Feature weighting as a tool for unsupervised feature selection

Deepak Panday[a], Renato Cordeiro de Amorim[b], Peter Lane[a]

[a]*School of Computer Science, University of Hertfordshire, College Lane AL10 9AB, UK.*
[b]*School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park CO4 3SQ, UK.*

Abstract

Feature selection is a popular data pre-processing step. The aim is to remove some of the features in a data set with minimum information loss, leading to a number of benefits including faster running time and easier data visualisation. In this paper we introduce two unsupervised feature selection algorithms. These make use of a cluster-dependent feature-weighting mechanism reflecting the within-cluster degree of relevance of a given feature. Those features with a relatively low weight are removed from the data set. We compare our algorithms to two other popular alternatives using a number of experiments on both synthetic and real-world data sets, with and without added noisy features. These experiments demonstrate our algorithms clearly outperform the alternatives.

Keywords: Feature selection, clustering, feature weighting.

1. Introduction

Dimensionality reduction is a common pre-processing step in data analysis. There are different reasons for this, including: (i) it may help save processing time when running a machine learning algorithm; (ii) a data set would require less space to be saved in a hard disk, or loaded into the main memory of a computer; (iii) it may allow the creation of more meaningful visualisation aids; (iv) it may reduce issues raised by the *curse of dimensionality* [1, 2, 3].

Generally speaking, there are two main classes of methods for dimensionality reduction: feature selection and feature extraction. Methods applying feature selection attempt to find the smallest subset of relevant features, according to a given criterion. Feature selection methods do not alter the features themselves, preserving their original meaning to the user. Methods applying feature extraction attempt to reduce the dimensionality of data sets by combining features. Such methods do attempt to minimise information loss, however, the original features and their meaning to the user are usually lost.

Feature weighting can be seen as a generalisation of feature selection. Consider a data set Y containing n entities y_i, each described over the same set of features $V = \{v_1, v_2, ..., v_m\}$. In this scenario a feature weighting algorithm will attempt to assign a weight w_v to each feature $v \in V$, usually in the interval $[0, 1]$. The weight w_v reflects the degree of relevance of v to the particular problem at hand. Feature selection algorithms substitute the $[0, 1]$ interval by the constraint $w_v \in \{0, 1\}$ for each $v \in V$. If $w_v = 1$, v is placed in the subset of features that have been selected, and discarded otherwise.

Clustering algorithms employ unsupervised learning to partition a data set Y into K clusters $S = \{S_1, S_2, ..., S_K\}$, according to some notion of similarity. This means they are capable of assigning an entity y_i to a particular cluster S_k without requiring labelled data to learn from. The main objective of this type of algorithm is to generate a clustering S in which there is homogeneity within clusters, but heterogeneity between clusters. Clustering algorithms have a long history (see for instance [4, 5] and references therein), and they can be generally divided into two main classes: hierarchical and partitional algorithms. The latter includes algorithms generating a set of disjoint clusters, so that $S_k \cap S_j = \emptyset$ for $k, j = 1, 2, ..., K$ and $k \neq j$. Hierarchical clustering algorithms generate a clustering S and provide information regarding the relationships between the clusters themselves, at a usually higher computational cost. These tree-like relationships can be easily visualised with a dendrogram.

Here, we are particularly interested in partitional clustering algorithms. Recent developments in this field

Email addresses: `d.panday@herts.ac.uk` (Deepak Panday), `r.amorim@essex.ac.uk` (Renato Cordeiro de Amorim), `p.c.lane@herts.ac.uk` (Peter Lane)

have led to various partitional clustering algorithms capable of assigning feature weights (see for instance [6] and references therein). These algorithms model the relevance of a feature v using a feature weight w_v. This fits well with the intuitive idea that even among relevant features there may be different degrees of relevance. However, they may be powerless in situations in which a user actually needs to reduce the dimensionality of a data set. This is because an irrelevant feature is usually assigned a very low weight, but not zero. The weight tends to be low enough for the feature to have a meaningless contribution to the clustering, but high enough for the feature to still be used in computations. In other words, if one needs to reduce the amount of space a data set takes, or if one intends to apply a different machine learning algorithm (one that does not support feature weights) after the clustering, feature weighting may not be the most appropriate solution.

In this paper we address the problem above by devising methods capable of taking the granularity given by feature weights, and generating a subset of selected features. That is, our feature selection methods are an extension of feature weighting. We have divided this paper into six sections. Section 2 sets the foundation by presenting related work. Section 3 introduces our new methods for unsupervised feature selection. We compare our methods to two popular unsupervised feature selection algorithms: feature selection with feature similarity [7] and multi-cluster feature selection [8]. Details of our settings and experiments can be found in Sections 4 and 5, respectively. Section 6 concludes our paper.

2. Related work

This section describes the work that is directly related to our paper. We begin by discussing clustering, including clustering algorithms that are capable of generating feature weights. In the following subsection we describe some popular unsupervised feature selection algorithms. In the following sections we use these algorithms for comparison.

2.1. Clustering and feature weighting

Clustering algorithms follow the unsupervised learning framework, and thus do not require any labelled samples for learning. The k-means algorithm [9, 10] is arguably the most popular partitional clustering algorithm [4, 5, 11]. It aims to partition a data set Y containing n entities y_i into K clusters $S = \{S_1, S_2, ..., S_K\}$, so that $\sum_{k=1}^{K} |S_k| = n$ and $S_k \cap S_j = \emptyset$, for $k, j = 1, 2, ..., K$ and $k \neq j$. For each $S_k \in S$, k-means generates a centroid $c_k \in C$, where C is the set of all centroids. This c_k can be used to describe the general characteristics of the entities assigned to a cluster S_k, and it is often called the prototype of S_k. K-means generates a clustering S by minimising

$$W(S, C) = \sum_{k=1}^{K} \sum_{y_i \in S_k} \sum_{v \in V} (y_{iv} - c_{kv})^2, \qquad (1)$$

where V is the set of features, y_{iv} and c_{kv} are the v^{th} coordinates of y_i and c_k, respectively.

