Coarse-grain cluster analysis of tensors with application to climate biome identification

IOP Publishing Mach. Learn.: Sci. Technol. 1 (2020) 045020 https://doi.org/10.1088/2632-2153/abb676

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
22 January 2020

REVISED
29 June 2020

ACCEPTED FOR PUBLICATION
2 September 2020

PUBLISHED
22 October 2020

PAPER

Coarse-grain cluster analysis of tensors with application to climate biome identification

Derek DeSantis, Phillip J Wolfram, Katrina Bennett and Boian Alexandrov

IOP Publishing, Temple Circus, Temple Way, Bristol BS1 6HG, United Kingdom

E-mail: ddesantis@lanl.gov

Keywords: machine learning, climate biomes, wavelet, information theory, clustering, interpretable machine learning

Abstract

A tensor provides a concise way to codify the interdependence of complex data. Treating a tensor as a d-way array, each entry records the interaction between the different indices. Clustering provides a way to parse the complexity of the data into more readily understandable information. Clustering methods are heavily dependent on the algorithm of choice, as well as the chosen hyperparameters of the algorithm. However, their sensitivity to data scales is largely unknown.

In this work, we apply the discrete wavelet transform to analyze the effects of coarse-graining on clustering tensor data. We are particularly interested in understanding how scale affects clustering of the Earth's climate system. The discrete wavelet transform allows classification of the Earth's climate across a multitude of spatial-temporal scales. The discrete wavelet transform is used to produce an ensemble of classification estimates, as opposed to a single classification. Each element of the ensemble is a clustering at a different spatial-temporal scale. Information theoretic approaches are used to identify important scale lengths in clustering the L15 Climate Datset. We also discover a sub-collection of the ensemble that spans the majority of the variance observed, allowing for efficient consensus clustering techniques that can be used to identify climate biomes.

1. Introduction

Data measured from a high-order complex system can be difficult to analyze. A convenient tool to store this data is in the form of a tensor, or d-way array. Each entry of the array describes the value obtained across the *d* parameters. Often, the dependencies between indices is not clear, making direct interpretation of the data a demanding task.

Numerous methods have been developed to allow one to readily parse these complex interdependencies to provide meaningful interpretations of the data. Among the most well studied methods are tensor factorizations and clustering techniques. Different forms of tensor factorizations have been shown to be effective at multi-way dimensionality reduction, blind source separation, data mining, and latent feature extraction [1–3]. Indeed, non-negative tensor factorizations, a constrained tensor factorization requiring the data and each factorized component to consist of non-negative real numbers, has had success at extracting hidden, interpretive features of the data [4–8]. This is because many types of real-world data are naturally non-negative, so the enforced positivity constraint on the factors can be interpreted as a mixing of physical signals [9].

While the goal of factorization techniques are often to discover latent structures within the data, the aim of clustering is to provide coherent groupings of objects. When data comes in the form of vectors (order-1 tensors), there is an immense, ever growing lexicon of clustering methods [10]. For clustering general tensors there is a larger flexibility of techniques due to the interactions along different fibers of the tensors. This leads to more complex clustering problems, requiring new optimization algorithms to approximately solve for the desired clustering [11, 12].

These two goals of factorization and clustering are often deeply related. For example, it has been shown that by adding constraints to non-negative matrix factorization (order-2 tensor factorization), one can produce an optimization function identically to K-means or spectral clustering (order-1 clustering) [13].

Similarly, there is an equivalence between higher order singular value decomposition and a K-means clustering for tensors with additional constraints [14].

1.1. Classifying climate biomes

We are interested in leveraging the unsupervised learning techniques discussed above to assist in the interpretation of climate data. Climate data generally arises as a collection of spatial-temporal measurements of various physical and biological features of The Earth system (e.g. temperature, precipitation, flora and fauna). The tensor of climate data compactly records the complex interdependence between space and time for different variables of interest. Recently, tensor factorization techniques have found success at extracting latent climate signals. For example, see [15, 16]. Clustering the climate into locally similar spaces is a classical problem with a longer history.

A *biome* is a region of space that has homogeneous climate features, e.g. similar temperature profiles. The process of finding climate biomes is clustering, and a biome is a synonym for a cluster of spatial points on the Earth. Throughout this text, we use the word biome and cluster interchangeably. Historically, the standard to classify climate biomes has been the Köppen–Geiger (KG) model [17]. The KG model is an expert-based judgment that describes climate zones using temperature and precipitation measurements. The KG model utilizes a fixed, expert opinion-based decision tree, where each branch uses various information about temperature and precipitation. This heuristic allows one to broadly assess climate regions.

While KG is interpretable, it is overly simplistic and somewhat arbitrary. In an attempt to remedy this problem, Thornthwaite [18] introduced a more nuanced model using moisture and thermal factors. However, the Thornthwaite model (along with its successors) still suffer from expertly chosen biases in their parameters.

A solution to this problem is to move towards data-driven methods of classification. Here, human bias is subsummed onto the machine learning algorithm that seeks to minimize some cost function. This is equivalent to a statistical assumption about the data generation and distribution [19]. In the views of the authors, this is often a more reasonable assignment of bias. In [20], Zscheischler *et al* compare KG to the K-means algorithm. They show that, unsurprisingly, K-means outperforms KG with respect to statistical measures, specifically explained predict and variance and variation of information. In [21], the authors use mean monthly climate data to perform hierarchical clustering and partition around medoids. In each clustering algorithm, two distance metrics are tested, and these results are compared to KG using an information-theoretic measure.

These data driven approaches to climate clustering are an epistemological improvement over the user chosen heuristics of KG. However, these clustering methods suffer from challenges external to the climate science application space.

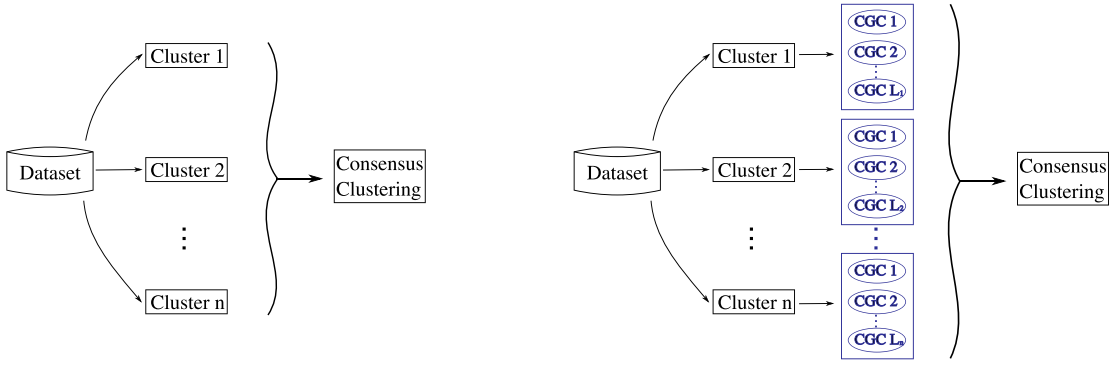
1.2. Clustering challenges

Abstractly, a *clustering* is any function that takes a dataset $X = \{x_n\}_{n=1}^N$ and returns a partitioning of X. One usually seeks a clustering that satisfies a chosen heuristic. For example, a K-means clustering is any partition of X into disjoint non-empty sets $U_1, \ldots U_K$ that minimizes inner cluster variance. Namely,

$$\text{Argmin}_{U_1,\ldots U_K} \sum_{k=1}^{K} \sum_{x_n \in U_k} \|x_n - m_k\|^2 \tag{1}$$

where m_k is the mean of U_k. The number of possible ways to put the N data points into K clusters is $\binom{N-1}{K-1}$. For $N \gg K$, it becomes infeasible to search for the optimal K-means clustering directly. Indeed, finding the optimal solution to equation (1) is NP-hard [22]. As a result, optimization schemes such as Lllyod's algorithm have been developed to approximate the optimal solution. These algorithms are necessarily nondeterminstic, and therefore may fail to adequately approximate the global minimum if the data is not sufficiently nice, e.g. low signal to noise ratio [23].

