MethodsX 7 (2020) 100916

Contents lists available at ScienceDirect

MethodsX

journal homepage: www.elsevier.com/locate/mex

Method Article

Evidence accumulation clustering using combinations of features

William Wong[a,b,c,*], Naotsugu Tsuchiya[a,b,c]

[a] School of Psychological Sciences and Turner Institute for Brain and Mental Health, Monash University
[b] Center for Information and Neural Networks (CiNet), National Institute of Information and Communications Technology (NICT), Suita, Osaka, Japan
[c] Advanced Telecommunications Research Computational Neuroscience Laboratories, Seika-cho, Soraku-gun, Kyoto, Japan

A B S T R A C T

Evidence accumulation clustering (EAC) is an ensemble clustering algorithm that can cluster data for arbitrary shapes and numbers of clusters. Here, we present a variant of EAC in which we aimed to better cluster data with a large number of features, many of which may be uninformative. Our new method builds on the existing EAC algorithm by populating the clustering ensemble with clusterings based on combinations of fewer features than the original dataset at a time. Our method also calls for prewhitening the recombined data and weighting the influence of each individual clustering by an estimate of its informativeness. We provide code of an example implementation of the algorithm in Matlab and demonstrate its effectiveness compared to ordinary evidence accumulation clustering with synthetic data.

- The clustering ensemble is made by clustering on subset combinations of features from the data
- The recombined data may be prewhitened
- Evidence accumulation can be improved by weighting the evidence with a goodness-of-clustering measure

A R T I C L E I N F O
Method name: Combinatorial evidence accumulation clustering
Keywords: Ensemble clustering, *k*-means clustering, Combination clustering
Article history: Received 24 October 2019; Accepted 3 May 2020; Available online 14 May 2020

* Corresponding author.
E-mail address: William.Wong1@monash.edu (W. Wong).

https://doi.org/10.1016/j.mex.2020.100916

Specifications table

Subject Area	Computer Science
More specific subject area	Unsupervised machine learning
Method name	Combinatorial evidence accumulation clustering
Name and reference of original method	Evidence accumulation clustering: A.L.N. Fred, A.K. Jain, Combining Multiple Clusterings Using Evidence Accumulation, IEEE Trans. Pattern Anal. Mach. Intell. 27 (2005) 835–850.
Resource availability	

Method details

We developed a novel clustering method for a study [15] where we were faced with the problem of clustering a set of data, composed of 54 observations of over 50,000 raw variables, in an unsupervised manner. A smaller number of attributes (approx. 2,500) or "features" were extracted for each observation, but a majority of the features were still unlikely to be informative for clustering—we will call these "dubious feature sets". The lack of ground truth combined with the stipulation to not disregard any of the features *a priori*, for reasons specific to our study, meant that there was a high risk of producing clusters conditioned on the noise rather than on any underlying structure. Our proposed clustering method, which we call **combinatorial evidence accumulation clustering** (or **combination clustering** for short in this article), is a variant of evidence accumulation clustering (EAC) that attempts to mitigate the influence of dubious features in order to obtain better results in unsupervised clustering.

EAC is an algorithm for finding data clusters of arbitrary shapes and numbers [3]. It belongs to the class of consensus, or ensemble, clustering algorithms [4,13] in that it combines the results of multiple individual clusterings of the same data to produce a partitioning solution that can be more accurate than those of the individuals. In EAC, an individual clustering is the result of applying some clustering algorithm to the data with variation imparted on either the algorithm itself or the data representation. Each of these individual clusterings are therefore considered pieces of **evidence** of how data are organised, which the EAC algorithm uses to determine the optimal clustering solution. Here, we shall call the clusterings making up the evidence "**sub-clusterings**" so as not to confuse them for the ultimate clustering step of EAC, which we will call "**evidence accumulation**". In Fred & Jain's implementation, henceforth referred to as **ordinary EAC**, sub-clustering consisted of multiple, randomly-seeded runs of k-means clustering [7]. The evidence is combined and expressed in a similarity matrix called the **co-association matrix** ($\mathcal{C}$), which counts the frequency of data points pairwise occurring in the same sub-cluster in each piece of evidence. The co-association matrix is then hierarchically clustered using the single- or average-linkage criterion, giving the final EAC result: this is the evidence accumulation step.

To address the specific problem of clustering a data set with many dubious features, our combination clustering method incorporated the following improvements to Fred & Jain's [3] EAC method:

a) Sub-clusterings are performed on subset combinations of features of data.
b) Evidence is weighted by a goodness-of-clustering measure of the originating sub-clustering and by the inverse number of total sub-clusterings within its dimensionality.
c) Data in each sub-clustering are prewhitened.

In the next section, we explain the motivation behind these improvements using simple examples.

Motivation

The 1-dimensional case

Given a data set with only one feature (or dimension, as we use here interchangeably), improvement b) is the only applicable factor to consider. Both ordinary EAC and combination

clustering produce a clustering ensemble composed of the results of other clusterings (i.e., sub-clusterings). Let us take an ensemble to be composed of $\mathcal{O}$ number of k-means clusterings, with variation between them given by different parameters and/or initialisations of the clustering algorithm as in Fred & Jain's [3] original description.

When sub-clustering n number of objects, we will represent the result of the l^{th} sub-clustering as an $n \times n$ **similarity matrix,**

$$S_l = (s_l(i, j)) \in \{0, 1\}^{n \times n}, \tag{1}$$

where S_l signifies the entire matrix, and $s_l(i, j)$ (for $i = 1, \ldots, n$ and $j = 1, \ldots, n$) is the entry for object pair i, j in the i^{th} row of the j^{th} column. When there is a co-association between the i^{th} and j^{th} objects due to being sub-clustered together, the corresponding entries of the similarity matrix are given a value of 1 (i.e., $s_l(i, j) = 1$), and otherwise 0 when the objects do not co-associate. The result S_l from sub-clustering l constitutes one piece of evidence. The total number of sub-clusterings is given by $\mathcal{O}$ (thus, $l = 1, \ldots, \mathcal{O}$). We obtain the clustering ensemble E as the set of all similarity matrices resulting from sub-clustering:

$$E = \{S_1, \ldots, S_{\mathcal{O}}\}. \tag{2}$$

In ordinary EAC, the co-association matrix $\mathcal{C}$ is produced by taking the average of the evidence in E. With improvement b), we propose to weight each piece of evidence by a measure of its sub-clustering's **goodness-of-clustering**. The goodness-of-clustering for sub-clustering l shall be given by g_l. Thus, to implement improvement b), the co-association matrix entry at the i^{th} row and j^{th} column will be given by

$$C(i, j) = \frac{1}{\mathcal{O}} \sum_{l=1}^{\mathcal{O}} g_l s_l(i, j). \tag{3}$$

We would expect that, compared to ordinary EAC, our weighted EAC should better cluster the data as long as g_l reflects the true goodness-of-clustering. To understand how this might be the case, consider that some parameters or initialisations of sub-clustering may produce evidence from local minima that are non-conducive to the optimal solution. For such cases, we would want to excise or penalise the evidence depending on how poor it were to be. In other words, if we can estimate the quality of the evidence, conceivably we can downweight poor-quality evidence to boost the quality of the clustering ensemble. We illustrated this concept in Fig. 1.