Since (1) applies the squared Euclidean distance between entities and respective centroids, we can set $c_{kv} = |S_k|^{-1} \sum_{y_i \in S_k} y_{iv}$. In other words, the centroid c_k is the component-wise centre of $y_i \in S_k$. K-means minimises (1) using three straightforward steps: (i) randomly select K entities of Y, and copy their values to the initial centroids $c_1, c_2, ..., c_K$; (ii) for each entity $y_i \in Y$ find c_k, the nearest centroid to y_i, and assign y_i to S_k; (iii) update each centroid $c_k \in C$ to the component-wise centre of $y_i \in S_k$. Steps (ii) and (iii) are repeated until convergence.

Very much like any other machine learning algorithm, k-means is not without weaknesses. Among these: (i) it requires the number of clusters, K, to be known beforehand; (ii) the minimisation of (1) may get trapped in local minima; (iii) the final clustering depends heavily on the initial centroids, usually chosen at random; (iv) the algorithm is biased towards spherical clusters (v) every feature is treated equally, regardless of its actual relevance.

The intelligent Minkowski k-means (*imwk*-means) [12] has been designed to address some of the above. This algorithm calculates distances using a weighted version of the Minkowski distance.

$$d_p(y_i, c_k) = \sum_{v \in V} w_{kv}^p |y_{iv} - c_{kv}|^p, \qquad (2)$$

where p is a user-defined Minkowski exponent. The reason for the use of (2) is twofold. First, any distance measure will introduce a shape bias to clusters. By using the Minkowski distance one can set this bias to shapes other than spherical. Second, we can use w_{kv} to change the contribution v makes to the clustering. The general idea is that w_{kv} reflects the relevance of v at cluster S_k. The above leads to the criterion

$$W(S, C, w) = \sum_{k=1}^{K} \sum_{y_i \in S_k} \sum_{v \in V} w_{kv}^p |y_{iv} - c_{kv}|^p, \qquad (3)$$

where w_{kv} is inversely proportional to the dispersion of $y_{iv} \in S_k$. This follows the intuitive idea that features with a relatively high dispersion should have lower weight than that of features with a relatively low dispersion. We calculate the within cluster dispersion of a feature $v \in V$, $D_{kv} = \sum_{i \in S_k} |y_{iv} - c_{kv}|^p$ and the weight itself can be set to

$$w_{kv} = \left[\sum_{u \in V} \left[\frac{D_{kv}}{D_{ku}} \right]^{\frac{1}{p-1}} \right]^{-1}. \tag{4}$$

The *imwk*-means algorithm makes use of a Minkowski based version of the Anomalous Pattern method present in intelligent k-means [13]. This is a popular k-means initialisation, which has been favourably compared with various others [14, 15]. It works by setting the centre of the data set as a reference point and then iteratively removing anomalous clusters of the data set. The centroids of these anomalous clusters can be used as initial centroids in virtually any k-means based algorithm. The cardinality of this set of centroids determines the number of clusters in the data set, allowing *imwk*-means to address the k-means weaknesses (i), (ii) and (iii).

We have chosen to use *imwk*-means in particular because of our previous success with it, and positive results in comparisons [12, 6, 16].

2.2. Unsupervised feature selection

The main objective of feature selection algorithms is to determine a proper subset $V' \subset V$, so that describing $y_i \in Y$ over V' instead of V does not lead to loss of information. Feature selection has a long history (see for instance [17, 18, 19, 20, 21] and references therein), however this tends to be for algorithms following the supervised learning framework. Such feature selection algorithms require labelled data to learn from. Here, we are interested in unsupervised learning algorithms. This section discusses two of the most popular feature selection algorithms that do not require labelled samples. Readers interested in a more general discussion are directed to surveys, such as [22, 23] and references therein.

Feature selection using feature similarity

Feature selection using feature similarity (FSFS) [7] is probably the most cited unsupervised feature selection algorithm. It calculates pairwise feature similarities in order to determine a set of maximally independent features, and then discards those that are considered redundant. This is the first algorithm to use the maximum information compression index, defined below for features v_t and v_j, in a feature selection scenario.

$$2\lambda_2(v_t, v_j) = var(v_t) + var(v_j) - \sqrt{(var(v_t) + var(v_j))^2 - 4var(v_t)var(v_j)(1 - \sigma(v_t, v_j)^2)}, \tag{5}$$

where $\sigma(v_t, v_j)$ denotes the Pearson correlation coefficient between v_t and v_j, and $var()$ represents the variance of a feature passed as a parameter. The value of λ_2 is zero when the features are linearly dependent, and increases as the amount of dependency decreases.