The above example of K-means summarizes an issue persistent across many approaches to clustering. A heuristic is chosen to group the data, but unfortunately finding the cluster(s) that satisfy this heuristic is computationally intractable. As a result, algorithms are developed to efficiently compute an approximation to these optimal solutions. Because these clustering methods cannot guarantee convergence to the optimal solution for all datasets, different clustering measures have been formulated to assess the quality of a clustering. However, often the measures are directly exported from the optimization functions used in the clustering algorithm. The algorithm that is designed to optimize this clustering measure will, by design, out perform other clustering methods with respect to that metric. As a result, this provides no further information as to what clustering strategy is better suited for the problem.



(a) Consensus Clustering: Each clustering arises from choice of different clustering algorithm or different hyper-parameters.

(b) Consensus Clustering + CGC & MIER: From each clustering, we propose to run CGC and the MIER algorithm. This results in a larger ensemble that will minimally capture the dependence of the data on scale.

Figure 1. Proposed augmentation to the consensus clustering workflow.

These challenges highlight that there is no true 'best' clustering in general. Rather, there are many ideal clusterings that arise from the specifics of the scientific inquiry pursued. None of the optimal clusterings is certifiably 'correct', but each provides different insights into the structure of the data. Unfortunately, the quality of the obtained clustering is not easy to evaluate.

These problems have led researchers to define the concept of an ensemble, or consensus clustering [24]. Here, many clusterings are combined to produce a single clustering of the data. Common features between the clusterings are amplified, and artifacts become dulled (figure 1(a)). There is evidence to suggest that selecting a smaller ensemble with good, diverse clusters outperforms larger, redundant ensembles [25–28]. For this reason, we believe it is preferable to find a smaller ensemble of quality, diverse clusters.

1.3. Our contributions

There are still unresolved issues not addressed by the current ensemble clustering framework. The choice of a heuristic for any single clustering is a proxy for a potentially ill-specified human objective. This ignores other aspects of the environment, weighting an indifference to other potentially important environmental variables [29]. Consequently when a user can identify potential problems with their ML architecture, they need to resolve the effect the issue has on the overall result.

One practical example in the climate sciences is the scale at which the data is acquired. Most natural or environmental data is formed by directly observing and measuring quantities where the underlying or driving processes are usually unknown. The hidden or latent features of the data may not clearly present themselves at the resolution that the data was sampled. For example, weather data is often gathered at fine temporal and spatial detail, e.g. daily temperature at a single weather station. However, climate signals are often observed on the order of years or decades and across a region.

The above discussion highlights two important problems:

(a) The need to address hidden parameters affecting the data, such as scale dependence
(b) The desire to build an ensemble of clusters for a consensus clustering algorithm.

In this work, we discuss a clustering workflow that tackles both of these problems. We develop a technique that uses the discrete wavelet transform to cluster slices of tensors at different scales that we call *coarse-grain clustering* (CGC). This highly parallelizable technique results in many potential clusterings, one for each chosen coarse-graining. Not all of these coarse-grain clusterings provide new information, however. Thus, we present a novel selection method that leverages mutual information between clusterings to quantify the loss of information between clusterings and select a small subset that best represents ensemble. We call this reduction algorithm *Mutual Information Ensemble Reduce* (MIER).

While the end-to-end workflow we have discussed involves ideas from traditional consensus clustering (figure 1(a)), the focus of this paper is specifically on a novel modification to this approach leveraging the CGC and the MIER algorithms. Specifically, we are interested in the effects coarse-graining has on clustering,

and desire to bolster our ensemble through the addition of distinct clusterings (figure 1(b)). This paper focuses on this workflow, by analyzing

(a) The effect scale can have on a clustering algorithm (via CGC algorithm and mutual information) and
(b) Automatically identify a small subset of the coarse-grain clusterings that span the observations (the MIER algorithm).

This paper is organized as follows. First background material used for development of the CGC and MIER algorithms is presented in section 2. The structure of these algorithms is detailed in section 3. In section 4, the algorithm is applied to a widely-used climate data set as a case study with presentation of results and discussion, followed by conclusions and a recommendation for future work in section 5.

2. Preliminaries

In this section, we briefly review key mathematical tools used throughout this work including 1) the discrete wavelet transform and its role in separating earth system data into spatio-temporal scales, 2) graph cuts and their connection to spectral clustering, and 3) use of mutual information to measure similarity between two clusters of the same data.

2.1. Discrete wavelet transform (DWT)

Given a one-dimensional function $f: \mathbb{N} \to \mathbb{R}$ the *discrete wavelet transform* (DWT) is a process of iteratively decomposing f into a series of low and high frequency signals. This process is accomplished by convolving the function f with low frequency and high frequency filter functions that arise from a choice of *mother wavelet* function Ψ, sometimes called a wavelet for short. The low frequency signal is often referred to as the *approximation coefficients*, and the high frequency is called the *detail coefficients*. The filtering process removes half the signals frequencies, so downsampling can then be performed without reconstruction loss on *f*. As a result, if the signal f consists of N data points, then the low and high frequency signals after the DWT contain approximately $\frac{N}{2}$ data points. See the appendix for more details.

Fixing a scale level $\ell \in \mathbb{N}$, the signal f is decomposed into its high and low frequency signals ℓ times. The discrete wavelet transform plays an important role in our work. The DWT is useful to our analysis for the following reasons:

(a) By filtering data down to different scale lengths, clustering the approximation coefficients allows us to directly compare how the scale of the data affects the clustering result. For our particular application, the approximation coefficients have an interpretation in terms of the analysis. Coarse-graining temporal signals captures seasonal, yearly, and eventually decadal trends, whereas coarse-graining spatial information captures city-, county-, and eventually state-sized features. Therefore, a comparison of clusterings from different coarse-grainings allows one to parse how these different scales affect clustering.
(b) The DWT runs fast. The filter bank implementation of the DWT runs in $\mathcal{O}(N)$ for each 1-D implementation [30], or $\mathcal{O}(N_1N_2N_3)$ for 3D tensors. Furthermore, we are only interested in the approximation coefficients, which have drastic size savings, as discussed above.
(c) The coarse coefficients at the end of the filter bank naturally creates a hierarchy of datasets that, when clustered, provide an ensemble for a consensus clustering algorithm.

2.2. Clustering algorithms and graph cuts

As discussed in the introduction, we are interested in analyzing how the scale of data affects clustering. In this subsection, we briefly discuss the unsupervised learning tools we use throughout the paper. We outline how they work, and discuss emphasize our choice for use.

2.2.1. K-means and spectral clustering

Clustering algorithms are diverse with varying advantages and disadvantages [10]. Arguably the most famous are partitioned based algorithms, where data are iteratively reassigned to clusters until an optimization function is minimized. The prototypical example of a partitioned based clustering algorithm is *K-means* discussed in the introduction. Given a natural number K, the K-means algorithm seeks to partition the data-set into K distinct groups that minimize the variance within the clusters. Lloyd's algorithm, the implementation of K-means we use in this work, is well known to run in $\mathcal{O}(NDKI)$, where N is the number of data points in dimension D, K is the cluster number, and I is the number of iterations.

Another popular method that builds on K-means is *spectral clustering*. Here, one leverages spectral graph theory to perform dimensionality reduction before applying the K-means algorithm. In spectral clustering,

an undirected weighted graph $\mathcal{G}$ is formed, where each vertex is a data point and the edge weight is a chosen affinity between vertices. Let $W = (w_{i,j})_{i,j=1}^{n}$ denote the weighted adjacency matrix for the graph $\mathcal{G}$. The *(unnormalized) graph Laplacian* L of W is a matrix that captures the combinatorial properties of the Laplacian on discrete data. The Laplacian L is a symmetric positive semi-definite matrix, so the eigenvalues may be ordered $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Finding the eigenvectors e_j corresponding to the lowest K eigenvalues, define $U = [e_1|e_2|\ldots|e_K]$ and cluster the rows using K-means. For more details, see [31].

A bottleneck of spectral clustering is its complexity. Given N data points, forming the adjacency matrix and computing eigenvectors carries a computational complexity of $\mathcal{O}(N^3)$. Given K-means' cost is $\mathcal{O}(NDKI)$, the total complexity of spectral clustering is $\mathcal{O}(N^3) + \mathcal{O}(NDKI)$. Though the computational complexity is roughly N^3, our application of spectral clustering will be on a smaller dataset.