The choice of method to measure goodness-of-clustering depends on the nature of the clustering problem, and is a discussion outside the scope of this paper. Here, we provisionally present a simple method to measure goodness-of-clustering based on the averaging of silhouette values. The silhouette value for object $\mathfrak{i}$ is defined by Rousseeuw [11] as

$$\mathfrak{s}(\mathfrak{i}) = \frac{\mathfrak{b}(\mathfrak{i}) - \mathfrak{a}(\mathfrak{i})}{\max\{\mathfrak{b}(\mathfrak{i}), \mathfrak{a}(\mathfrak{i})\}}, \tag{4}$$

where $\mathfrak{a}(\mathfrak{i})$ is the average dissimilarity (e.g., the Euclidean distance in feature space) of object $\mathfrak{i}$ to all other members of its cluster, and $\mathfrak{b}(\mathfrak{i})$ is the average dissimilarity of object $\mathfrak{i}$ to all members of the next nearest cluster. $\mathfrak{s}(\mathfrak{i})$ takes a value on the interval $[-1, 1]$, with higher values corresponding to closer association with its own cluster, and negative values corresponding to closer association with the other cluster. Thus, the value of $\mathfrak{s}(\mathfrak{i})$ averaged over all objects $\mathfrak{i} = 1, \ldots, n$ will be high when well-separated clusters exist and cluster memberships are correctly identified. It will be low when clusters are poorly-separated, do not exist, or when cluster memberships are incorrectly assigned.

We measure goodness-of-clustering by taking the mean of these silhouette values over all objects of the data, indexed by $\mathfrak{i}$, and then applying a ramp function to the result (to ensure that sub-clusterings with negative means do not contribute to evidence accumulation), as follows:

$$g_l = \max\left\{\left(\frac{1}{n}\sum_{\mathfrak{i}=1}^{n} \mathfrak{s}_l(\mathfrak{i})\right), 0\right\}. \tag{5}$$

With g_l and s_l, we can now find co-association matrix $\mathcal{C}$ via Eq. 3. Any clustering method may then be applied to $\mathcal{C}$ for the evidence accumulation step: this will produce the algorithm's final clustering result.

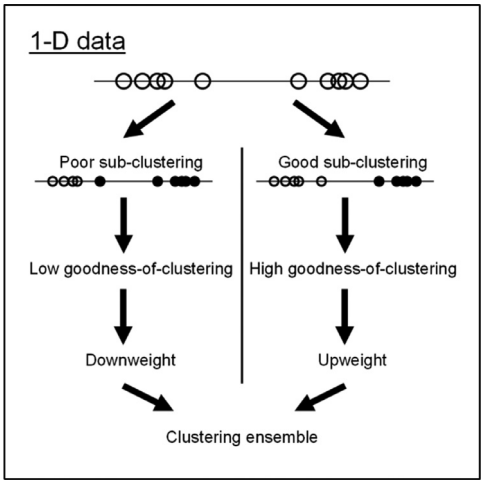

Fig. 1. Application of improvement b) in the 1-dimensional case. A set of data is illustrated as circles, whose coordinates along the real number line represent its value magnitudes for a single feature. In this example, two different sub-clusterings are produced due to using different initial parameters for the sub-clustering subroutine. The clusters are represented by groups of empty or filled circles. The sub-clustering on the left is an example of a relatively poor sub-clustering result, which is estimated by a goodness-of-clustering measure. The clustering ensemble, which is the collection of all sub-clusterings, may therefore be improved by weighting the contribution of individual sub-clusterings by their goodness-of-clustering.

The 2-dimensional case

With data represented by two features, we can explain improvement a) of combination clustering.

Let us first represent the set of all features in the data as $\mathbb{F} = \{f_1, f_2\}$, where f_m is one of those features for $m = 1, 2$. Improvement a) permits sub-clusterings to be performed on subset combinations of features: that is, on f_1 or f_2 alone, or both f_1 and f_2. All possible subsets are given by the power set of $\mathbb{F}$, ignoring the null set; in this case, they are $\mathcal{P}^+(\mathbb{F}) = \{(f_1), (f_2), (f_1, f_2)\}$.

To understand the motivation behind improvement a), consider a data set with two clusters that are well separated in the first dimension (f_1), but randomly mixed in the second (f_2). The second dimension is essentially noise, which adds variance to the data. Since the expected distance separating the clusters in f_1 is not changed from the addition of f_2, the signal-to-noise ratio decreases overall in (f_1, f_2). For ordinary EAC, whose sub-clusterings are all performed within the whole feature space, clustering effectiveness is thereby reduced. In contrast, combination clustering may collect evidence based on subset combinations of the features, which includes the more informative (f_1) combination. This information boost can then be leveraged by the weighting procedure proposed earlier, which by a goodness-of-clustering metric, effectively selects for sub-clusterings with combinations of informative features and against those with uninformative features. In our hypothetical 2-D example, we expect the weighting procedure to automatically put more weight on sub-clusterings using only (f_1) than those using (f_1, f_2), which may only be partly informative. We furthermore expect it to put much less weight on sub-clusterings using only (f_2), as they would only produce spurious, uninformative sub-clusters associated with an ideally low goodness-of-clustering measure. We illustrate this concept in Fig. 2.

Recall that the possible combinations of $\mathbb{F}$ are $\{(f_1), (f_2), (f_1, f_2)\}$. You will notice that there are twice as many combinations of one feature than there are of two. Looking beyond the 2-dimensional case, as the number of features in $\mathbb{F}$ increases, the number of possible combinations of features grows factorially. Among $\mathfrak{n}$ candidate features, the number of ways to choose $\mathfrak{k}$ features without repetition is given by the binomial coefficient $\binom{\mathfrak{n}}{\mathfrak{k}} = \frac{\mathfrak{n}!}{\mathfrak{k}!(\mathfrak{n}-\mathfrak{k})!}$, which is maximal when $\mathfrak{k}$ approaches half of $\mathfrak{n}$. Thus, there can be very large imbalances in the number of sub-clusters for each dimensionality of

