the data set) that does not necessarily mean it is removing a relevant feature. We do not know which of the original features are actually relevant. We address this issue by computing the average Entropy over the features of a data set, before and after feature selection.

Feature selection using Feature Similarity (FSFS, for details see Section 2, or (Mitra et al., 2002)) is arguably the most popular unsupervised feature selection algorithm there is, and certainly the most cited. Thus, we have decided to use it as a benchmark. We considered Multi-Cluster Feature Selection (MCFS, for details see Section 2, or (Cai et al., 2010)) because this is also a popular algorithm with hundreds of citations but our initial experiments deemed it unsuitable for large data sets (after three days processing the Record Linkage data set on an Intel Xeon 3.2 GHz with 64GB of RAM using the source code provided by the original authors, we gave up).

One could argue our experiments comparing FSFS to our method are biased towards the former. FSFS had access to the whole of the data, while our method requires access only to a fraction of each data set. Our method works this way so that it can be used in very large data sets that may or may not fit in the memory of a computer, unlike FSFS. Also, FSFS requires a user-defined parameter which is roughly the number of features to be removed. In real-world scenarios a user is unlikely to know this number, but we have supplied FSFS with the correct number for all data sets containing noise features. Since such features have no meaningful structure (they are uniformly random values) FSFS should remove the noise features. For data sets with no noise features we set this parameter to $\lceil 0.05 \times V \rceil$.

The data sets we selected for our experiments are indeed large, but they all easily fit in the memory of a modern computer. They are large enough to be relevant, but not so large that we can not experiment with FSFS or measure the entropy of features. The latter aims to establish whether FSFS and our method are removing solely irrelevant features by measuring average entropy of data sets before and after feature selection. In short, we did not run experiments with even larger data sets solely because we need a benchmark.

Table 1. The list of data sets used in our experiments. The column 'Features' includes noise features if there are any. The data sets were obtained from the popular UCI machine learning repository (Dheeru and Karra Taniskidou, 2017).

	Entities	Clusters	Features	Noise
Data sets	N	K	V	features
Covertype	581,012	7	54	0
MoCap Hand Postures	78,095	5	36	0
IDA 2016 Challenge	76,000	2	169	0
Online News Popularity	39,644	6	58	0
Poker Hand	1,000,000	10	25	0
Record Linkage Comparison P.	5,749,132	2	9	0
Sensorless Drive Diagnosis	58,509	11	48	0
Skin Segmentation	245,057	2	3	0
+ approx. 10% noise features				
Covertype	581,012	7	60	6
MoCap Hand Postures	78,095	5	40	4
IDA 2016 Challenge	76,000	2	186	17
Online News Popularity	39,644	6	64	6
Poker Hand	1,000,000	10	28	3
Record Linkage Comparison P.	5,749,132	2	10	1
Sensorless Drive Diagnosis	58,509	11	53	5
Skin Segmentation	245,057	2	4	1
+ approx. 20% noise features				
Covertype	581,012	7	65	11
MoCap Hand Postures	78,095	5	44	8
IDA 2016 Challenge	76,000	2	203	34
Online News Popularity	39,644	6	70	12
Poker Hand	1,000,000	10	30	5
Record Linkage Comparison P.	5,749,132	2	11	2
Sensorless Drive Diagnosis	58,509	11	58	10
Skin Segmentation	245,057	2	5	2

5. Results and discussion

In our experiments we ran WSMW*k*-means 100 times, and arbitrarily set $b = \sqrt{N} * K$. An optimal value for b should take into account the size of a given data set, the amount of free memory in the computer running WSMW*k*-means, as well as the speed of that computer. By consequence the optimal value for b is clearly problem dependent. In this paper we set $b = \sqrt{N} * K$ solely to have a single rule that we could use in all of the data sets we experiment with.

Table 2 shows the average frequency features were selected in our first set of experiments. These experiments compare FSFS, analysing the whole of the data and feed with the correct number of noise features, against the average over 100 runs of WSMW*k*-means. Let us first consider the experiments in which $T = 5$. These clearly show that in the vast majority of cases noise features were selected with a much lower average frequency by WSMW*k*-means than by FSFS. The only meaningful exception was the Poker Hand data set. In this WSMW*k*-means was unable to remove the noise features while FSFS selected them with an average frequency of 0.333 and 0.200 for the data sets with 10% and 20% of noise, respectively.