Beyond the clustering challenges discussed in the introduction, K-means (and therefore spectral clustering) require the user to choose the number of clusters K in advance. In most applications, this parameter is not known, and additional heuristics are required to select this value. In spectral clustering, one can use the eigenvalues of the Laplacian L to determine the cluster number. Specifically, as the eigenvalues are ordered, search for a value of K such that the first $\lambda_1, \ldots \lambda_k$ are small, and λ_{k+1} is large. This method is justified by the fact that the spectral properties of L are closely related to the connected components of $\mathcal{G}$ [32]. Use of graph Laplacian eigenvalues to decide the cluster number K is called the *eigen-gap* heuristic.

2.2.2. Graph cut clustering

Given a notion of distance of data, the adjacency graph or matrix records the pairwise similarity. Clustering the data X into K clusters is equivalent to providing a K-cut of the adjacency graph $\mathcal{G}$. Graph cut strategies vary depending upon application. For example, the min cut algorithm minimizes the cost between components of the graph, but this can result in an undesirable clustering, e.g. a cluster with one element.

The *Ratio cut* is a graph cut that seeks to ameliorate this issue by incorporating the size of each component. Concretely, let $I \subset \{1, 2, \ldots, n\}$, I^c is the complement of I, and let $W(I) := \sum_{i \in I, j \in I^c} w_{i,j}$ be the total cost between the component I and its complement. Given disjoint subsets $I_1, I_2, \ldots I_k$ such that $\bigcup I_j = \{1, 2, \ldots, n\}$, its *ratio cut* is defined as

$$\mathrm{RC}(I_1, I_2, \ldots, I_k) := \frac{1}{2} \sum_{i=j}^{k} \frac{W(I_j)}{|I_j|}. \tag{2}$$

Finding $I_1, \ldots I_k$ such that equation (2) is minimized is NP-hard [33]. Luckily, the optimization problem in equation (2) can be written as a trace minimization problem involving a particular matrix arising from the graph cut. By relaxing this optimization problem to one where we do not fix the form of the matrix in the trace, one obtains an optimization problem whose solution can be obtained by spectral clustering. See [32] for further details.

2.3. Usage of K-means and ratio-cut in this work

In the subsequent implementation of CGC and MIER algorithms in section 4, we will require both a clustering algorithm for CGC and a graph cut algorithm for MIER. In our implementation of CGC, we will use K-means algorithm (executed with Llyods algorithm) for clustering, and in MIER we will use Ratio-Cut (executed by spectral clustering). We have made these choices because:

(a) K-means has historical precedence in clustering for climate applications [20, 21] and straightforward implementation. The scope of this work is not to find the 'best clustering' of our data; instead, we wish to understand how coarse-graining affects clustering and can be used to develop an ensemble of clusterings for use in understanding cluster method sensitivity to latent data scales.
(b) The MIER algorithm will require a graph cut of a particular adjacency matrix formed by a large ensemble of coarse-grain clusterings. As discussed in section 3.2, the adjacency graph is formed using normalized mutual information (see section 2.3 for Mutual Information). The ratio cut creates reasonable large components of this graph. Each component will be heterogenous, and distinct from one another on an information-theoretic level.

2.4. Mutual information

Mutual information provides a method to quantify the shared information. Here, we outline how the mutual information is computed. For a more detailed account of mutual information and clustering, see [34] and [35].

Let $X = \{x_i\}_{i=1}^{n}$ be a collection of data points. Suppose that we partition the data X into two clusterings $U = \{U_i\}_{i=1}^{k}$ and $V = \{V_i\}_{i=1}^{l}$. The *entropy* of the clustering U, denoted $\mathcal{H}(U)$ is the average amount of

information (e.g. in bits) needed to encode the cluster label for each data points of U. If the clustering V is known, U can be encoded with less bits of information. The *conditional entropy* $\mathcal{H}(U|V)$ denotes the average amount of information needed to encode U if V is known.

The *mutual information* $\mathcal{I}(U,V)$ measures how knowledge of one clustering reduces our uncertainty of the other. Formally,

$$\mathcal{I}(U,V) = \mathcal{H}(U) - \mathcal{H}(U|V).$$

Explicit formulas for $\mathcal{H}(U)$ and $\mathcal{H}(U|V)$ can be derived as follows. Let $n_{i,j} = |U_i \cap V_j|$ denote the number of points in both U_i and V_j. We set $a_i = |U_i| = \sum_{j=1}^{l} n_{i,j}$ to be the size of U_i, and $b_j = |V_j| = \sum_{i=1}^{k} n_{i,j}$ to be the size of V_j.

Assume points of X are sampled uniformly. Then the probability that a random point in $x \in X$ is in cluster U_i is $p(x) = \frac{a_i}{n}$. Moreover, the probability that points $x, y \in X$ satisfy $x \in U_i$ and $y \in V_j$ is $p(x,y) = \frac{n_{i,j}}{n}$. Therefore, it follows that

$$\mathcal{H}(U) = -\sum_{x \in U} p(x)\log(p(x)) = -\sum_{i=1}^{k} \frac{a_i}{n} \log\left(\frac{a_i}{n}\right),$$

$$\mathcal{H}(U|V) = -\sum_{x \in U, y \in V} p(x,y)\log\left(\frac{p(x,y)}{p(y)}\right) = -\sum_{i=1}^{k}\sum_{j=1}^{l} \frac{n_{i,j}}{n} \log\left(\frac{n_{i,j}/n}{b_j/n}\right),$$

which yields,

$$\mathcal{I}(U,V) = \sum_{i=1}^{k}\sum_{j=1}^{l} \frac{n_{i,j}}{n} \log\left(\frac{n_{i,j}/n}{a_i b_j/n^2}\right).$$

As the values of $n_{i,j}$, and therefore a_i, b_j are determined by the cluster values for the N^2 pairs of datapoints, the complexity to compute the mutual information is $\mathcal{O}(N^2)$. Notice that $\mathcal{I}(U,V) \geq 0$, and $\mathcal{I}(U,V) = \mathcal{I}(V,U)$. It then follows that

$$\mathcal{I}(U,V) \leq \min(\mathcal{H}(U), \mathcal{H}(V))) \leq \frac{1}{2}(\mathcal{H}(U) + \mathcal{H}(V)). \tag{3}$$

Equation (3) shows that we can normalize the mutual information by dividing by the average of the entropies [35]. Throughout, we define the *normalized mutual information* as

$$\mathcal{NI}(U,V) := \frac{2\mathcal{I}(U,V)}{\mathcal{H}(U) + \mathcal{H}(V)}.$$

We will use the (normalized) mutual information as a way to measure how similar two clusterings are. Since mutual information quantifies how much information from one clustering is obtained from another, it is ideal for measuring how much structure is lost by coarse-graining. As such, mutual information will play a key role in the MIER algorithm for ensemble selection. A $\mathcal{NI}(U,V)$ value of p can be interpreted as $100 \times p$ percent of information was 'lost' between the clusters U and V. We will be using this interpretation in section 4.

3. The CGC and MIER algorithms

Here, we present our wavelet-based workflow for analyzing scale dependence and ensemble assembly. We detail the clustering algorithm *Coarse-Grain Clustering* (CGC) and present a method for selecting clusters to include in an ensemble based off the mutual information, which we call *Mutual Information Ensemble Reduce* (MIER). Table 1 contains all the notation used throughout both algorithms for ease of reference.

3.1. Coarse-grain clustering (CGC)

This manuscript considers 4-way climate data tensors $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3 \times N_4}$. We will index the modes of the tensor using subscripts, namely

$$\mathcal{X} = (\mathcal{X}_{i_1,i_2,i_3,i_4})_{i_1,i_2,i_3,i_4=1}^{N_1,N_2,N_3,N_4}.$$

Each of the coordinates coordinates $i_1, \ldots, i_4$ describes a feature of the abstract dataset $\mathcal{X}$. Correspondingly, we will always make the following physical identifications: the first and second indices i_1 and i_2 refer to

Table 1. Notation for CGC and MIER algorithms.