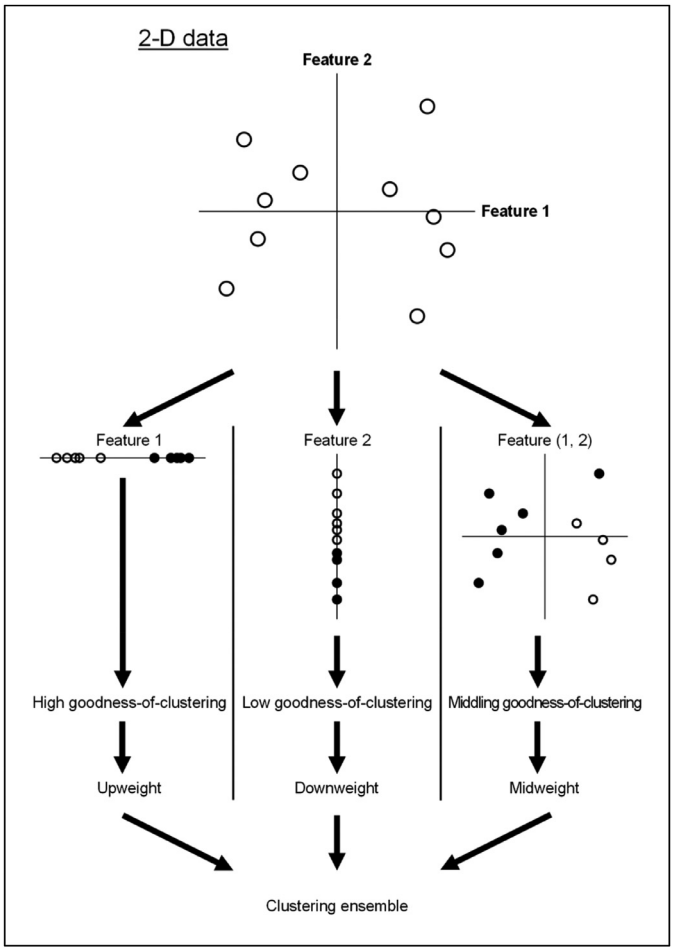

Fig. 2. Application of improvement a) in the 2-dimensional case. A set of data is illustrated as circles in 2-D space. In the application of improvement a), Feature 1, Feature 2, and Feature (1, 2) are used separately by different sub-clusterings. In this example, Feature 1 is informative of the underlying groups in the data while Feature 2 is uninformative. Concordantly, the sub-clustering results from only Feature 1 tend to reflect the underlying groups, while results from only Feature 2 tend to be uninformative. Results from both features tend to be intermediately informative. This improvement contrasts with ordinary EAC in which Feature (1, 2) would be used by all sub-clusterings. As per improvement b) (see Fig. 1), the poor sub-clustering result based on Feature 2 (middle sub-clustering) produces a relatively low goodness-of-clustering measure, which correspondingly downweights the sub-clustering's contribution to the clustering ensemble. Similarly, the sub-clustering based on Feature 1 (left sub-clustering) has a better goodness-of-clustering measure than the other two's, hence it has more influence in the clustering ensemble.

features if they are exhaustively sampled. We can mitigate this problem by proportionately decreasing the contribution of sub-clusterings by the total number of sub-clusterings that has its order of dimensionality—this becomes part of the weighting procedure.

To be methodical, we shall sort the sub-clusterings into sets according to the number of features they use: that is, their **order of dimensionality**. Say that our clustering ensemble is composed of all possible combinations of two features: $\{(f_1), (f_2), (f_1, f_2)\}$. We should group together (f_1) and (f_2) due to their being combinations of one feature: making the set $F_1 = \{(f_1), (f_2)\}$. We call these "first-order combinations" of features. Combination (f_1, f_2) would be called a "second-order combination", and should be put in a separate set as its only member: $F_2 = \{(f_1, f_2)\}$. More generally, we would say

that all k-order combinations used for sub-clustering belong to set F_k, where

$$F_k \subseteq \{f \in p^+(\mathbb{F}) \mid |f| = k\};$$
$$\text{identically } F_k \subseteq \binom{\mathbb{F}}{k}. \tag{6}$$

Thus, when we denote the total number of combinations $\mathcal{O}_k = |F_k|$, we may balance the number of sub-clusterings in proportion to the total number of sub-clusterings that has its order of dimensionality by weighting the contribution of all k-order sub-clusterings by $1/\mathcal{O}_k$.

Sub-clustering using the k-means algorithm with prewhitening

Fred & Jain [3] explored the use of k-means clustering for sub-clustering in EAC, and provided recommendations on the number of sub-clusterings and choice of k to achieve better convergence. We make an additional recommendation to prewhiten the data for each combination of features before sub-clustering.

k-means clustering achieves better results when clusters are described by multiple features that are uncorrelated with each other. However, the empirical data that we aimed to cluster were usually correlated among its features. Prewhitening data is known to result in better clustering with the k-means algorithm [5,8], as it effectively reveals any data relationships arising from the combination of features that is otherwise obscured by their strong correlation. We applied it in our method for each sub-clustering using zero-phase component analysis [1]. This method is also equivalent to performing k-means clustering using Mahalanobis distances [6]. We provide experimental results demonstrating the effectiveness of prewhitening later in this paper.

Description

Now, we describe our method for the general case of data with any number of features. Given a data set of n objects with N number of features, the set of all available features is

$$F = \{f_1, \ldots, f_N\}. \tag{7}$$

We construct a clustering ensemble (E) by sub-clustering on the data in many k-order combinations of $\mathbb{F}$ (for various chosen k). Among all possible k-order combinations of features, we may use a subset of them—numbering $\mathcal{O}_k$—which forms a set F_k (Eq. 6). For each combination, we prewhiten the data before performing sub-clustering on them. Out of practicality, we may consider only feature combinations of chosen order N' and lower (i.e., $k = 1, \ldots, N'$), where $N' < N$ if N is infeasibly large.

Each sub-clustering result, or evidence, is encoded as an $n \times n$ distance matrix,

$$S_{kl} = (s_{kl}(i, j)) \in \{0, 1\}^{n \times n}, \tag{8}$$

where $l = 1, \ldots, \mathcal{O}_k$ (i.e., the l^{th} sub-clustering of the k^{th} order of feature combinations) for $i = 1, \ldots, n$ and $j = 1, \ldots, n$ (i.e., the co-association between objects i and j of the data: which is 1 if they belong to the same sub-cluster, and 0 if not). The set of all evidence is the clustering ensemble

$$E = \begin{Bmatrix} S_{1,1}, S_{1,2}, \ldots, S_{1,\mathcal{O}_1}, \\ S_{2,1}, S_{2,2}, \ldots, S_{2,\mathcal{O}_2}, \\ \vdots \\ S_{N',1}, S_{N',2}, \ldots, S_{N',\mathcal{O}_{N'}} \end{Bmatrix}. \tag{9}$$

Note that E is not a matrix, since $\mathcal{O}_1, \ldots, \mathcal{O}_{N'}$ are not necessarily the same.

Next, we measure the goodness-of-clustering of each sub-clustering in E by its ramped average silhouette value,

$$g_{kl} = \max\left\{\left(\frac{1}{n}\sum_{i=1}^{n} s_{kl}(i)\right), 0\right\}, \tag{10}$$

at corresponding indices k, l.

Finally, we find the co-association matrix as the average of all evidence in E, weighted by their goodness-of-clustering and the inverse of the number of sub-clusterings of the same order.

$$C(i,j) = \frac{1}{N'} \sum_{k=1}^{N'} \frac{1}{\mathcal{O}_k} \sum_{l=1}^{\mathcal{O}_k} g_{kl} s_{kl}(i,j) \tag{11}$$

An implementation

We provide code of an example implementation of combination clustering as a function for the Matlab software platform [14] in Table 1. A summary overview of the algorithm is shown in Fig. 3 with some omissions. Notably, our implementation declares variables for all selected combinations prior to sub-clustering, preallocates memory for ensemble-related variables (e.g., E and g), and parallelises the sub-clustering loop for efficiency.