Feature selection using feature similarity (FSFS)

```
1. Select the value for $k$, subject to $k \leq m - 1$. Set $V' \leftarrow V$.
2. For each $v' \in V'$, compute $r^k_{v'}$.
3. Find the feature $v'^*$ for which $r^k_{v'^*}$ is minimum.
4. Remove from $V'$ the $k$ nearest-neighbours of $v'^*$.
5. Set $\epsilon = r^k_{v'^*}$.
6. If $k > m_{V'}$ go to Step 9.
7. While $r^k_{v'^*} > \epsilon$
   (a) $k = k - 1$.
       $r^k_{v'^*} = \inf_{v' \in V'}$
       ($k$ is decremented by 1, until the $k$th nearest-neighbour of at least one of the features in $V'$ is less than $\epsilon$-dissimilar with the feature).
   (b) If $k = 1$ go to Step 9.
8. Go to Step 2.
9. Return the feature set $V'$ as the reduced feature set.
```

In the algorithm above $m_{V'}$ represents the cardinality of V', and k relates to the kth nearest-neighbours algorithm. By setting k we can calculate $r^k_{v'}$ as the dissimilarity between feature v' and its kth nearest-neighbour in V' using (5). Having k as a user-defined parameter allows for multi-scale representation of data sets. However, the use of k also means that to generate the smallest V' (that with the lowest $m_{V'}$) representing the information of Y, one needs to find its optimal value - we present one way to do this in Section 5.

Multi-Cluster Feature Selection

Traditionally, unsupervised feature selection algorithms select the top ranked features of a data set based on scores computed independently for each feature. This approach certainly decreases the computational effort required to find a meaningful subset of features. However, since it neglects the possible correlations between features it may not be able to produce an optimal feature subset.

With the above in mind, Deng Cai et al. introduced Multi-Cluster Feature Selection (MCFS) [8]. Their

method has been inspired by recent developments in spectral analysis (manifold learning) [24, 25] and L1-regularized models for subset selection [26, 27]. MCFS uses multiple eigenvectors of graph Laplacian matrix, allowing for the selection of those features so that the multi-cluster structure of the data can be best preserved.
Multi-cluster feature selection

1. Construct a k-nearest-neighbours graph.
2. Solve a generalised eigen-problem, leading to the K top eigenvectors with respect to the smallest eigenvalues.
3. Solve K L1-regularised regression problems, leading to K sparse coefficient vectors.
4. Compute the MCFS score for each feature.
5. Return the top $m_{V'}$ features according to their MCFS scores.

For details on the above, particularly, steps 2 to 4, please refer to the original paper[8]. MCFS has received considerable attention, with evidence of such in citations to the original paper.

3. Feature selection via feature weighting

We build on previous work in feature weighting by using the *imwk*-means as a tool to find feature weights. We then use these weights as a foundation for feature selection. Given a data set $Y = (y_i)$ where each y_i is described over m features $v \in V$, *imwk*-means generates a clustering $S = \{S_1, S_2, ..., S_K\}$, where $| \cup_{k=1}^{K} S_k| = n$, and $S_k \cap S_j = \emptyset$ for $k, j = 1, 2, ..., K$ and $k \neq j$. This algorithm also generates a centroid c_k for each cluster $S_k \in S$, as well as cluster dependent weights w_{kv} for $k = 1, 2, ..., K$ and $v \in V$.

A feature weight w_{kv} models the degree of relevance of a feature v at a cluster S_k. A feature weight w_{kv} is defined in the interval [0, 1], subject to $\sum_{v \in V} w_{kv} = 1$ for $k = 1, 2, ..., K$. Our methods (explained below) are capable of mapping $w_{kv} \in [0, 1]$ to $w_v \in \{0, 1\}$, so that w_v effectively selects or deselects a feature v. The key to this mapping is, as one would imagine, to find a threshold θ so that

$$w_v = \begin{cases} 1, \text{if } w_v \geq \theta, \\ 0, \text{if } w_v < \theta. \end{cases} \tag{6}$$

Here we introduce two feature selection methods based on the above. These are simple but rather powerful methods, as the results from our experiments show (see Section 5).

The *imwk*-means generates K weights for each feature v, however, to apply (6) we need each feature to have one weight. To do so we first present a method called meanFSFW. In this method we set $w_v = K^{-1} \sum_{k=1}^{K} w_{kv}$ for $v = 1, 2, ..., m$. We then set $\theta = m^{-1}$ and apply (6). Those features with $w_v = 0$ are then simply removed from the data set.

Our second method maxFSFW first sets $w_v = max(\{w_{1v}, w_{2v}, ..., w_{Kv}\}$, and then normalises the weights $w_v = w_v / \sum_{v \in V} w_v$, so that $\sum_{v \in V} w_v = 1$. Now that we have one weight per feature we can set $\theta = m^{-1}$ and apply (6). We remove those features with $w_v = 0$ from the data set.

Both of the methods described above set $\theta = m^{-1}$. Consider (4) in the case of noise being truly random and uniformly distributed. In this case $D_{ku} = c$ for $u \in V'$, where V' represents the set of noisy features and c is a constant. Given that each D_{kv} is divided by the same value c, and that (4) is subject to $\sum_{v \in V} w_{kv} = 1$ for $k = 1, 2, ..., K$, this leads to $w_{ku} = m^{-1}$. Therefore, meaningful features should have a higher value than that.

4. Setting of experiments

We validate our methods through a number of experiments using synthetic and real-world data, in both cases with and without noisy features: we define a noisy feature as one composed of within-domain uniformly random values. In an ideal scenario a feature selection algorithm would remove all noisy features from a data set.

We experiment with 12 configurations for our synthetic data, generating 50 data sets for each configuration totalling 600 synthetic data sets. Each data set contains Gaussian clusters with diagonal covariance matrices of $\sigma^2 = 0.5$. Each centroid was independently generated from the Gaussian distribution $N(0, 1)$, and each entity had a chance of $1/K$ to come from any cluster. Table 1 presents the configurations of our synthetic data.

For our experiments with real-world data, we selected some of the most popular data sets from the popular UCI macine learning repository [28] (see Table 2 for details). Since we cannot be sure which features are relevant, we used each of these data sets to generate two noisy versions. To the first we have added $\lceil m/2 \rceil$ noisy features, to the second we added m noisy features. These give a total of 30 real-world data sets.