Notation—CGC	Description
$\mathcal{X} = (\mathcal{X}_{i_1,i_2,i_3,i_4})_{i_1,i_2,i_3,i_4=1}^{N_1,N_2,N_3,N_4}$	Tensor of climate data
$\mathcal{X}_l, l = 1,\dots N_4$	Tensors via fixing i_4 index
$\Psi_j, j = 1,2,3$	Wavelet for the indices i_1, i_2, i_3
$\ell_j, j = 1,2,3$	Scale level of the DWT on index i_j
Notation—MIER	**Description**
$\mathcal{L} \subset \mathbb{N}^3$	Permissible set of wavelet resolutions (ℓ_1, ℓ_2, ℓ_3)
$\vec{\ell}$	A permissible point $(\ell_j)_{j=1}^3 \in \mathcal{L}$
$U^{\vec{\ell}}$	CGC at resolution $\vec{\ell}$
$\mathcal{G}$	$\mathcal{NI}$ graph for $\{U^{\vec{\ell}}\}_{\ell \in \mathcal{L}}$
W	Weighted adjacency matrix for $\mathcal{G}$
$\{\mathcal{L}_j\}_{j=1}^K$	Components of $\mathcal{L}$ corresponding to $K-$ cut of $\mathcal{G}$
$\mathcal{A}(U^{\vec{\ell}})$	Average $\mathcal{NI}$ between $U^{\vec{\ell}}$ and its component
$\{U^j\}_{j=1}^K$	Reduced ensemble - output of MIER

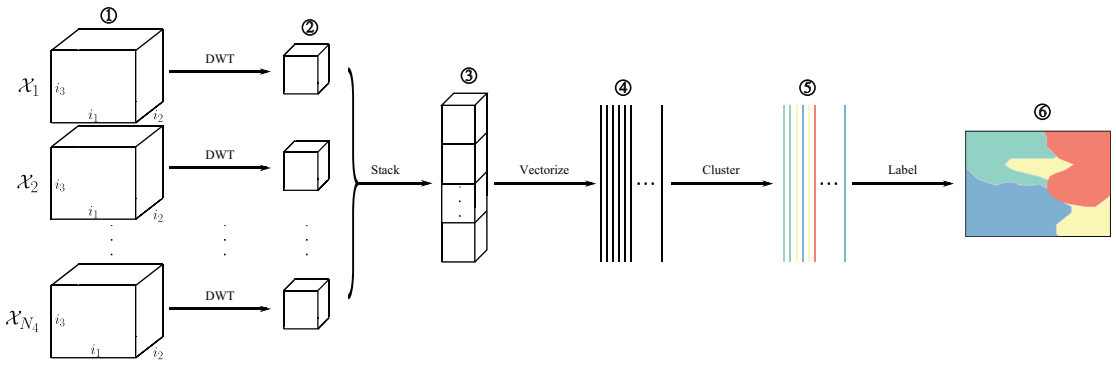


Figure 2. Diagram of each step of the CGC algorithm.

latitude and longitude coordinates, respectively; the index i_3 denotes time, and i_4 refers to state variables (e.g. temperature or precipitation).

The goal of this work is to provide meaningful clusterings for the spatial location, namely the coordinates corresponding to i_1 and i_2. Hence, we seek clusterings of the indices $(i_1, i_2) \in \{1,2,\dots,N_1\} \times \{1,2,\dots,N_2\}$ using the data $\mathcal{X}$. While our focus is on clustering two indices of 4-way tensors, we note that this method does generalize to clustering d-way tensors along any number of indices.

We now describe the Coarse-Grain Clustering algorithm. Figure 2 schematically displays the key features of CGC, while Algorithm 1 contains the pseudo-code.

Step One—Split Tensor: The first step in the coarse-grain clustering (CGC) algorithm is to separate the tensor $\mathcal{X}$ into sub-tensors that are largely statistically uncorrelated across the dataset. For example, temperature and precipitation are locally correlated—e.g. seasonal rainfall. However, they are weakly correlated at large spatial scales. Indeed, there are hot dry deserts, cold dry deserts, wet cold regions, and wet hot regions. Therefore in the climate dataset $\mathcal{X}$, one would separate by climate variables, but not by space or time. In a generic, non-climate specific tensor, one might split across different variables or runs of an experiment. We let $\mathcal{X}_1, \mathcal{X}_2, \dots, \mathcal{X}_{N_4}$ be the 3-way tensors obtained by fixing the i_4 index to the N_4 possible values. Note that each of these tensors $\mathcal{X}_l$ for $l = 1,\dots,N_4$ have the same size.

Step Two—DWT: After splitting the tensor $\mathcal{X}$ into sub-tensors, the next step is to select the inputs. The user chooses wavelets for each of the remaining indices i_1, i_2 and i_3. We let Ψ_j denote the wavelet for the index $i_j, j = 1,2,3$. Non-negative integers ℓ_j for $j = 1,2,3$ are selected to control the scale level of the DWT on index i_j. For each 3-way climate variable tensor $\mathcal{X}_l, l = 1,\dots N_4$, take the DWT transform.

Step Three—Stack: Since the same wavelets are used on each $\mathcal{X}_l$, the DWT of $\mathcal{X}_l$ will each have the same shape. These tensors can therefore be stacked along the face we wish to classify. For the climate biome problem, this would be the (i_1, i_2) face.

Step Four—Vectorize: Once the approximation coefficients are stacked, they may be vectorized along the face of interest. These vectors will be clustered according to a clustering algorithm of choice. This will result in a clustering of the face of interest on the DWT stack.

Algorithm 1: Coarse-Grain Cluster.

Input: $\mathcal{X} = (\mathcal{X}_{i_1,i_2,i_3,i_4})_{i_1,i_2,i_3,i_4=1}^{N_1,N_2,N_3,N_4}$, $\{\Psi_j, \ell_j\}_{j=1}^3$, $\mathcal{C}$
Result: Clustering of (i_1, i_2) face
1 form tensors $\mathcal{X}_l$, $l = 1, \ldots N_4$;
2 take DWT of each $\mathcal{X}_l$;
3 stack approximation coefficients along (i_1, i_2) face;
4 vectorize stack of DWT;
5 cluster vectors according to algorithm $\mathcal{C}$;
6 return labels to (i_1, i_2) face

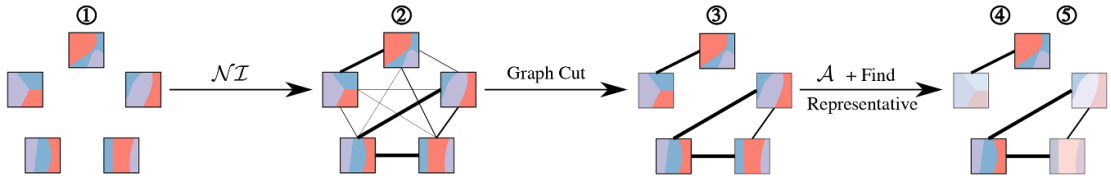


Figure 3. Diagram of each step of the MIER algorithm. In step (2), thicker lines correspond to larger mutual information. In step (5), the clustering with largest $\mathcal{A}(U^{\vec{\ell}})$ is highlighted.

Step Five—Clustering: The final input is the choice of clustering algorithm, as well as any hyper-parameters required for the chosen algorithm. For example, one may choose K-means, in which case the user needs to specify the number of clusters k. Let $\mathcal{C}$ denote the chosen clustering algorithm, along with its chosen hyper-parameters. With the inputs chosen, the algorithm proceeds as follows. Algorithm $\mathcal{C}$ is applied to the vectorized DWT coefficients from step four.

Step Six—Return Labels: The final step is to translate these labels on the coarse-grain stack to the face of the original data set. This is done using the inverse DWT. Specifically, cluster labels corresponding with the largest value appearing in the inverse DWT filter are used to propagate the coarse label to finer detail.

The complexity of CGC determined by the DWT and the clustering algorithm $\mathcal{C}$. The cost of the DWT (and its inverse) is $\mathcal{O}(N_1N_2N_3)$. The cost of $\mathcal{C}$ depends, of course, on the choice of clustering algorithm. For example, suppose we use K-means (as we will in our applications). Then since the size of the approximation coefficients is $\hat{N} := \frac{N_1}{2^{\ell_1}}\frac{N_2}{2^{\ell_2}}\frac{N_3}{2^{\ell_3}}$, the total cost of CGC is $\mathcal{O}(N_1N_2N_3) + \mathcal{O}(K\hat{N}I)$.

We remark that the process of computing coarse-grain clusterings is extremely parallelizable. Indeed, a directed tree structure can be implemented to semi-parallelize all scales of the DWT. Each of these scales can then be independently clustered. This fact, combined with the extreme size savings of the DWT, means the cost to compute many CGC's is no worse than computing the most expensive CGC in the collection.