The function, *CombClust*, takes as arguments X, k, and *combinations*. Argument X should be a matrix describing the data, consisting of n objects with N features, with dimensions $n \times N$. Argument k should be the number of clusters for the desired final clustering result. Argument *combinations* is a logical matrix, with N columns and O rows, specifying the combinations of features used by each sub-clustering. O is the total number of sub-clusterings over the whole ensemble. The N columns

Table 1
Example Matlab implementation of combination clustering. The subfunction "Whiten" by original author Colorado Reed is included.

```
function [clusters, C, E, g, Ns, F] = CombClust( ...
    X, k, combinations, doGood, doWhiten, doWeightOrder)
%COMBCLUST Combination clusterer
%   Example code for performing combination clustering, a variant of
%   evidence accumulation clustering. It clusters based on the ensemble
%   results of multiple "sub-clusters".
%
%   In combination clustering, sub-clustering is performed for combinations
%   of features, and then weighted for evidence accumulation. Also, data
%   for each sub-clustering are pre-whitened.
%
%   Syntax:
%       [clusters, C, E, g, Ns] = COMBCLUST(X, [k, combinations, doGood,
%                                               doWhiten, doWeightOrder])
%
%   Input:
%       X             - Feature set of n objects (matrix [n x N] for N
%                       features)
%       k             - Number of clusters to output (default: 2)
%       combinations  - Combinations of features for each sub-clustering;
%                       one combination/sub-clustering specified per row
%                       (logical matrix [O x N], for O sub-clusters)
%                       (default: exhaustive/random combinations of up to
%                       9th order, up to 1,000 combinations per order, at
%                       least 50 combinations per order)
%       doGood        - Flag for estimating goodness-of-clustering and
%                       weighting by it (logical) (default: true)
%       doWhiten      - Flag for pre-whitening (logical) (default: true)
%       doWeightOrder - Flag for weighting by number of sub-clusterings per
%                       order of dimensionality (logical) (default: true)
%
%   Output:
%       clusters  - Assigned clusters by combination clustering
%       C         - Weighted co-association matrix (matrix [n x n])
%       E         - Sub-clustering co-associations (cell {N' x 1} for N'
%                   orders of dimensionality; cell contents: logical matrix
%                   [n x n x d] for d sub-clusters)
%       g         - Goodness-of-clustering estimates (cell {N' x 1}; cell
%                   contents: vector length d)
%       Ns        - Unique orders of dimensionality (vector length N')
%       F         - Orders of dimensionality of each sub-clustering
```

(*continued on next page*)

Table 1 (*continued*)

```
%                   (vector length N)
%
%   Citation:
%       William Wong and Naotsugu Tsuchiya, "Evidence accumulation
%       clustering using combinations of features," MethodsX (in review).
%
%   Author: William Wong
%   Date: 22 October 2019
%   Copyright and licensing: See bottom of code
%
%   See also KMEANS, CLUSTERDATA.



%% Initialise %%

% Change the following 3 values to override default combination behaviour.
maxOrd = 9;
maxPerOrd = 1000;
minPerOrd = 50;

% Default parameters
if nargin < 6
    doWeightOrder = true; end
if nargin < 5
    doWhiten = true; end
if nargin < 4
    doGood = true; end
if nargin < 3
    combinations = []; end
if nargin < 2
    k = 2; end
if nargin < 1
    error('Not enough arguments!')
end
if isempty(doWeightOrder) || ~isscalar(doWeightOrder)
    warning('Using doWeightOrder = true.')
    doWeightOrder = true;
else
    doWeightOrder = logical(doWeightOrder);
end
if isempty(doWhiten) || ~isscalar(doWhiten)
    warning('Using doWhiten = true.')
    doWhiten = true;
else
    doWhiten = logical(doWhiten);
end
if isempty(doGood) || ~isscalar(doGood)
    warning('Using doGood = true.')
    doGood = true;
else
    doGood = logical(doGood);
end
if isempty(k) || ~isscalar(k) || k < 1
    error('Invalid input k!'), end
if isempty(X) || ~ismatrix(X)
    error('Invalid input X!')
elseif size(X,1) < k
    error('Input k too large!')
else
    X = double(X);
    % We get the number of objects and features in X.
    [n, N] = size(X);
end

% Default combinations (exhaustive/random)
if isempty(combinations) || isscalar(combinations) || ...
    ~ismatrix(combinations) || size(combinations,2) ~= N
    % We default to random combinations of features up to 9 dimensions, up
    % to 1,000 combinations per dimensionality.
    if not(maxOrd == 9 && maxPerOrd == 1000 && minPerOrd ~= 10)
```

(*continued on next page*)

Table 1 (*continued*)

```
        warning(['Default combination behaviour overriden:\nNow using ' ...
            'random combinations of up to %u features, up to %u ' ...
            'combinations per dimensionality, at least %u ' ...
            'combinations per dimensionality.'], ...
            maxOrd, maxPerOrd, minPerOrd)
    else
        warning('Using default behaviour for combinations.')
    end
    maxOrd = min(maxOrd, N);
    combinations = logical([]);
    for i = 1:maxOrd
        % Loop through the unique dimensionalities (i)
        thisDims = nchoosek(N, i);
        if thisDims > maxPerOrd
            % Choose random combinations
            thisComb = NaN(maxPerOrd, i);
            for j = 1:maxPerOrd
                thisComb(j,:) = randperm(N, i);
            end
        elseif thisDims < minPerOrd
            % Choose all combinations and fill with random combinations up
            % to minimum
            extraDims = minPerOrd - thisDims;
            thisComb = [nchoosek(1:N, i); NaN(extraDims, i)];
            for j = thisDims + 1 : minPerOrd
                thisComb(j,:) = randperm(N, i);
            end
        else
            % Choose all combinations
            thisComb = nchoosek(1:N, i);
        end
        % Append to 'combinations'
        thisOffset = size(combinations,1);
        combinations = [combinations; false(size(thisComb,1), N)];
        for j = 1:size(thisComb,1)
            combinations(thisOffset + j, thisComb(j,:)) = true;
        end
    end
elseif ~islogical(combinations)
    warning('Converting input combinations to logical array.')
    combinations = logical(combinations);
end

% The order of each sub-clustering
F = sum(combinations, 2);
% Unique orders
Ns = unique(F);
% The number of unique orders
Nprime = length(Ns);
% Index of sub-clusterings for each order
I = cell(Nprime,1);
% The number of sub-clusterings for each order
O = NaN(Nprime,1);
for i = 1:Nprime
    I{i} = find(F == Ns(i));
    O(i) = length(I{i});
end
% The goodness-of-clustering estimates
g = cell(Nprime,1);
% The clustering ensemble (sub-clustering co-associations)
E = cell(Nprime,1);


%% Sub-cluster %%
% We do all sub-clusterings and estimate their goodness-of-clusterings.

for i = 1:Nprime
    % Loop through the unique dimensionalities (i)
    thisI = I{i};
    if doGood, thisG = NaN(O(i),1); end
    thisS = false(n,n,O(i));
```

(*continued on next page*)