We then proceeded to standardise the data. Numerical features are put into the range [0, 1], using:

$$y_{iv} = \frac{y_{iv} - \bar{y_v}}{max(y_v) - min(y_v)}, \tag{7}$$

where $\bar{y_v} = n^{-1} \sum_{i=1}^{n} y_{iv}$. We have decided to apply (7) rather than the popular z-score because the latter is bi-

Table 1: The twelve different configurations of the synthetic data sets containing Gaussian clusters we use in our experiments. NF refers to the number of extra features composed of uniformly random noise added to the data set.

			Features		
Data set	Entities (n)	Clusters (K)	Original	Noise	Total (m)
1000x8-2	1000	2	8	0	8
1000x8-2 + 4NF	1000	2	8	4	12
1000x8-2 + 8NF	1000	2	8	8	16
1000x12-3	1000	3	12	0	12
1000x12-3 + 6NF	1000	3	12	6	18
1000x12-3 + 12NF	1000	3	12	12	24
1000x16-4+8NF	1000	4	16	0	16
1000x16-4	1000	4	16	8	24
1000x16-4+16NF	1000	4	16	16	32
1000x20-5	1000	5	20	0	20
1000x20-5+10NF	1000	5	20	10	30
1000x20-5+20NF	1000	5	20	20	40

Table 2: Real-world data sets used in our experiments. We generated two more data sets from each data set by adding either $\lceil m/2 \rceil$ or m extra features composed of uniformly random noise.

			Original Features		
Name	Entities (n)	Clusters (K)	Numerical	Categorical	Total (m)
Australian credit approval	690	2	6	8	14
Car evaluation	1,728	4	0	6	6
Ecoli	336	8	7	0	7
Glass	214	6	9	0	9
Ionosphere	351	2	33	0	33
Iris	150	3	4	0	4
Wine	178	3	13	0	13
Zoo	101	7	1	15	16
Low Resolution Spectrometer	531	10	101	0	101
Gas Sensor Array Drift Batch10	3,600	6	129	0	129

ased towards unimodal distributions [11, 6]. We replaced each categorical $v \in V$ containing t categories by t binary features. For a given entity y_i only one of these t binary features was set to one, that representing the original category in y_{iv}.

We measure the performance of our methods, and those we use as a benchmark (see Section 2) in two ways. First, we are interested to know if the selected features are representative of the data. Clearly, there are a number of algorithms we could use to measure this representativeness. We use a clustering algorithm so that our results are completely data-driven. Thus, we run k-means 100 times using solely the selected features. The final clustering is that with the lowest output criterion (1). We then measure accuracy by comparing this clustering with the known labels using the adjusted Rand Index (ARI) [29], a popular corrected-for-chance version of the Rand index [30].

Second, we would also like the set of selected features to have the lowest possible cardinality, without losing representativeness. In order to analyse this, we compare all methods based on the number of features they select. This reveals the possible trade offs between the representativeness of the features (measured as an accuracy), and the number of selected features.

5. Results and analysis

In this section we compare our new methods mean-FSFW and maxFSFW (see Section 3) with FSFS and MCFS (see Section 2). Some of the algorithms above require the setting of a parameter. Unfortunately, the original authors of FSFS and MCFS did not clearly state a method to select such parameters under the unsupervised learning framework. Thus, we begin this section by describing how these parameters can be selected.

Given a clustering $S = \{S_1, S_2, ..., S_K\}$ there are a number of cluster validity indices that can be used to measure how good S is. There is no index that is better than all others in all cases, but usually the Silhouette width is among the top performers (for a recent comparison see [31] and references therein). We can define the Silhouette width of an entity $y_i \in S_k$ as

$$SIL(y_i) = \frac{b(y_i) - a(y_i)}{max\{a(y_i), b(y_i)\}}, \tag{8}$$

where $a(y_i)$ is the average dissimilarity between $y_i \in S_k$ and all other entities in S_k, and $b(y_i)$ is the lowest average dissimilarity between y_i to any other cluster. A Silhouette width close to one indicates $a(y_i) << b(y_i)$ meaning that y_i is well placed in S_k, an index value close to minus one indicates the opposite. With (8) we can calculate the Silhouette width for Y as $SIL(Y) = n^{-1} \sum_{i \in Y} SIL(y_i)$.

In the case of FSFS, its parameter k is used for a neighbourhood search using the popular k-nearest neighbours algorithm. Since k is directly related to the number of selected features, we experiment with $k = 1, 2, ..., m-1$. Given a value for the FSFS parameter (k), we then run k-means 100 times selecting as final clustering that with the lowest output criterion, and then calculate the Silhouette width of the clustering. We select the value of k leading to the highest Silhouette width as being the optimal. It is probably worth noting that the k in k-nearest neighbours is not the same as the k in k-means. The latter represents the number of clusters in a data set, something we know for all the data sets we experiment with. We also present the best case scenarios. In these we simply select for k the value leading to the highest ARI in relation to the known labels. This is an unrealistic data analysis scenario, but we are interested in finding the best possible case for FSFS.

The MCFS algorithm has three parameters. First we have a parameter k also related to the use of k-nearest neighbours. In this case the original authors of MCFS suggest a default value of five, and we use their suggestion in our experiments. Their second parameter relates to the number of eigenfunctions MCFS uses: the original authors also suggest a default of five. The last parameter indicates the number of features to be selected. We searched for the optimal value for this last parameter in the exact same way we searched for the FSFS parameter. We also report the best possible cases for MCFS, based on trying all possible values for this last parameter and reporting that leading to the highest ARI in relation to the ground truth.