3.2. Mutual information ensemble reduce (MIER)

The CGC algorithm describes how to produce a single clustering at a fixed coarse-graining. This coarse-graining arises from the choice of wavelets and wavelet levels $\{\Psi_j, \ell_j\}_{j=1}^3$. **The power behind CGC is its ability to produce many clusterings by simply varying the wavelet levels ℓ_j, $j = 1, 2, 3$**, which as discussed above, can be readily parallelized via a single instruction.

This process results in a large ensemble of clusters, one that is potentially too big to analyze. In this section, we discuss a method to select a small subset of this large ensemble of coarse-grained clusters. Our method leverages the mutual information to find a compact subset of clusters that contains most of the information across the large ensemble. This is accomplished by computing the mutual information between all the clusters in the large ensemble. This results in a connected graph. This connected graph is then ratio-cut to find heavily connected and therefore information theoretically similar clusters. For each component, we again use mutual information to select a single representative of the component. We call this method *Mutual Information Ensemble Reduce* (MIER).

The MIER algorithm is summarized in figure 3 and Algorithm 2. The details of the algorithm are as follows.

Step One—Large Ensemble: Let $\mathcal{L} \subset \mathbb{N}^3$ denote the permissible set of wavelet resolutions (ℓ_1, ℓ_2, ℓ_3) for the chosen wavelets $\{\Psi_j, \ell_j\}_{j=1}^3$. Reasonable values for $\mathcal{L}$ can be deduced from the dataset and problem of interest, e.g. scale of data and anticipated importance of embedded features. Once $\mathcal{L}$ has been decided, CGC is run for each $\vec{\ell} = (\ell_j)_{j=1}^3 \in \mathcal{L}$. We denote the clustering using the wavelet resolutions by $U^{\vec{\ell}}$. This results in an ensemble of clusters $\{U^{\vec{\ell}}\}_{\vec{\ell}\in\mathcal{L}}$.

Algorithm 2: Mutual Information Ensemble Reduce

Input: $\{U^{\vec{\ell}}\}_{\vec{\ell}\in\mathcal{L}}$
Result: $\{U^j\}_{j=1}^k$
1 import the large cluster ensemble $\{U^{\vec{\ell}}\}_{\vec{\ell}\in\mathcal{L}}$;
2 build mutual information graph $\mathcal{G}$;
3 ratio cut $\mathcal{G}$;
4 Compute $\mathcal{A}(U^{\vec{\ell}})$ for each $\vec{\ell}\in\mathcal{L}$;
5 return reduced ensemble $\{U^j\}_{j=1}^k$

Step Two—Mutual Information: Next, we compute the normalized mutual information between each clustering $U^{\vec{\ell}}$ in our ensemble $\mathcal{L}$. This results in a complete weighted graph $\mathcal{G}$ on nodes indexed by the set $\mathcal{L}$. The weight between node $\vec{\ell}=(\ell_1,\ell_2,\ell_3)$ and node $\vec{\ell'}=(\ell'_1,\ell'_2,\ell'_3)$ is the normalized mutual information $\mathcal{NI}(U^{\vec{\ell}},U^{\vec{\ell'}})$. We call the graph $\mathcal{G}$ the *mutual information graph*, and let W denote the weighted adjacency matrix for $\mathcal{G}$.

Step Three—Graph Cut: Having built the mutual information graph, we now perform a graph cut. Recall, spectral clustering solves a relaxed version of the ratio cut problem. Hence, we use spectral clustering on W to find a ratio-cut of $\mathcal{G}$. The eigen-gap heuristic is used when selecting the number of clusters K for spectral clustering W [32]. Let $\mathcal{L}_1,\mathcal{L}_2,\ldots\mathcal{L}_K$ denote the K components of $\mathcal{L}$ corresponding to the $K-$ cut of $\mathcal{G}$.

Step Four—Average $\mathcal{NI}$: For each component of the cut mutual information graph, we seek a best representative. Let $\mathcal{A}(U^{\vec{\ell}})$ denote the average mutual information between $U^{\vec{\ell}}$ and all other members of its component. That is, for $\vec{\ell}\in\mathcal{L}_j$,

$$\mathcal{A}(U^{\vec{\ell}})=\frac{1}{|\mathcal{L}_j|-1}\sum_{\vec{\ell'}\in\mathcal{L}_j,\vec{\ell'}\neq\vec{\ell}}\mathcal{NI}(U^{\vec{\ell}},U^{\vec{\ell'}})$$

where $\mathcal{NI}(U^{\vec{\ell}},U^{\vec{\ell'}})$ is the normalized mutual information between the clusters $U^{\vec{\ell}}$ and $U^{\vec{\ell'}}$).

Step Five—Choose Representative: For each $j=1,\ldots K$, the goal is to select the clustering $U^{\vec{\ell}}$ that best represents all the clusterings in $\mathcal{L}_j$. If $U^{\vec{\ell}}$ is a good representative for all the other clusterings within its component, then the mutual information between $U^{\vec{\ell}}$ and the other members of the component will be high on average. Thus, $\mathcal{A}(U^{\vec{\ell}})$ will be large. Consequently, we select a cluster in $\mathcal{L}_j$ for which $\mathcal{A}$ is maximized:

$$U^j\in\text{Argmax}_{\vec{\ell}\in\mathcal{L}_j}\mathcal{A}(U^{\vec{\ell}}).$$

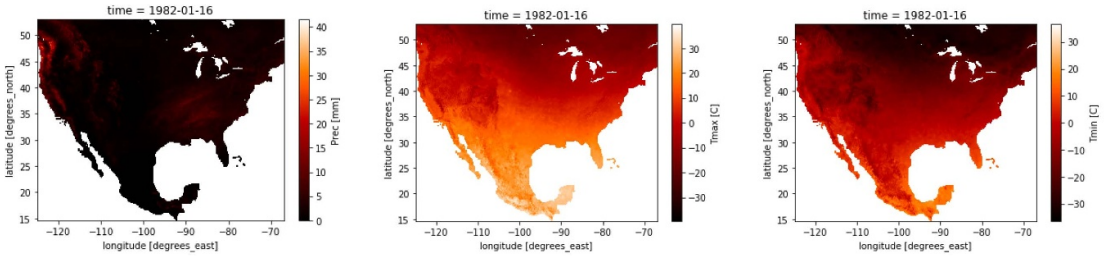
In the unlikely event that the Argmax consists of more than one element, one is selected randomly.

To compute the complexity of the MIER algorithm, we note there are $|\mathcal{L}|^2$ pairs of clusterings whose normalized mutual information needs to be computed. If there are N total data-points, the cost to form $\mathcal{G}$ is $\mathcal{O}(|\mathcal{L}|^2\,\mathrm{N}^2)$. Next, spectral clustering on the $|\mathcal{L}|^2$ connections of the graph needs to be performed. This costs $\mathcal{O}(|\mathcal{L}|^3)+\mathcal{O}(|\mathcal{L}|KI)$. This puts the total cost of MIER at $\mathcal{O}(|\mathcal{L}|^2\,\mathrm{N}^2)+\mathcal{O}(|\mathcal{L}|^3)+\mathcal{O}(|\mathcal{L}|KI)$.

This cost is deceptively high however. In general, $|\mathcal{L}|$ is very small compared to N. Indeed, $|\mathcal{L}|$ might be on the order of 10's or worst case 100's, while N is many orders of magnitude higher. The complexity of the MIER algorithm is therefore dominated by the cost normalized mutual information, and is really more like $\mathcal{O}(N^2)$.

4. Application—gridded climate dataset

As a proof of concept, we apply the MR-Cluster to a gridded historical climate data set of North America [36], referred to hereafter as L15. This data set ingests station data and interpolates results for each grid point, integrating the effects of topography on local weather patterns. The gridded data is six by six kilometers a side and consists of 614 latitudinal, 928 longitudinal, and 768 temporal steps for the years 1950–2013. The available monthly variables in the L15 data set are averaged values of daily total precipitation, daily maximum temperature, daily minimum temperature, and daily average wind speed. A representative snapshot of precipitation, maximum and minimum temperature is shown in figure 4. The datasets contains key inputs needed for biome classification using the KG model [17] and allows ready comparison against this expert judgement based approach. As this dataset is freely available, as well as widely



(a) Precipitation (mm) (b) Max Temp. ($^\circ C$) (c) Min Temp. ($^\circ C$)

Figure 4. Representative variables within the from L15 dataset, illustrating broad range of both coarse and fine spatial scales for precipitation, maximum temperature, and minimum temperature.

used within the climate community (e.g. Henn *et al* 2017), it provides a good benchmark application to illustrate capabilities of the method.