Table 1 (*continued*)

```
    parfor j = 1:O(i)
        % Parallel loop through each combination (j)
        thisX = X(:,combinations(thisI(j),:));
        c = SubClusterer(thisX, k, doWhiten);
        if doGood, thisG(j) = GoodEst(thisX, c); end
        thisS(:,:,j) = c == c';
    end
    if doGood, g{i} = thisG; end
    E{i} = thisS;
end


%% Evidence accumulation %%
% We find the co-association matrix as the weighted sum of the clustering
% ensemble, then perform hierarchical clustering on it.

% The co-association matrix
C = zeros(n);
for i = 1:Nprime
    % Loop through the unique dimensionalities (i)
    for j = 1:O(i)
        % Loop through each combination (j)
        if doGood
            if doWeightOrder
                C = C + E{i}(:,:,j) * g{i}(j) / O(i);
            else
                C = C + E{i}(:,:,j) * g{i}(j);
            end
        else
            if doWeightOrder
                C = C + E{i}(:,:,j) / O(i);
            else
                C = C + E{i}(:,:,j);
            end
        end
    end
end
if doWeightOrder
    C = C / Nprime;
else
    C = C / sum(O);
end

% Ensemble clustering using hierarchical clustering
clusters = cluster(linkage(squareform(max(C(:)) - C), 'average'), ...
    'maxclust',k);

end



%% Subfunction - Sub-clusterer %%

function c = SubClusterer(x, k, doWhiten)
% This is the function for sub-clustering data for a combination of
% features. It uses the k-means clustering algorithm for k(+) clusters; and
% may do prewhitening.

if doWhiten
    x = Whiten(x); end
% We randomly choose between k and k+1 for more robust results (see Fred &
% Jain, 2005)
k = k + randi(2) - 1;

c = kmeans(x, k, 'start','sample');

end



%% Subfunction - Goodness-of-clustering estimator %%
```

(*continued on next page*)

Table 1 (*continued*)

```
function g = GoodEst(x, c)
% This is the function for estimating the goodness-of-clustering of a
% sub-clustering. We use the clipped mean silhouette for this.

g = max(0, mean(silhouette(x, c)));

end



%% Subfunction - Whiten (by Colorado Reed) %%

function out = Whiten(in)
%{
Original author: Colorado Reed (colorado-reed@uiowa.edu)

Copyright (c) 2012, Colorado Reed
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

* Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in
the documentation and/or other materials provided with the distribution

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
%}

mu = mean(in);
X = bsxfun(@minus, in, mu);

A = X'*X;
[V,D,~] = svd(A);

% Note hard-coded fudge factor
whMat = sqrt(size(X,1)-1) * V * sqrtm(inv(D + eye(size(D)) * 1e-4)) * V';

out = X*whMat;

end



%% MIT License %%
%
% Copyright (c) 2019 William Wong
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal in the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
%
% The above copyright notice and this permission notice shall be included
```

(*continued on next page*)

Table 1 (*continued*)

```
% in all copies or substantial portions of the Software.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN
% NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
% DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
% OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE
% USE OR OTHER DEALINGS IN THE SOFTWARE.
```

correspond to the N features extracted from data X. Each row of this matrix is a logical vector that specifies which features of X constitute a combination to be used for one sub-clustering. For example, a first-order combination of features would be specified by a row containing one entry that is *True*, and all other entries that are *False*.

CombClust takes three further arguments: Boolean flags *doGood*, *doWhiten*, and *doWeightOrder*. They respectively control whether to perform weighting by goodness-of-clustering, prewhitening, and weighting by the number of sub-clusterings within each order of dimensionality of their combination of features.

CombClust returns variables *clusters*, *C*, *E*, *g*, and *Ns*. Variable *clusters* is a vector of size n that assigns integer labels to each object representing the final clustering result of the algorithm. C is the final co-association matrix (sized $n \times n$) resulting from evidence accumulation and weighting. *Ns* is a vector listing all orders of dimensionalities used in the clustering ensemble; let us use N' to denote its length. E contains the evidence resulting from the ensemble of all sub-clusterings. It is an $N' \times 1$ cell array where the l^{th} cell corresponds to an order of dimensionality given by $Ns(l)$ and contains the evidence from all sub-clusterings of that order. This evidence is conditioned as an $n \times n \times \mathcal{O}_l$ logical array, formed as the concatenation of all applicable $n \times n$ distance matrices of logical type in the third dimension. Variable g is also an $N' \times 1$ cell array, containing the goodness-of-clustering values from the ensemble for each order of dimensionality, conditioned as vectors.

The sub-clustering routine uses Matlab's *kmeans* function for the k-means clustering with initial seeds randomly chosen from the sample. The number of sub-clusters k is also chosen at random between the user-given k and user-given $k+1$; according to Fred & Jain [3], such variation increased the robustness of the ensemble solution.

We also set a default behaviour for combination selection based on exhaustive/random selection. This particular strategy was used because sub-clustering of all possible combinations of features is computationally infeasible for large numbers of features. Thus, the strategy selected all possible k-order combinations for tractable values of k, and a limited number of combinations for intractable values of k. To prevent a systematic bias in the latter case, those combinations were selected randomly. By default, all possible combinations of features would be selected up to and including the 9^{th} order, with a maximum of 1,000 combinations per order and a minimum of 50. For orders whereof the number of possible combinations exceeds this limit, 1,000 would be selected randomly. For those whereof the number subceeds the limit, additional combinations would be selected at random to make up 50. For comparison, Fred & Jain [3] reported being able to achieve convergent results for simple data sets using 50 sub-clusterings, and for complex data sets using 200 sub-clusterings.

In the evidence accumulation step, the co-association matrix is converted to a distance matrix and clustered using hierarchical clustering with the average linkage criterion [12] until k clusters are produced.

Experimental results

We tested the effectiveness of our proposed method in four experiments. In the first experiment, we compared the clustering performance of combination clustering to ordinary EAC; in the second experiment, we made a broader comparison between the two algorithms for a range of

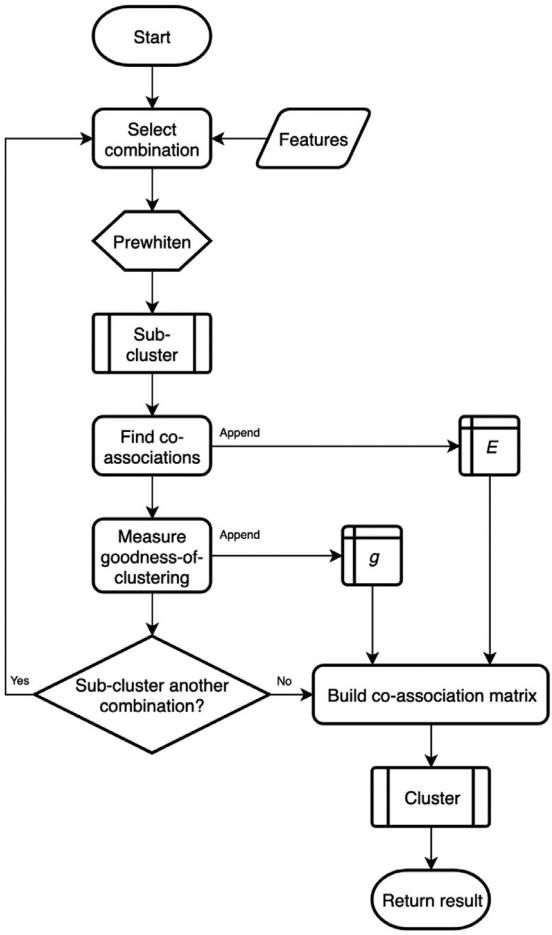

Fig. 3. Conceptual flowchart of a combination clustering algorithm. Each pass through the primary loop adds another sub-cluster to the ensemble (E) with a corresponding goodness-of-clustering measure (g). This loop may be paralellised.

signal-to-noise (SNR) distributions among features of the data; in the third experiment, we contrasted the
performance of combination clustering with and without the prewhitening procedure; and in the fourth experiment, we contrasted the performance of combination clustering with and without the evidence weighting factors.