Since our methods are based on the *imwk*-means, we need to define the value for the Minkowski exponent p. For simplicity we have chosen to set this exponent to two, leading to a weighted version of the popular Euclidean distance.

Table 3 presents the accuracy measured using the adjusted Rand index (ARI) in relation to the ground truth. In this and the following tables, we put the best results in bold. In all cases the results are over the 50 data sets generated for each data set configuration. The reader will notice that the results under "Best Case" sometimes may be equal to or even higher than those in bold. In the "Best Case" experiments we simply provide a given algorithm with the parameter leading to the best result. Clearly this is unrealistic as it is impossible for a user to simply guess the optimal parameter value. The only reason we provide the results under "Best Case" is to demonstrate that the Silhouette width is a good parameter estimator. Table 3 clearly shows our method meanFSFW to be competitive or better than FSFS and MCFS. In this case, it is particularly interesting to see that meanFSFW is even superior to the best cases of FSFS and MCFS when the data sets contain noisy features.

Table 4 presents the average percentage of selected features, and respective standard deviation. We divided this table into two parts, so it is easier to understand the results. The upper part of the table presents the average percentage of original features selected. These original features are those generated using a Gaussian distribution (see details in Section 4). It is tempting to think that a successful method would select 100% of such features, however, it is possible to meaningfully represent the original data with less features. To avoid confusion we have not put any of these results in bold. For instance, meanFSFW selected on average 84.5% of the original features for 1000x8-2 +4NF, while FSFS selected 88.5%. However, Table 3 shows the accuracy of meanFSFW to be far superior to that of FSFS. The best method would be that with a low percentage of original features selected paired with a high accuracy. The second part of Table 4 presents the average percentage of noisy features selected. In this case it is fair to state that such noisy features do not add any meaningful information to the data set (for details on their generation see Section 4). Thus, a good feature selection algorithm would select the least amount of such features. Again, we can clearly see the superiority of meanFSFW, which typically removes all the noisy features.

Table 5 presents the accuracy results of our experiments in real-world data sets with and without noisy features. In these experiments meanFSFW presented the best results in 14 out of 30 cases. At first this may not seem an impressive result, but MCFS presented the best results in eight cases (one of which tied with meanFSFW), while FSFS presented the best results in six cases (three of which tied with meanFSFW). Here, maxFSFW outperformed all other methods by providing the best results in 21 out of 30 cases. The performance of maxFSFW was considerably better than its performance in previous experiments, probably because in real-world data there would be more cases of one feature being relevant to a single cluster. In these cases meanFSFW may end up removing such a feature.

Table 6 presents the average percentage of noisy features selected. For obvious reasons this table does not present experiments in data sets with no noisy features. One can clearly see the superiority of maxFSFW and meanFSFW. Each removed all noisy features except in two cases, out of 20. It is fair to conclude that if there is a reason to believe a data set contains Gaussian clusters meanFSFW should be applied and, if in doubt, apply maxFSFW.

5.1. Performance of classifiers after feature selection

Our previous experiments analysed results in the unsupervised learning scenario. This was indeed the main goal of our paper as the use of clustering and unsupervised feature selection algorithms is well-aligned. In this subsection we briefly show results using two popular classification algorithms: k-nearest neighbours (kNN) [32], and decision trees [33]. Our objective is not to claim one should use our methods in a supervised scenario, but to further analyse the behaviour of our methods.

Table 7 shows the results for experiments on the synthetic data sets. We can clearly see that in data sets with noise features there is an increase in the average adjusted Rand index. In these experiments, we set the number of neighbours for kNN to the square root of the number of entities in the data set, rounded. We do not present results for experiments on real-world data because of limitations in space.

Table 3: The average adjusted Rand Index and its standard deviation for the cluster recovery in synthetic data sets with and without noisy features.

		FS using Feature Similarity		Multi-Cluster FS		FS using Feature Weight	
Data sets	*k*-means	SIL	Best Case	SIL	Best Case	meanFSFW	maxFSFW
1000x8-2	**0.965/0.04**	0.935/0.08	0.937/0.08	0.958/0.06	0.958/0.06	0.932/0.10	0.898/0.13
1000x8-2 +4NF	0.788/0.39	0.772/0.39	0.920/0.13	0.789/0.39	0.891/0.25	**0.965/0.05**	0.818/0.36
1000x8-2 +8NF	0.768/0.40	0.771/0.39	0.922/0.12	0.770/0.41	0.847/0.32	**0.965/0.05**	0.859/0.32
1000x12-3	**0.984/0.03**	0.973/0.04	0.974/0.04	0.983/0.03	0.984/0.03	0.949/0.09	0.915/0.14
1000x12-3 +6NF	0.930/0.19	0.917/0.19	0.963/0.07	0.930/0.19	0.937/0.18	**0.983/0.03**	0.947/0.16
1000x12-3 +12NF	0.915/0.20	0.917/0.19	0.964/0.06	0.926/0.19	0.934/0.18	**0.984/0.03**	0.958/0.13
1000x16-4	**0.996/0.01**	0.993/0.01	0.993/0.01	0.995/0.01	0.996/0.0	0.977/0.03	0.946/0.07
1000x16-4 +8NF	0.987/0.04	0.977/0.07	0.988/0.02	0.987/0.05	0.989/0.05	**0.996/0.01**	0.991/0.01
1000x16-4 +16NF	0.985/0.04	0.973/0.07	0.986/0.02	0.985/0.05	0.987/0.05	**0.996/0.01**	0.994/0.01
1000x20-5	0.998/0.02	0.997/0.01	0.997/0.01	**0.999/0.00**	0.999/0.00	0.991/0.01	0.976/0.03
1000x20-5 +10NF	0.992/0.03	0.995/0.01	0.996/0.01	0.994/0.03	0.994/0.03	**0.999/0.00**	0.996/0.01
1000x20-5 +20NF	0.988/0.04	0.994/0.01	0.995/0.01	0.994/0.02	0.995/0.02	**0.999/0.00**	**0.999/0.00**

Table 4: The average percentage of noisy features selected, and respective standard deviations, on synthetic data sets.