On the Record Linkage data set WSMW*k*-means (still with $T = 5$) selected noise features with an average frequency of 0.14 and 0.055 for the data sets with 10% and 20% of noise features, respectively. In the same data set FSFS was able to remove all noise features, however, one should note WSMW*k*-reached a very good result after analysing only 0.4% of the data. Reaching the best results in the majority of cases, using only a fraction of the data, is the major advantage provided by our method. For instance, we have removed nearly all noise features from the CoverType and IDA 2016 data sets processing less than 5% of the available data.

Table 2 also shows the average frequency features were selected in our second set of experiments. These are similar to our first set of experiments, the only difference is that now we set $T = 10$ for WSMW*k*-means. In this case our method processes double the amount of data than at $T = 5$, leading to a small improvement in the results. This is particularly true for the data sets to which we added 10% noise features. We can see our method now select a noise feature with an average frequency of 0.03 and 0.05 for the data sets to which we added 10% and 20% of noise features, respectively. This is an excellent result, particularly if we take into account our method used only 0.8% of the data. In general we can see WSMW*k*-means clearly outperforms FSFS.

Our previous experiments show that in most cases our method does remove noise features from data sets, and in this

Table 2. The average frequency of features selected by FSFS (feed with the correct number of noise features), and the average over 100 runs of WSMW*k*-means. Under 'Orig' we show the average frequency of original features being selected(those that originally belonged to the data set). Under 'Noise' we show the average frequency of noisy features being selected (those composed entirely of uniformly random values). The proportion of data used is given by $b \times T/N$ for WSMW*k*-means, and the whole data set for FSFS.

	No noise	10% of noise		20% of noise		Proportion of data used
		Orig	Noise	Orig	Noise	
WSMW*k*-means (T = 5)						
CoverType	0.915	0.957	0.002	0.971	**0.000**	0.046
Hand Postures	0.786	0.828	0.025	0.941	**0.001**	0.090
IDA 2016	0.866	0.931	**0.000**	0.931	**0.000**	0.036
Online News Pop	0.819	0.844	**0.002**	0.862	**0.000**	0.151
Poker Hand	0.945	0.934	1.000	0.932	1.000	0.050
Record Linkage	0.641	0.644	0.140	0.644	0.055	0.004
Sensorless Drive	0.773	0.826	**0.000**	0.939	**0.000**	0.227
Skin Segmentation	0.640	0.637	**0.170**	0.713	0.155	0.020
WSMW*k*-means (T = 10)						
CoverType	0.912	0.958	**0.000**	0.969	**0.000**	0.092
Hand Postures	0.753	0.821	**0.018**	0.928	0.012	0.179
IDA 2016	0.885	0.930	**0.000**	0.941	**0.000**	0.073
Online News Pop	0.816	0.843	**0.002**	0.858	**0.000**	0.303
Poker Hand	0.947	0.937	1.000	0.930	1.000	0.100
Record Linkage	0.624	0.654	**0.030**	0.649	**0.050**	0.008
Sensorless Drive	0.754	0.813	**0.000**	0.944	**0.000**	0.455
Skin Segmentation	0.657	0.607	**0.170**	0.700	**0.135**	0.040
FSFS						
CoverType	0.944	0.963	0.333	0.944	0.273	1.000
Hand Postures	0.944	0.972	0.250	0.972	0.125	1.000
IDA 2016	0.935	0.888	1.000	0.787	1.000	1.000
Online News Pop	0.948	0.914	0.833	0.793	1.000	1.000
Poker Hand	0.920	0.880	**0.333**	0.960	**0.200**	1.000
Record Linkage	0.889	1.000	0.000	1.000	0.000	1.000
Sensorless Drive	0.917	0.917	0.600	0.896	0.400	1.000
Skin Segmentation	0.667	0.667	1.000	0.333	1.000	1.000