Our experiments were performed with synthetic data, generated to represent data sets of varying numbers of features and SNR distributions. We used the Matlab implementation of combination clustering, given in Table 1, and its default parameters throughout the experiments unless stated otherwise. Clustering performance was measured as normalised mutual information—also used in Fred & Jain [3]—but between the clustered groups and the true groups. Its formula is given in Eq. 4 of their paper.

For the first experiment, we compared the performance of combination clustering to ordinary EAC with respect to the number of features of the data set being clustered. We wrote a Matlab implemention for ordinary EAC based on Fred & Jain's [3] description, whose evidence also derived from k-means clustering. Here, the data were sub-clustered verbatim over 1,000 runs with randomised seeds; the values of k were chosen between the user-given k and user-given $k+1$, the same as in

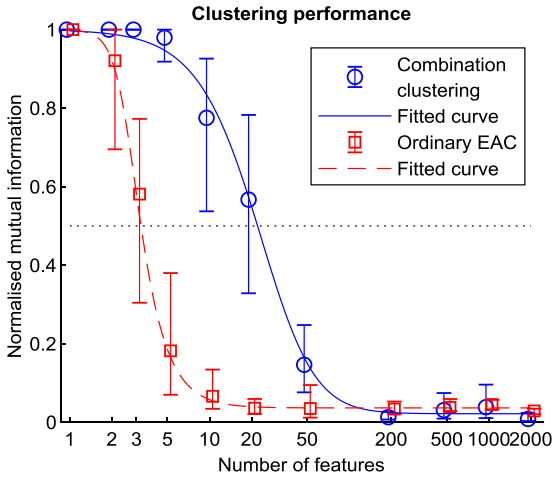

Fig. 4. Comparison of the performance between the original EAC and combination clustering. As the number of noisy features increased, both the original method (red squares, dotted line) and our method (blue circles, solid line) degraded in performance, measured by normalised mutual information. Our proposed method performed significantly better than the old method for numbers of features in the range 3–50 (one-tailed sign tests, $W(10) \geq 9$, $p_{\text{adj}} < .024$, Benjamini–Hochberg correction). Error bars are bootstrapped 95% confidence intervals; curves are fitted generalised logistic functions [10].

our combination clustering implementation. The evidence accumulation step was also identical. The total number of sub-clusterings performed is about an order of magnitude larger for the combination clustering than ordinary EAC. This difference was also reflected in their computational times (see supplementary material Table S2). In our environment,[1] the typical computational times for data of 50 objects with 300 features was 0.5 seconds for ordinary EAC and 6.7 seconds for combination clustering.

Data were generated as consisting of one informative feature that separated two underlying groups of objects (each $n = 25$) with a 10:1 SNR, and the remaining features having zero SNR. All the features were normalised to have zero mean and variance of one; noise components were generated as independent Gaussian processes with zero covariance between the components. We tested the performance of both clustering algorithms over 11 feature set sizes ranging from 1 to 2,000—each randomly generated 10 times. We plot the average of their performance in Fig. 4. The curves here and throughout the experiments were generalised logistic functions, fitted using the trust-region method [9], with a fixed upper asymptote of 1, and where the independent variable was the number of features (log-transformed).

As may be expected, clustering performance generally decreased with increasing number of features carrying no clusterable information. The performance of our proposed method outperformed ordinary EAC over a larger range of set sizes: statistically significantly over the range 3–50 features (one-tailed, paired sample sign tests, $W(10) \geq 9$, $p_{\text{adj}} < .024$, Benjamini–Hochberg correction).

In the next experiment, we repeated our initial experiment with differently generated data sets. Here, we generated data sets with a distribution of SNRs among the features following Zipf's law. Zipf's law originally related the frequency of words in a corpus of natural language as inversely proportional to its rank, but has since been found to describe many other real-world observations [2,16]. Its discrete probability distribution may be given by

$$Z(k; s, N) = \frac{k^{-s}}{\sum_{n=1}^{N} (n^{-s})}, \tag{12}$$

[1] A PC workstation with one CPU (Intel Core i7-6700, 4 physical or 8 logical cores at 3.40 GHz) and 16 GB of RAM, running with MATLAB (R2018a Update 4) with a parallel pool of 8 local workers in the Windows 10 Enterprise (64-bit) operating system. One GPU was installed but not used.

Table 2
Combination clustering advantage over ordinary EAC for varying Zipf exponent (s). For each tested s, the proportion of data sets that resulted in higher normalised mutual information by the combination clustering algorithm is given, after collapsing over all feature set sizes and discarding tied performances.

s	Proportion better (%)	Sign test statistic (paired sample, two-tailed)	p
0	52	$W(50) = 26$	.888
1	49	$W(49) = 24$	1
2	60	$W(60) = 36$	.155
3	66	$W(73) = 48$	.010*
4	65	$W(81) = 53$	.007*
∞	64	$W(87) = 56$	.010*

where, in this context, k is the SNR rank of the features, N is the total number of features, and the term of the denominator is a normalisation factor. Parameter s controls the steepness of the SNR distribution as a function of k. When $s = 0$, all features are equally informative. When $s = \infty$, only 1 feature is informative, equivalent to the situation in the first experiment. Using this probability mass function, we fixed the feature-wise SNR distribution of our generated data sets to Zipfian distributions with varied s and N.

Like in the first exerpiment, we explored 11 values of N ranging from 1 to 2,000. In addition, we also systematically varied $s = 0, 1, 2, 3, 4, \infty$. Thus, 66 parameter combinations were used to generate the data sets.

The results of the second experiment are summarised in Fig. 5. Note that the first experiment's results are replotted in the curves for where $s = \infty$. In general, we observed that combination clustering performance was superior to ordinary EAC's for larger numbers of features (statistics given in Table 2), and this advantage diminishes for smaller s. These results are consistent with our intention of making EAC more robust to data that have many uninformative features.

We additionally observed that the rate of clustering failure with respect to the number of features was slower in combination clustering compared to ordinary EAC, as signified by the shallower slope of the fitted curves belonging to the combination clustering condition. We numerically computed and compared the maximum slopes of each curve. In this way, we found that all six slopes of the tested s parameters, except for $s = 4$, were shallower for combination clustering than ordinary EAC, and the mean slope difference was statistically significant (one-tailed, paired sample t-test, $t(5) = 2.14$, $p = .043$). This suggested that some residual clustering of the true underlying groups was exhibited by combination clustering at numbers of features where ordinary EAC had practically reached asymptotic minimum (e.g., at 500 features for $s = 0$, Fig. 5).

For the third experiment, within the combination clustering method, we looked at the effect of prewhitening the recombined data of each sub-clustering. In the previous experiments, because there was no noise covariance between the feautres, we would not expect prewhitening have much effect on clustering performance (see supplementary material Figure S1). Therefore, for this experiment, we generated data sets that do have correlated noise between the features.