		FS using Feature Similarity		Multi-Cluster FS		FS using Feature Weight	
	Data sets	SIL	Best Case	SIL	Best Case	meanFSFW	maxFSFW
avg % of original feat. selected	1000x8-2	74.500/20.61	73.250/21.21	82.500/11.73	80.500/15.24	41.500/10.74	32.500/11.46
	1000x8-2 +4NF	88.500/12.46	85.000/15.61	60.750/22.91	50.750/24.67	84.500/11.61	49.000/10.56
	1000x8-2 +8NF	88.500/13.88	85.000/16.58	62.000/25.24	53.250/25.35	99.000/3.39	71.000/13.79
	1000x12-3	86.335/11.15	84.668/11.47	85.501/9.55	83.834/11.36	45.500/8.20	37.000/9.15
	1000x12-3 +6NF	91.168/5.87	89.001/8.41	75.500/17.51	67.833/21.15	93.334/7.07	60.166/7.51
	1000x12-3 +12NF	91.001/7.04	88.667/8.46	74.334/19.28	68.001/22.20	99.833/1.17	82.333/9.07
	1000x16-4	92.500/3.06	92.875/2.50	82.125/13.17	80.500/13.03	46.000/6.59	39.125/8.08
	1000x16-4 +8NF	93.000/5.81	91.875/7.63	77.375/16.20	76.500/15.49	93.875/5.66	67.875/7.07
	1000x16-4 +16NF	91.250/9.01	90.375/8.59	80.625/15.97	76.875/15.63	99.750/1.22	86.750/8.07
	1000x20-5	93.700/5.81	93.200/6.54	71.700/15.74	69.000/13.78	46.700/5.53	39.800/6.63
	1000x20-5 +10NF	93.000/5.48	92.300/6.34	69.500/18.09	66.500/15.01	96.700/3.95	72.300/8.90
	1000x20-5 +20NF	92.500/6.02	90.300/7.51	74.600/20.42	69.700/17.93	99.900/0.70	89.600/6.77
avg % noisy feat. sel.	1000x8-2 +4NF	73.500/31.78	51.000/44.71	93.000/24.00	79.000/39.80	**0.000/0.00**	7.500/16.77
	1000x8-2 +8NF	73.750/35.64	44.500/42.22	89.750/24.96	76.500/36.02	**0.500/2.45**	4.250/9.56
	1000x12-3 +6NF	80.000/33.50	63.333/43.33	95.666/9.89	91.333/21.15	**0.000/0.00**	1.000/5.17
	1000x12-3 +12NF	81.001/32.62	60.667/43.69	85.834/18.50	77.667/23.95	**0.000/0.00**	0.833/3.00
	1000x16-4 +8NF	89.750/22.32	81.750/34.02	82.250/14.60	80.250/14.16	**0.000/0.00**	**0.000/0.00**
	1000x16-4 +16NF	87.875/27.28	73.125/40.80	69.500/20.60	62.250/20.54	**0.000/0.00**	0.500/2.11
	1000x20-5 +10NF	83.400/32.16	77.600/37.18	47.000/22.11	42.000/18.11	**0.000/0.00**	**0.000/0.00**
	1000x20-5 +20NF:20	78.700/36.83	66.400/42.24	43.600/29.33	32.300/20.40	**0.000/0.00**	**0.000/0.00**

Table 5: The average adjusted Rand Index and its standard deviation for the cluster recovery in real-world data sets with and without noisy features.

	Accuracy						
		FS using Feature Similarity		Multi-Cluster FS		FS using Feature Weight	
Data set	*k*-means	SIL	Best Case	SIL	Best Case	meanFSFW	maxFSFW
Austra C.A.	**0.500**	0.220	0.220	**0.500**	0.500	**0.500**	**0.500**
Austra C.A.+21NF	**0.500**	0.220	0.220	**0.500**	0.500	0.220	0.220
Austra C.A.+42NF	**0.500**	0.220	0.220	**0.500**	0.500	0.220	**0.500**
Car Evaluation	0.010	0.010	0.220	**0.020**	0.150	0.110	**0.020**
Car Evaluation +11NF	**0.040**	0.030	0.220	0.030	0.170	0.000	0.010
Car Evaluation +22NF	0.010	**0.150**	0.220	**0.150**	0.150	0.140	0.130
Ecoli	0.440	0.390	0.390	**0.450**	0.450	0.030	0.030
Ecoli +4NF	0.160	0.020	0.200	**0.220**	0.320	0.100	0.030
Ecoli +8NF	0.130	0.130	0.200	0.110	0.190	**0.210**	0.100
Glass	0.170	0.160	0.260	0.160	0.210	0.010	**0.190**
Glass +5NF	0.090	0.100	0.260	0.100	0.180	0.160	**0.180**
Glass +10NF	0.070	0.080	0.260	0.090	0.180	0.130	**0.260**
Ionosphere	0.170	**0.210**	0.330	0.160	0.420	**0.210**	**0.210**
Ionosphere +17NF	0.170	**0.210**	0.330	0.160	0.420	**0.210**	**0.210**
Ionosphere +34NF	0.170	**0.210**	0.330	0.150	0.330	**0.210**	**0.210**
Iris	0.710	0.620	0.850	0.850	0.850	**0.890**	**0.890**
Iris +2NF	0.660	0.480	0.560	0.480	0.890	0.720	**0.890**
Iris +4NF	0.490	0.490	0.560	0.500	0.890	**0.720**	**0.720**
Wine	0.860	**0.900**	0.900	0.850	0.900	0.880	0.830
Wine +7NF	0.800	0.770	0.820	0.900	0.900	0.740	**0.910**
Wine +14NF	0.820	0.680	0.790	0.770	0.850	0.870	**0.900**
Zoo	0.630	0.680	0.810	0.680	0.690	**0.840**	**0.840**
Zoo +8NF	0.680	0.690	0.840	0.680	0.690	**0.830**	**0.830**
Zoo +16NF	0.680	0.690	0.840	0.690	0.690	**0.830**	**0.830**
Low Resolution Spectrometer	0.202	0.210	0.230	0.240	0.260	0.260	**0.270**
Low Resolution Spectrometer +51NF	0.207	0.180	0.220	0.180	0.240	0.220	**0.280**
Low Resolution Spectrometer +101NF	0.194	0.180	0.200	0.190	0.230	**0.230**	0.070
Gass Sensor Array Batch10	0.126	0.120	0.210	0.170	0.170	**0.190**	0.140
Gass Sensor Array Batch10 +65NF	0.128	0.130	0.210	0.130	0.190	0.120	**0.140**
Gass Sensor Array Batch10 +129NF	0.126	**0.140**	0.210	**0.140**	0.160	0.130	**0.140**