4.1. CGC hyperparameter selection for L15

The first step of CGC is to split the tensor $\mathcal{X}$ into sub-tensors corresponding to the climate variables. The historical precedent has been to use temperature and precipitation data to prescribe the biomes [17, 18, 20, 21]. Hence, we will only consider the sub-tensors $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3$ corresponding to precipitation and temperature values in the data set—namely the averaged values of daily total precipitation, daily maximum temperature, daily minimum temperature for each month. The next step is to determine the inputs to Algorithms 1 and 2. We describe these now.

L15 is a gridded observational dataset that achieves a six km spatial resolution, while each time slice of the data represents monthly timescale data. Whenever a wavelet transform is taken, the spatial and/or temporal scales are approximately doubled. For example, the L15 dataset has a six km spatial resolution. Thus, the coarse wavelet coefficients have a spatial resolution of 12, 24, 48, etc, km for one, two, and three wavelet transforms, respectively. Similarly, wavelet transforms of the monthly time scales will result in 2, 4, 8, etc, month long scales.

There is a scale at which both the spatial and temporal information is too coarse and begins to lose meaning. For example, on one extreme the spatial scale of the entire dataset is meaningless. On the other, the six km initial resolution is too fine scale for adequate characterizations into distinctly visible biomes at the North American scale.

These scales demarcate our set of permissible wavelet resolutions $\mathcal{L}$. At least one wavelet transform is taken in both space and time. The maximum for the spatial indices ℓ_1, ℓ_2 is four (roughly 96 km). The maximum number of temporal wavelet transforms is six (roughly 5 years). Further, we opted for a parsimony with regards to the spatial wavelet transforms—a wavelet transform is taken along i_1 (latitude) if and only if it is also taken along i_2 (longitude). For example, if we take two wavelet transforms in space laterally, we will also take two in space longitudinally so that horizontal spatial resolution is uniformly scaled. Thus, we have

$$\mathcal{L} = \{(i, i, j) : i = 1, \ldots, 4, j = 1, \ldots, 6\}. \tag{4}$$

Note, while it was possible to push the maximum levels to coarser grain, we wanted to avoid the risk of over-coarsening the result. For our choice of wavelets, we choose Daubechies 2 (db2) to match the time signals and Haar for space, corresponding to anticipated smooth periodicities in time and sharp gradients, e.g. near mountains, in space.

For the algorithm $\mathcal{C}$, we have chosen to use K-means clustering for $k = 4, 5, \ldots 20$ due to the historical precedence this algorithm has in clustering for climate applications [20, 21] and straightforward implementation. Recall that the aim is not to find the 'best clustering' of our data; instead, we wish to understand how coarse-graining affects clustering and can be used to develop an ensemble of clusterings for use in understanding cluster method sensitivity to latent data scales.

4.2. Results

4.2.1. L15 CGC algorithm results—effects of coarse-graining

We begin with a qualitative analysis by visualizing the effect scale has on clustering. For a parsimonious exposition, we have opted to display the sensitivity of CGC on L15 to scale only for a fixed value of $K = 10$. Different values of K result in similar qualitative results. Figure 5 explores this sensitivity.

Visually, scale can be seen to have a drastic effect on the overall structure of the clusters. For example, decreased temporal scale increases resolution from two to three eastern US classifications and shown in

figure 10(c) versus 10(d) and 10(f). Coarsened classifications are observed as a direct role of spatial scale, e.g. figure 10(f) versus 10(c) to 10(e). Cluster boundary shape is also affected by the wavelet resolution. For example, a vertical boundary can be found in the middle of the United States across each classification. However, the shape of that boundary depends on the resolution, e.g. figure 10(d) versus 10(e).

However, note that several coherent features are observed. Strong latitudinal dependence in the eastern portion of the US is consistent across clusterings as scales are modified, e.g. figure 5(a) to 5(d). Moreover, the location of the class boundary lines is relatively stable across all resolutions, in particular, in the Midwestern United States.

For a quantitative measure of the effects scale has on CGC for the L15 dataset, we use statistics derived from the normalized mutual information across various cluster numbers. Specifically, we compute the CGC clusterings $\{U^{\vec{\ell}}\}_{\ell\in\mathcal{L}}$ for each $K = 4,\ldots,20$. For each fixed K, we then computed the normalized mutual information between 'adjacent' scales:

$$\mathcal{NI}(U^{(1,1,1)},U^{(1,1,2)}),\mathcal{NI}(U^{(1,1,1)},U^{(2,2,1)}),\mathcal{NI}(U^{(1,1,2)},U^{(1,1,3)}),\ldots,\mathcal{NI}(U^{(3,3,6)},U^{(4,4,6)}).$$

By using adjacent resolutions, we are measuring the sensitivity of K-means to adjusting scale. Computing basic statistics across all K allows us to infer the expected information loss in coarse-grain clustering. Table 2 and figures 6–8 present this statistical data at different levels of granularity.

In table 2, we display $\mathcal{NI}$ pertaining to each possible adjacent coarse-graining. Since there are 38 adjacent coarse-grainings, each with 16 total $\mathcal{NI}$ values, a parsimonious representation is required. For table 2, we have chosen to only present the minimum, average, and maximum $\mathcal{NI}$ values.

Next, we begin grouping the data based on a fixed pair of adjacent spatial or temporal scale lengths. In figure 6, we have binned all the $\mathcal{NI}$ values for adjacent spatial resolutions. Once the data is binned, the minimum, average, and maximum are computed, as well as a histogram and a kernel density estimate. For example, figure 6(a) contains all the $\mathcal{NI}$ values between $(1,1,j)$ and $(2,2,j)$ for all $j = 1,\ldots,6$ and all $K = 4,\ldots,20$. The spread of these plots is the range of observed $\mathcal{NI}$. These are the worst and best case scenarios for information loss transitioning between the stated resolutions. The peaks correspond to values $\mathcal{NI}$ that were frequently observed. These are the approximate expected information loss transitioning between the stated resolutions.

In figure 7, we follow the same procedure only binning temporal scales instead of spatial scales. Finally, figure 8 takes the entire collection of $\mathcal{NI}$ values, plotting the PDF as well as the minimum, average and maximum.

4.2.2. L15 MIER algorithm results—creating the coarse-grain ensemble

For different K values, the CGC algorithm was run across all the resolutions $\mathcal{L}$ as in equation (4). For each fixed K, the MIER algorithm was applied to the outputs to discover the reduced ensemble. Once again for a parsimonious exposition, we will only display the MIER algorithm for $K = 10$.

Figure 10 shows the result of the MIER algorithm for $K = 10$. Figure 10(a) displays the value of $\mathcal{A}(U^{\vec{\ell}})$ for each resolution $\vec{\ell}\in\mathcal{L}$. The value on the vertical access denotes the number of spatial wavelet transforms, while the horizontal axis displays the number of temporal wavelet transforms. Figure 10(b) shows the results of the ratio cut algorithm. The key resolutions found by running the MIER algorithm are highlighted in a darker shade.

The clusters plotted in figure 10(c) through figure 10(f) are the best representative clusterings found by the MIER algorithm for $K = 10$. Each clustering encapsulates different observed qualitative features from the large ensemble of clusterings $\mathcal{L}$.

For each $K = 4,\ldots,20$, the MIER algorithm was run to find a small diverse subset of the resolutions $\mathcal{L}$. For a given K, MIER found either three or four clusters was adequate to represent the entire ensemble of clusters. Looking across all values of K, some resolutions were never discovered to be good representatives of the whole ensemble, while others appeared rather frequently. In figure 9, we plot the average number of times each resolution has appeared as representatives. As before, the value on the vertical access denotes the number of spatial wavelet transforms, while the horizontal axis displays the number of temporal wavelet transforms. We note that of the 24 resolutions, only five consistently appear, namely $\ell = (1,1,5),(2,2,1),(2,2,3),(2,2,4)$ and $(4,4,4)$.

