Information Processing Letters 129 (2018) 44–52

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

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, Colchester CO4 3SQ, UK*

A R T I C L E   I N F O

Article history:
Received 8 February 2017
Received in revised form 8 September 2017
Accepted 15 September 2017
Available online 21 September 2017
Communicated by Jef Wijsen

Keywords:
Feature selection
Clustering
Algorithms

A B S T R A C T

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.

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* [1–3].

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

* Corresponding author.
E-mail addresses: d.panday@herts.ac.uk (D. Panday), r.amorim@essex.ac.uk (R. Cordeiro de Amorim), p.c.lane@herts.ac.uk (P. Lane).

https://doi.org/10.1016/j.ipl.2017.09.005
0020-0190/© 2017 Elsevier B.V. All rights reserved.

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 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

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

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.

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.

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

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

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

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