Table 6: The average percentage of noisy features selected, and respective standard deviations, on real-world data sets containing about 50% or 100% extra features composed of uniformly random noise.

	Percentages of Noisy Feature Selected					
	FS using Feature Similarity		Multi-Cluster FS		FS using Feature Weight	
Data sets	SIL	Best Case	SIL	Best Case	meanFSFW	maxFSFW
Austra C.A.+21NF	100.000	100.000	**0.000**	0.000	**0.000**	**0.000**
Austra C.A.+42NF	100.000	100.000	40.480	0.000	**0.000**	**0.000**
Car Evaluation +11NF	9.090	9.090	**0.000**	9.090	100.000	**0.000**
Car Evaluation +22NF	22.730	4.550	**0.000**	0.000	100.000	100.000
Ecoli +4NF	100.000	25.000	100.000	25.000	**0.000**	**0.000**
Ecoli +8NF	87.500	0.000	100.000	25.000	**0.000**	**0.000**
Glass +5NF	80.000	0.000	100.000	0.000	**0.000**	**0.000**
Glass +10NF	80.000	0.000	100.000	0.000	**0.000**	**0.000**
Ionosphere +17NF	5.880	0.000	**0.000**	0.000	**0.000**	**0.000**
Ionosphere +34NF	2.940	0.000	**0.000**	0.000	**0.000**	**0.000**
Iris +2NF	100.000	0.000	100.000	0.000	**0.000**	**0.000**
Iris +4NF	100.000	0.000	100.000	0.000	**0.000**	**0.000**
Wine +7NF	100.000	0.000	85.710	85.710	**0.000**	**0.000**
Wine +14NF	100.000	28.570	85.710	50.000	**0.000**	**0.000**
Zoo +8NF	100.000	0.000	62.500	0.000	**0.000**	**0.000**
Zoo +16NF	100.000	0.000	75.000	0.000	**0.000**	**0.000**
Low Resolution Spectrometer +51NF	100.000	100.000	98.040	25.490	**0.000**	**0.000**
Low Resolution Spectrometer +101NF	100.000	100.000	97.030	48.510	**0.000**	100.000
Gas Sensor Array Drift Batch10 +65	78.290	100.000	**0.000**	**0.000**	**0.000**	**0.000**
Gas Sensor Array Drift Batch10 +129	83.720	100.000	**0.000**	**0.000**	**0.000**	**0.000**

Table 7: The average adjusted Rand index of two classifiers, and respective standard deviations, on synthetic data sets. Experiments followed a 10-fold cross validation, which was ran 50 times for each data set.

	*k*NN			Decision Tree		
Data sets	Orginal	After meanFWFS	After maxFWFS	Original	After meanFWFS	After maxFWFS
1000x8-2	**0.967**/0.05	0.934/0.09	0.899/0.13	**0.897**/0.10	0.884/0.12	0.852/0.16
1000x8-2 +4NF	0.960/0.05	**0.968**/0.05	0.945/0.09	0.895/0.10	**0.897**/0.10	0.890/0.12
1000x8-2 +8NF	0.943/0.07	**0.967**/0.05	0.960/0.06	0.892/0.10	0.896/0.10	**0.899**/0.1
1000x12-3	**0.984**/0.03	0.948/0.09	0.917/0.13	**0.891**/0.07	0.876/0.11	0.853/0.14
1000x12-3 +6NF	0.973/0.05	**0.984**/0.03	0.974/0.05	0.890/0.07	**0.892**/0.07	**0.892**/0.08
1000x12-3 +12NF	0.954/0.07	**0.985**/0.03	0.980/0.05	0.889/0.07	**0.891**/0.07	**0.891**/0.08
1000x16-4	**0.996**/0.01	0.977/0.03	0.948/0.06	**0.887**/0.05	0.885/0.05	0.856/0.07
1000x16-4 +8NF	0.987/0.01	**0.995**/0.01	0.990/0.01	0.886/0.05	0.887/0.05	**0.888**/0.05
1000x16-4 +16NF	0.972/0.02	**0.996**/0.01	0.994/0.01	0.885/0.05	**0.886**/0.05	**0.886**/0.05
1000x20-5	**0.999**/0.00	0.989/0.01	0.976/0.02	**0.888**/0.04	0.882/0.04	0.870/0.04
1000x20-5 +10NF	0.992/0.01	**0.999**/0.00	0.996/0.01	**0.888**/0.04	**0.888**/0.04	0.887/0.04
1000x20-5 +20NF	0.979/0.02	**0.999**/0.00	0.998/0.00	0.887/0.04	0.887/0.04	**0.888**/0.04

6. Conclusion

This paper introduces two novel unsupervised feature selection algorithms, meanFSFW and maxFSFW. Our methods are based on the concept of cluster-dependent feature weights, meaning we model the degree of relevance of a particular feature at a particular cluster. Each feature weight is defined in the interval $[0, 1]$. Our methods are capable of setting a feature weight to either one or zero. This effectively selects or deselects a given feature.