We controlled the covariance in our data sets with a five-step procedure. First, we generated N independent noise components like in previous experiments. Second, we scaled the relative variances of the noise components to follow a Zipfian distribution ($s = 1$). At this step, the resulting noise covariance matrix would have diagonal values given by $Z(k; 1, N)$, and zeroes in all other entries. Third, we applied a random rotation to the matrix to introduce non-zero covariance between the features. Fourth, we normalised their means to zero and variances to one. Fifth, we added the signal component as in the first experiment, and then normalized this final set of features.

We compared the clustering performance of combination clustering over the same range of feature set sizes as previous experiments, with and without prewhitening, by setting the *doWhiten* flag of our method implementation accordingly. We randomly generated 30 sets of each number of features instead of 10 (like in the previous experiments). We plot the average of their clustering performance in Fig. 6.

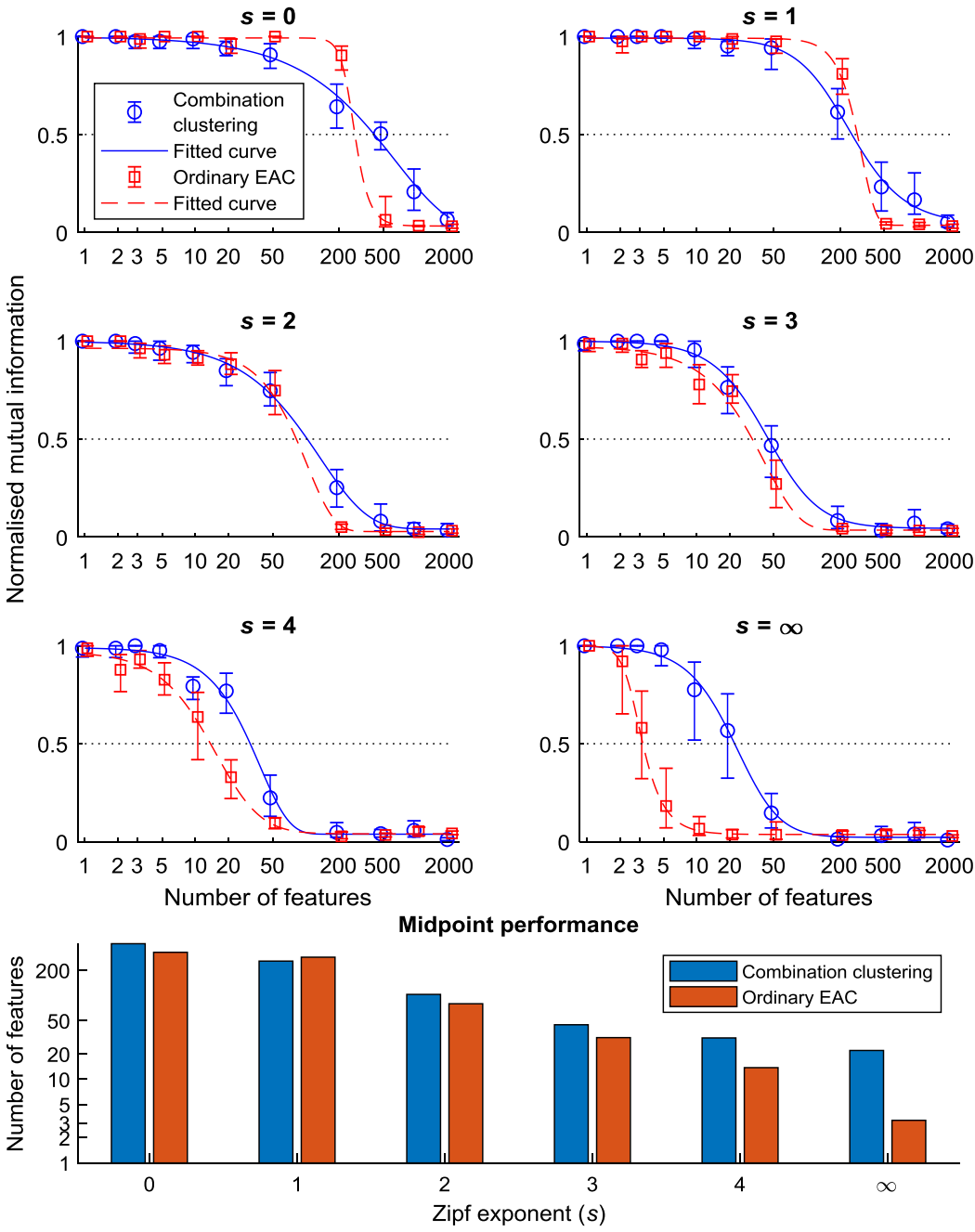

Fig. 5. The performance of combination clustering and ordinary EAC for various SNR distributions via the Zipf exponent (s). The results are identical to Fig. 4 where $s = \infty$. Error bars are bootstrapped 95% confidence intervals; curves are fitted generalised logistic functions [10]. The bottom panel plots the number of features where the fitted curves reached 0.5 normalised mutual information, for each clustering algorithm and Zipf exponent.

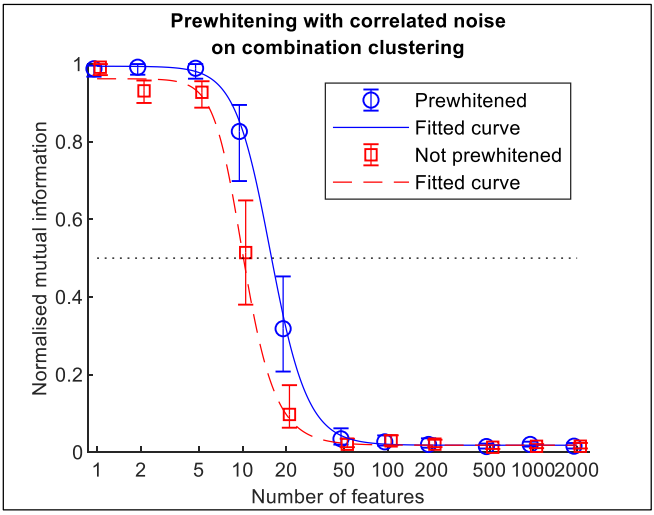

Fig. 6. Effect of prewhitening within combination clustering for dubious data sets with correlated noise. Prewhitening generally improved clustering performance; this was statistically significant for feature sizes 2–10 (one-tailed, paired sample sign tests, $p_{\text{adj}} \leq .014$, Benjamini–Hochberg correction).

For dubious feature sets with correlated noise, we found that prewhitening of the sub-clustering data did produce generally improved clustering performance, most notably for feature set sizes of 2–20. The difference was statistically significant for sizes of 2–10 features (one-tailed, paired sample sign tests, $p_{\text{adj}} \leq .014$, Benjamini–Hochberg correction).

For our fourth and final experiment, we examined the effect of evidence weighting on clustering performance. We generated data sets in the way that we did in the first experiment, but randomly generated 20 sets of each number of features rather than 10. Through the setting of the *doGood* and *doWeightOrder* flags of our combination clustering implementation, we clustered over four conditions: with no weighting ("none"), with weighting by "order of dimensionality", with weighting by "goodness-of-clustering", and with weighting by both order of dimensionality and goodness-of-clustering ("both"). We plot the average of their resulting performance in Fig. 7.