respect it outperforms FSFS. These experiments also show both methods to remove, usually in a lower proportion, some of the features that originally belonged to each data set. For instance, in the CoverType data set with no noise features our method selects a feature 91.5% of times ($T = 5$) and FSFS does so 94.4% of times. There are two points we must make now. First, the fact FSFS tends to select more original features than our method is no indication of its capabilities. We have no information regarding the relevance of the original features, there may be irrelevant features among those. Second and probably more important, FSFS has a user-defined parameter that is roughly the number of features to be removed. In the real-world a user would not usually have this information. We have set this to the number of noise features in the data sets containing such features, and $\lceil 0.05 \times V \rceil$ to those with no noise features. This generates a clear bias in favour of FSFS when experimenting with a data set containing noise features, and, a bias towards an average frequency of about 0.95 in the other cases.

With the above in mind we decided to run more experiments in order to determine whether the feature selection algorithms were removing relevant features. We calculated the entropy of each data set, given by the average entropy over each feature of a data set, before and after applying the feature selection methods. Table 3 presents the average entropy for each of the data sets we experiment with. These results clearly show our feature selection method reduces the entropy of each data set in all cases, and clearly outperforms FSFS. The only exception is the Poker Hand data set with noise features (in both cases, 10% and 20% added noise features), which is well aligned with our previous results.

Table 3. The average entropy computer over the features of a given data set. Under 'Without FS' we show the average entropy for each data set without the use of a feature selection algorithm (that is the reason why the values repeat, this is our baseline). Under 'With FS' we show the average entropy computed over the features that have been selected by one of the methods we experiment with. The results for WSMW*k*-means were computed over 100 runs.

	Without FS			With FS		
	0%	10%	20%	0%	10%	20%
WSMW*k*-means *(T=5)*						
CoverType	1.389	2.050	2.508	0.806	0.973	0.974
Hand Postures	5.012	5.311	5.555	3.597	3.498	3.795
IDA 2016	1.493	2.088	2.582	**1.045**	1.131	**1.062**
Online News Pop	3.092	3.552	3.932	**2.449**	2.330	2.190
Poker Hand	1.389	2.097	2.491	**1.344**	2.050	2.444
Record Linkage	1.100	1.790	2.354	0.541	0.607	0.524
Sensorless Drive	4.389	4.729	5.011	2.922	2.864	**3.277**
Skin Segmentation	7.563	7.672	7.737	**4.829**	3.943	3.726
WSMW*k*-means *(T=10)*						
CoverType	1.389	2.050	2.508	**0.794**	**0.972**	**0.960**
Hand Postures	5.012	5.311	5.555	**3.370**	**3.458**	**3.729**
IDA 2016	1.493	2.088	2.582	1.101	**1.130**	1.071
Online News Pop	3.092	3.552	3.932	2.456	**2.313**	**2.180**
Poker Hand	1.389	2.097	2.491	1.346	2.052	2.443
Record Linkage	1.100	1.790	2.354	**0.512**	**0.524**	**0.476**
Sensorless Drive	4.389	4.729	5.011	**2.780**	**2.781**	3.302
Skin Segmentation	7.563	7.672	7.737	4.954	**3.773**	**3.602**
FSFS						
CoverType	1.389	2.050	2.508	1.439	1.664	1.804
Hand Postures	5.012	5.311	5.555	5.307	5.234	5.234
IDA 2016	1.493	2.088	2.582	1.555	2.283	3.082
Online News Pop	3.092	3.552	3.932	3.192	3.677	4.510
Poker Hand	1.389	2.097	2.491	1.439	**1.752**	**1.561**
Record Linkage	1.100	1.790	2.354	1.204	1.100	1.100
Sensorless Drive	4.389	4.729	5.011	4.361	4.593	4.643
Skin Segmentation	7.563	7.672	7.737	7.543	7.695	7.866

WSMW*k*-means is a clustering-based method. Thus, we can reinforce our previous results by demonstrating it improves cluster compactness. We demonstrate this is the case by measuring the average component-wise distance between entities and their respective centroids, given by

$$P'(S,Z) = \sum_{k=1}^{K} \sum_{i \in S_x} V^{-1} \sum_{v=1}^{V} (x_{iv} - z_{kv})^2. \qquad (9)$$

We measure cluster compactness with (9) rather than Equation (1) because the latter would tend to be higher as the number of features increases (ie. when features are not removed by our method). Table 4 presents the average of Equation (9) over 100 runs for each data set. We can clearly see that our feature selection algorithm decreases the average component-wise distance between entities and centroids, which means clusters are more compact.