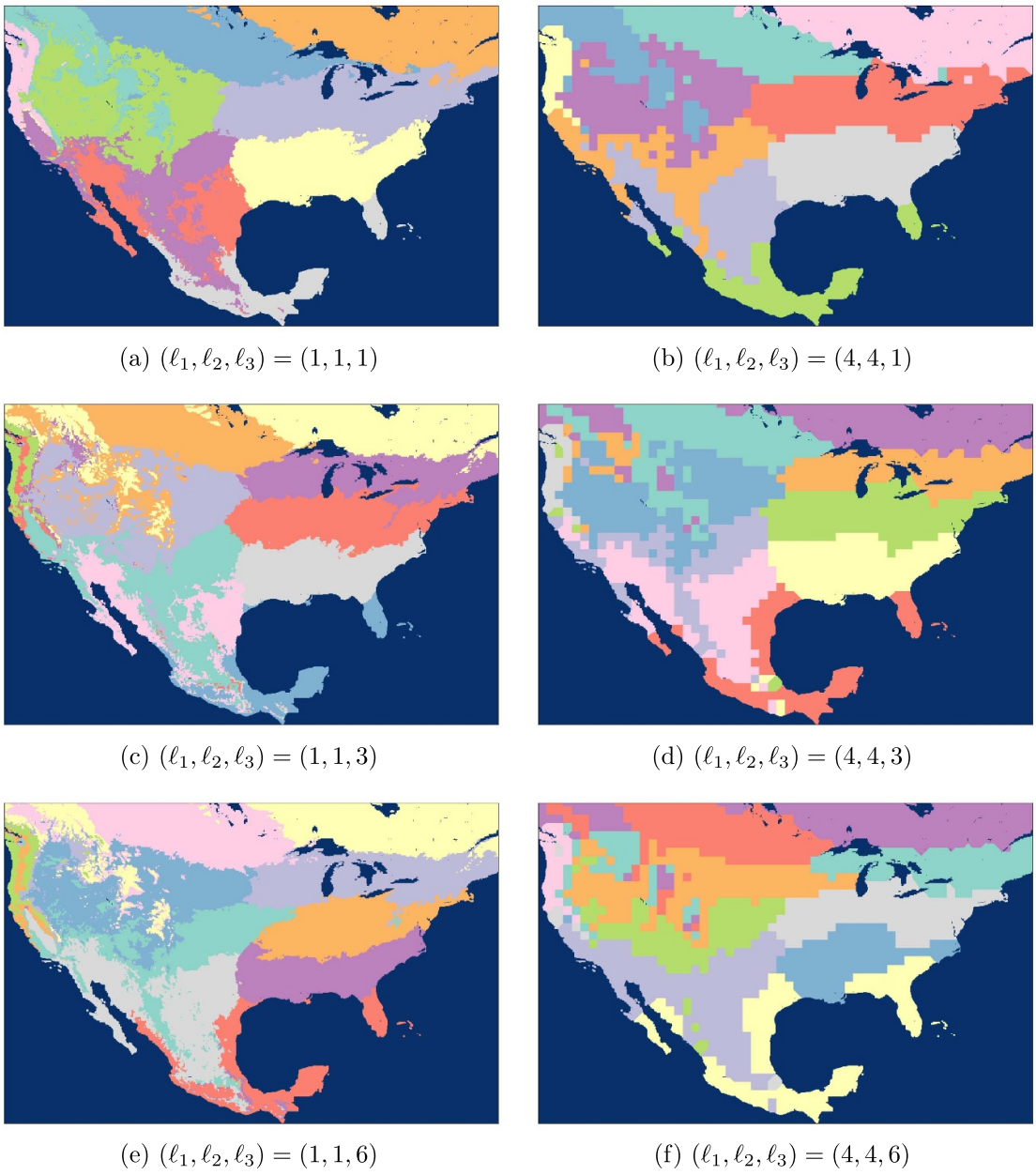
4.3. Discussion

4.3.1. CGC discussion

CGC resolution dependence plots in figure 5 highlight the variability that data resolution introduces into the clustering process. It perhaps comes as little surprise that increasing the number of spatial wavelet transform results in a coarser clustering. High variance regions, such as the Rocky Mountains, become less resolved as

Table 2. $\mathcal{NI}$ between adjacent clusterings in scale, averaged across all clusterings $K = 4, \ldots, 20$. Values **(i,i,j)** indicate the scale for CGC. The values min, average, max between adjacent scales are arising from the 16 normalized mutual information.

(1,1,1)	{0.83, 0.89, 0.91}	**(2,2,1)**	{0.77, 0.88, 0.94}	**(3,3,1)**	{0.74, 0.79, 0.83}	**(4,4,1)**
{0.77, 0.88, 0.94}		{0.77, 0.88, 0.94}		{0.78, 0.88, 0.95}		{0.72, 0.88, 0.97}
(1,1,2)	{0.83, 0.87, 0.89}	**(2,2,2)**	{0.78, 0.84, 0.87}	**(3,3,2)**	{0.66, 0.77, 0.83}	**(4,4,2)**
{0.7, 0.77, 0.87}		{0.73, 0.77, 0.86}		{0.71, 0.76, 0.87}		{0.64, 0.74, 0.87}
(1,1,3)	{0.73, 0.82, 0.87}	**(2,2,3)**	{0.76, 0.81, 0.85}	**(3,3,3)**	{0.64, 0.73, 0.77}	**(4,4,3)**
{0.77, 0.89, 0.95}		{0.84, 0.91, 0.95}		{0.78, 0.88, 0.94}		{0.77, 0.86, 0.96}
(1,1,4)	{0.73, 0.82, 0.87}	**(2,2,4)**	{0.76, 0.81, 0.86}	**(3,3,4)**	{0.64, 0.71, 0.76}	**(4,4,4)**
{0.81, 0.86, 0.92}		{0.76, 0.86, 0.91}		{0.79, 0.86, 0.91}		{0.76, 0.83, 0.92}
(1,1,5)	{0.81, 0.84, 0.88}	**(2,2,5)**	{0.72, 0.78, 0.84}	**(3,3,5)**	{0.63, 0.71, 0.78}	**(4,4,5)**
{0.78, 0.83, 0.9}		{0.77, 0.82, 0.87}		{0.76, 0.82, 0.87}		{0.74, 0.79, 0.85}
(1,1,6)	{0.78, 0.83, 0.89}	**(2,2,6)**	{0.75, 0.8, 0.85}	**(3,3,6)**	{0.67, 0.71, 0.79}	**(4,4,6)**

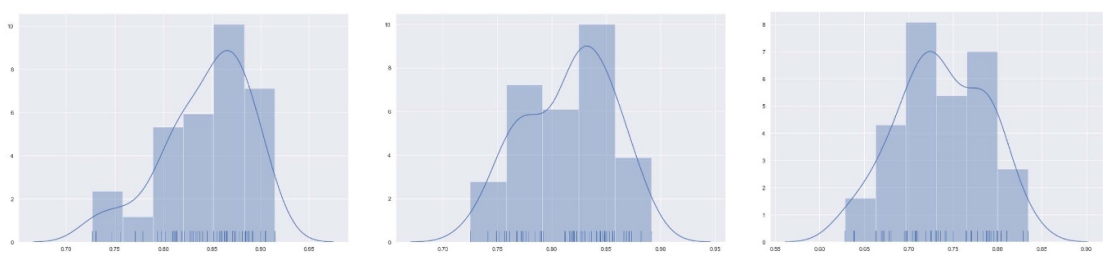


(a) $(\ell_1, \ell_2, \ell_3) = (1, 1, 1)$ (b) $(\ell_1, \ell_2, \ell_3) = (4, 4, 1)$

(c) $(\ell_1, \ell_2, \ell_3) = (1, 1, 3)$ (d) $(\ell_1, \ell_2, \ell_3) = (4, 4, 3)$

(e) $(\ell_1, \ell_2, \ell_3) = (1, 1, 6)$ (f) $(\ell_1, \ell_2, \ell_3) = (4, 4, 6)$