We observed a small difference between the weighting conditions after factoring out feature set size (Friedman test, $\chi^2(3) = 10.1$, $p = .018$). The largest apparent difference was seen for data sets of 10 features (Friedman test, $\chi^2(3) = 28.7$, $p < .001$). In both views, the "none" condition had the lowest median performance; and two conditions, "goodness-of-clustering" and "both", had higher median performances than the remaining conditions, "none" and "order of dimensionality". These results suggest that weighting by goodness-of-clustering improves clustering performance, and weighting by order of dimensionality neither improves nor worsens the performance—at least for these generated data sets. Given our original reasoning for weighting by order of dimensionality, we suspect it may still be useful for other types of data sets; however, this matter shall have to be left for a future investigation.

Final remarks

We have proposed, and given an implementation for, a variant of EAC that was designed to handle dubious data consisting of many features that are uninformative to clustering. Our experiments have shown that our proposed method is superior to ordinary EAC for a range of data set characteristics—particularly when informative components are concentrated in only a few of the features, and for larger feature set sizes.

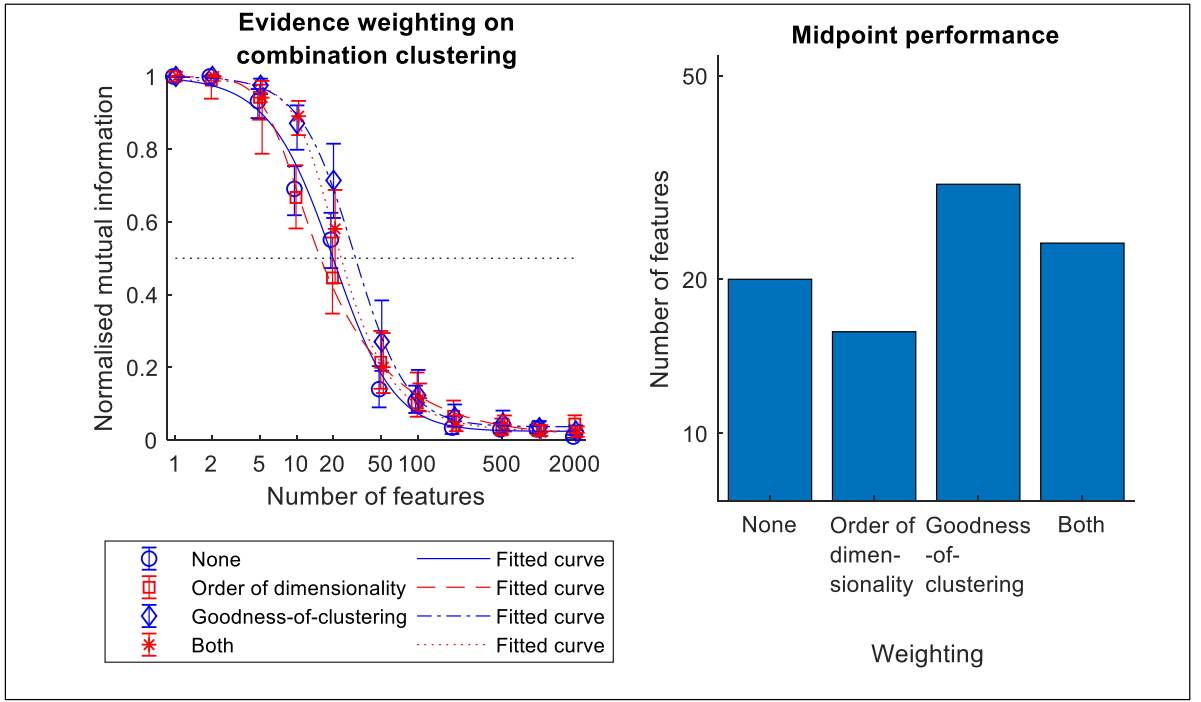

Fig. 7. Effect of evidence weighting on clustering performance. Weighting by "goodness-of-clustering" and "both" resulted in better clustering performance most notably for data sets consisting of 10 features (Friedman test, $\chi^2(3) = 28.7$, $p < .001$). The right panel plots the number of features where the fitted curves reached 0.5 normalised mutual information, for each weighting condition.

Acknowledgements

WW was supported by an Australian Government Research Training Program Scholarship. NT was funded by an Australian Research Council Future Fellowship (FT120100619) and Discovery Project grants (DP130100194, DP180104128, DP180100396).

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.mex.2020.100916.

References

[1] A.J. Bell, T.J. Sejnowski, The "independent components" of natural scenes are edge filters, Vis. Res. 37 (23) (1997) 3327–3338.
[2] A. Clauset, C. Shalizi, M. Newman, Power-Law Distributions in Empirical Data, SIAM Rev. 51 (4) (2009) 661–703.
[3] A.L. Fred, A.K. Jain, Combining multiple clusterings using evidence accumulation, IEEE Trans. Pattern Anal. Mach. Intell. 27 (6) (2005) 835–850.
[4] A. Goder, V. Filkov, Consensus clustering algorithms: comparison and refinement, in: Proceedings of the Meeting on Algorithm Engineering & Experiments, Society for Industrial and Applied Mathematics, Philadelphia, 2008, pp. 109–117.
[5] A.K. Jain, Data clustering: 50 years beyond K-means, Pattern Recognit. Lett. 31 (8) (2010) 651–666.
[6] A. Kessy, A. Lewin, K. Strimmer, Optimal Whitening and Decorrelation, Am. Stat. 72 (4) (2018) 309–314.
[7] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, University of California Press, Berkeley, 1967, pp. 281–297.
[8] J. Mao, A.K. Jain, A self-organizing network for hyperellipsoidal clustering (HEC), IEEE Trans. Neural Netw. 7 (1) (1996) 16–29.
[9] J.J. Moré, D.C. Sorensen, Computing a Trust Region Step, SIAM J. Sci. and Stat. Comput. 4 (3) (1983) 553–572.
[10] F.J. Richards, A Flexible Growth Function for Empirical Use, J. Exp. Bot. 10 (2) (1959) 290–301.
[11] P.J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, J. Comput. Appl. Math., 20 (1987) 53–65.
[12] R.R. Sokal, C.D. Michener, A statistical method for evaluating systematic relationships, Univ. Kans. Sci. Bull. 38 (2) (1958) 1409–1438.
[13] A. Strehl, J. Ghosh, Cluster Ensembles – A Knowledge Reuse Framework for Combining Multiple Partitions, J. Mach. Learn. Res., 3 (2002) 583–617.
[14] The MathWorks, Inc. (2018). MATLAB (R2018a). Natick, Massachusetts, United States.
[15] W. Wong, V. Noreika, L. Móró, A. Revonsuo, J. Windt, K. Valli, N. Tsuchiya, The Dream Catcher experiment: Blinded analyses disconfirm markers of dreaming consciousness in EEG spectral power, 2019, doi:10.1101/643593.
[16] Zipf, G. K. (1949). *Human Behavior and the Principle of Least Effort*. Cambridge: Addison-Wesley Press, Inc.