Table 4. The average computed over 100 runs, of the average component-wise distance between entities and their respective centroids. Under 'Without FS' we show these results for the data sets with all features. Under 'With FS' we show the results computed over the features that have been selected by our method at each run.

	Without FS			With FS		
	0%	10%	20%	0%	10%	20%
WSMW*k*-means (*T=5*)						
CoverType	0.122	0.167	0.199	0.110	0.109	0.112
Hand Postures	0.060	0.086	0.116	0.052	**0.053**	**0.058**
IDA 2016	0.007	**0.021**	**0.034**	**0.002**	**0.002**	**0.002**
Online News Pop	0.229	0.256	0.275	0.173	0.177	0.179
Poker Hand	1.337	1.284	1.255	1.307	1.248	1.218
Record Linkage	0.140	0.147	0.148	0.060	0.069	**0.059**
Sensorless Drive	0.025	0.101	0.173	0.014	0.012	**0.021**
Skin Segmentation	0.064	**0.091**	0.108	0.042	0.046	0.059
WSMW*k*-means (*T=10*)						
CoverType	**0.116**	**0.161**	**0.192**	**0.105**	**0.103**	**0.103**
Hand Postures	**0.058**	0.085	0.115	**0.049**	**0.053**	**0.058**
IDA 2016	**0.006**	**0.021**	**0.034**	**0.002**	**0.002**	**0.002**
Online News Pop	**0.225**	**0.249**	**0.271**	**0.170**	**0.175**	**0.175**
Poker Hand	**1.321**	**1.268**	**1.236**	**1.290**	**1.231**	**1.196**
Record Linkage	**0.133**	**0.140**	**0.143**	**0.052**	**0.063**	0.064
Sensorless Drive	**0.022**	**0.100**	**0.171**	**0.011**	**0.011**	**0.021**
Skin Segmentation	**0.061**	0.091	**0.107**	**0.039**	**0.044**	**0.055**

The results shown in Tables 2, 3, and 4 seem to suggest higher values of T lead to better results. This is a sensible conjecture as the higher the value of T the more data our method has access to, but it requires further investigation. With this in mind we ran a series of experiments with $T = 10, 11, ..., 30$ on two data sets: Record Linkage (the largest data set we experiment with in this paper) and Skin Segmentation, each with added 20% noise features. Figure 1 shows the average frequency of noise features being selected by our method over T. These experiments show a tendency for better results the higher the value of T.

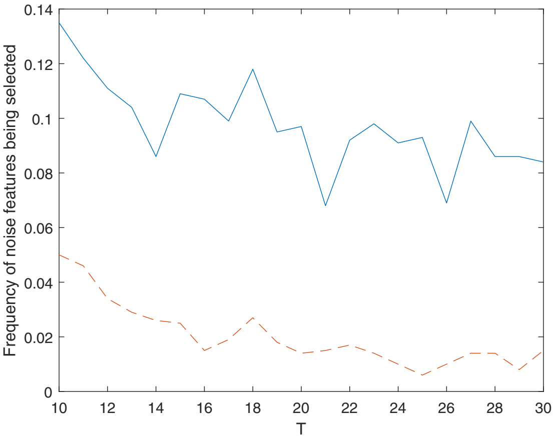


Fig. 1. Average frequency of noise features being selected by WSMW*k*-means over T on data sets with 20% extra noise features. The solid and dashed lines represent the Skin Segmentation and Record Linkage data sets, respectively.

The results in this section show that our method is capable of removing irrelevant features that may be present in large data sets. These experiments also show that our method requires a small amount of data to work. In the case of Record Linkage (the largest data set we experiment with in this paper), our method identified the vast majority of irrelevant features (approximately 86.0% when adding 10% noise features, and 94.5% when adding 20% noise features) using only 0.4% of the data (for details, see Table 2, with $T = 5$).