We have run an extensive number of experiments using synthetic and real-world data sets, in both cases with and without noisy features. These experiments compared our methods to two of the most popular unsupervised feature selection algorithms: feature selection using feature similarity (FSFS) and multi-cluster feature selection (MCFS). Our results demonstrate that our methods outperform the others in terms of selecting the most appropriate features, and removing noisy features.

References

[1] T. Pavlenko, On feature selection, curse-of-dimensionality and error probability in discriminant analysis, Journal of Statistical Planning and Inference 115 (2) (2003) 565–584.

[2] E. Chávez, G. Navarro, Probabilistic proximity search: Fighting the curse of dimensionality in metric spaces, Information Processing Letters 85 (1) (2003) 39–46.

[3] V. Pestov, On the geometry of similarity search: dimensionality curse and concentration of measure, Information Processing Letters 73 (1) (2000) 47–51.

[4] A. K. Jain, Data clustering: 50 years beyond k-means, Pattern recognition letters 31 (8) (2010) 651–666.

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

[6] R. C. de Amorim, A survey on feature weighting based k-means algorithms, Journal of Classification 33 (2) (2016) 210–242.

[7] P. Mitra, C. Murthy, S. K. Pal, Unsupervised feature selection using feature similarity, Pattern Analysis and Machine Intelligence, IEEE Transactions on 24 (3) (2002) 301–312.

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

[9] G. H. Ball, D. J. Hall, A clustering technique for summarizing multivariate data, Behavioral science 12 (1967) 153–155.

[10] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Vol. 1, Oakland, CA, USA., 1967, pp. 281–297.

[11] B. Mirkin, Clustering: a data recovery approach, CRC Press, 2012.

[12] R. C. De Amorim, B. Mirkin, Minkowski metric, feature weighting and anomalous cluster initializing in k-means clustering, Pattern Recognition 45 (3) (2012) 1061–1075.

[13] B. Mirkin, Clustering: A Data Recovery Approach, Computer Science & Data Analysis, Chapman & Hall/CRC, 2012.

[14] M. M.-T. Chiang, B. Mirkin, Intelligent choice of the number of clusters in k-means clustering: an experimental study with different cluster spreads, Journal of classification 27 (1) (2010) 3–40.

[15] R. C. de Amorim, An empirical evaluation of different initializations on the number of k-means iterations, in: Mexican International Conference on Artificial Intelligence, Springer, 2012, pp. 15–26.

[16] R. L. Melvin, R. C. Godwin, J. Xiao, W. G. Thompson, K. S. Berenhaut, F. R. Salsbury Jr, Uncovering large-scale conformational change in molecular dynamics without prior knowledge, Journal of Chemical Theory and Computation.

[17] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, Journal of machine learning research 3 (Mar) (2003) 1157–1182.

[18] H. Liu, H. Motoda, R. Setiono, Z. Zhao, Feature selection: An ever evolving frontier in data mining., FSDM 10 (2010) 4–13.

[19] G. Chandrashekar, F. Sahin, A survey on feature selection methods, Computers & Electrical Engineering 40 (1) (2014) 16–28.

[20] J.-C. Lamirel, P. Cuxac, K. Hajlaoui, A novel approach to feature selection based on quality estimation metrics, in: Advances in Knowledge Discovery and Management, Springer, 2017, pp. 121–140.

[21] J.-C. Lamirel, I. Falk, C. Gardent, Federating clustering and

cluster labelling capabilities with a single approach based on feature maximization: French verb classes identification with igngf neural clustering, Neurocomputing 147 (2015) 136–146.

[22] J. G. Dy, Unsupervised feature selection, Computational methods of feature selection (2008) 19–39.

[23] S. Alelyani, J. Tang, H. Liu, Feature selection for clustering: A review., Data Clustering: Algorithms and Applications 29.

[24] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering., in: NIPS, Vol. 14, 2001, pp. 585–591.

[25] A. Y. Ng, M. I. Jordan, Y. Weiss, et al., On spectral clustering: Analysis and an algorithm, Advances in neural information processing systems 2 (2002) 849–856.

[26] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, et al., Least angle regression, The Annals of statistics 32 (2) (2004) 407–499.

[27] T. Hastie, R. Tibshirani, J. Friedman, The elements of statistical learning: data mining, inference, and prediction, Springer Series in Statistics, Springer, 2009.

[28] C. Blake, C. J. Merz, {UCI} repository of machine learning databases.

[29] W. M. Rand, Objective criteria for the evaluation of clustering methods, Journal of the American Statistical association 66 (336) (1971) 846–850.

[30] L. Hubert, P. Arabie, Comparing partitions, Journal of classification 2 (1) (1985) 193–218.

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

[32] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE transactions on information theory 13 (1) (1967) 21–27.

[33] S. R. Safavian, D. Landgrebe, A survey of decision tree classifier methodology, IEEE transactions on systems, man, and cybernetics 21 (3) (1991) 660–674.