6. Conclusion

In this paper we introduced a new unsupervised feature selection algorithm designed specifically to deal with large data sets. Our method models the degree of relevance of each feature at each cluster using a feature weight that is found using samples of a given data set. It then applies a clearly defined threshold to these feature weights in order to decide which features should be selected and which should be discarded.

We empirically show that our method is capable of removing irrelevant features, leading to a lower average entropy and clusters that are more compact. The major advantages of our method is that it reaches such results by processing only a fraction of a given data set, and it does not require the whole data set to be small enough to fit in the memory of a computer. We believe this is of particular interest to those analysing data sets that are too large for other methods to be applied. Our future work will attempt to optimise the memory management of our method to better deal with sparse data sets, as well as the fact it considers one feature at a time.

References

Altman, N.S., 1992. An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician 46, 175–185.

de Amorim, R.C., 2016. A survey on feature weighting based k-means algorithms. Journal of classification 33, 210–242.

de Amorim, R.C., Hennig, C., 2015. Recovering the number of clusters in data sets with noise features using feature rescaling factors. Information Sciences 324, 126–145.

de Amorim, R.C., Mirkin, B., 2012. Minkowski metric, feature weighting and anomalous cluster initializing in k-means clustering. Pattern Recognition 45, 1061–1075.

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

Cai, D., Zhang, C., He, X., 2010. Unsupervised feature selection for multi-cluster data, in: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM. pp. 333–342.

Chan, E.Y., Ching, W.K., Ng, M.K., Huang, J.Z., 2004. An optimization algorithm for clustering using weighted dissimilarity measures. Pattern recognition 37, 943–952.

Chandrashekar, G., Sahin, F., 2014. A survey on feature selection methods. Computers & Electrical Engineering 40, 16–28.

Chen, C.P., Zhang, C.Y., 2014. Data-intensive applications, challenges, techniques and technologies: A survey on big data. Information Sciences 275, 314–347.

Dheeru, D., Karra Taniskidou, E., 2017. UCI machine learning repository. URL: http://archive.ics.uci.edu/ml.

Guyon, I., Elisseeff, A., 2003. An introduction to variable and feature selection. Journal of machine learning research 3, 1157–1182.

Hans-Hermann, B., 2008. Origins and extensions of the k-means algorithm in cluster analysis. Journal Electronique dHistoire des Probabilités et de la Statistique Electronic Journal for History of Probability and Statistics 4.

Huang, J.Z., Ng, M.K., Rong, H., Li, Z., 2005. Automated variable weighting in k-means type clustering. Pattern Analysis and Machine Intelligence, IEEE Transactions on 27, 657–668.

Jain, A., 2010. Data clustering: 50 years beyond k-means. Pattern Recognition Letters 31, 651–666. doi:10.1016/j.patrec.2009.09.011.

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

Kaisler, S., Armour, F., Espinosa, J.A., Money, W., 2013. Big data: Issues and challenges moving forward, in: System sciences (HICSS), 2013 46th Hawaii international conference on, IEEE. pp. 995–1004.

Kim, Y., Street, W.N., Menczer, F., 2000. Feature selection in unsupervised learning via evolutionary search, in: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM. pp. 365–369.

MacQueen, J., 1967. Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, California, USA. pp. 281–297.

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

Mirkin, B., 2012. Clustering: A Data Recovery Approach. Computer Science and Data Analysis, CRC Press, London, UK.

Mitra, P., Murthy, C., Pal, S.K., 2002. Unsupervised feature selection using feature similarity. IEEE transactions on pattern analysis and machine intelligence 24, 301–312.

Panday, D., de Amorim, R.C., Lane, P., 2018. Feature weighting as a tool for unsupervised feature selection. Information Processing Letters 129, 44–52.

R Core Team, 2014. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. URL: http://www.R-project.org.

Sculley, D., 2010. Web-scale k-means clustering, in: Proceedings of the 19th international conference on World wide web, ACM. pp. 1177–1178.

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

Wishart, D., 1998. Clustan. URL: http://www.clustan.com/.

Declaration of interest: none