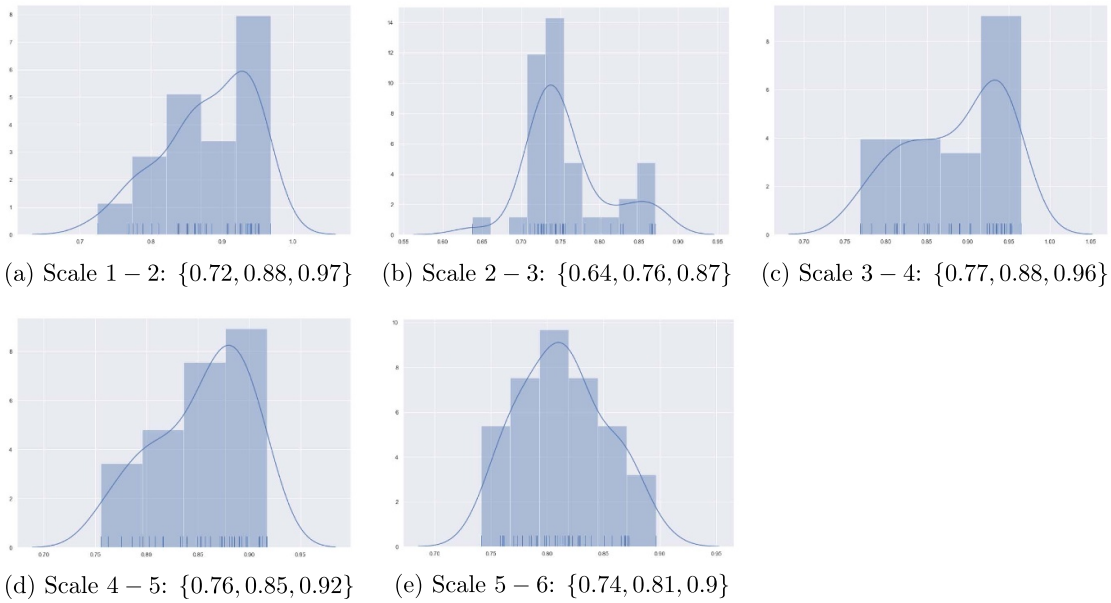
Figure 5. Clusterings obtained for $k = 10$ at resolution (ℓ_1, ℓ_2, ℓ_3).



(a) Scale $1-2$: $\{0.73, 0.84, 0.91\}$ (b) Scale $2-3$: $\{0.72, 0.82, 0.89\}$ (c) Scale $3-4$: $\{0.63, 0.74, 0.83\}$

Figure 6. $\mathcal{NI}$ values binned by spatial resolution. Scale $i - i+1$ indicates the levels of the spatial wavelet transform bin. The $\{\text{min}, \text{average}, \text{max}\}$ is displayed for each bin.

the number of spatial resolutions increases. However, large structural features such as The Great Plains are persistent across the spatial wavelet coarse-graining.



(a) Scale $1-2$: $\{0.72, 0.88, 0.97\}$ (b) Scale $2-3$: $\{0.64, 0.76, 0.87\}$ (c) Scale $3-4$: $\{0.77, 0.88, 0.96\}$

(d) Scale $4-5$: $\{0.76, 0.85, 0.92\}$ (e) Scale $5-6$: $\{0.74, 0.81, 0.9\}$

Figure 7. $\mathcal{NI}$ values binned by temporal resolution. Scale $j-j+1$ indicates the levels of the temporal wavelet transform bin. The $\{\text{min}, \text{average}, \text{max}\}$ is displayed for each bin.

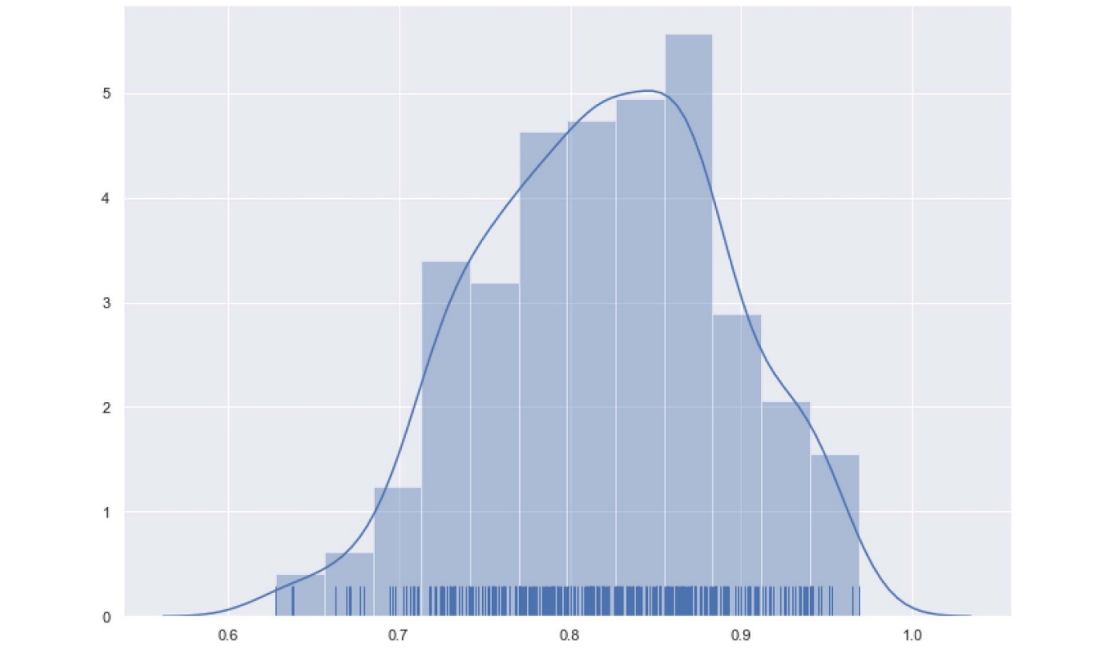


Figure 8. The PDF of adjacent $\mathcal{NI}$ values for CGC with K-means, $K = 4, \ldots, 20$ on L15 dataset $\{\text{min}, \text{average}, \text{max}\} = \{0.63, 0.82, 0.92\}$.

What is more unexpected is the effect that coarse-graining time has on the clustering. High variability regions remain high variability, however distinctly different clustering patterns do begin to emerge. For instance, how CGC clusters the Northern Rocky Mountains and the Pacific Northwest does seem to depend on the temporal resolution selected. Indeed, in figure 2 we see that increasing temporal scale results adds more 'micro' biomes to these regions. Low variability regions also depend heavily on the temporal scale. For example, the North Eastern U.S. splits into more biomes as the temporal scale becomes coarser.

To better understand which resolutions affect the clustering greatest, we computed the $\mathcal{NI}$ for various adjacent scales. In table 2, we see that there is a large variability in the average $\mathcal{NI}$. This shows that increasing the coarseness at some scale lengths results in a greater loss of information then others. For instance, $\mathcal{NI}(U^{(1,1,1)}, U^{(2,2,1)})$ is large - roughly 90% of the information is retained on average. This indicates that the difference between clustering at the $(1, 1, 1)$ resolution (12 kilometers) versus the $(2, 2, 1)$ resolution (24

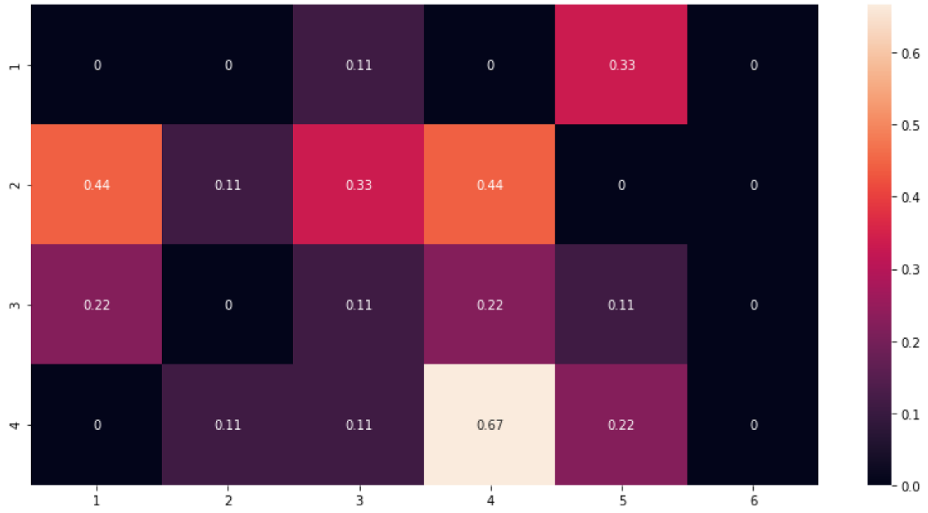


Figure 9. Average number of times a given resolution was chosen by the MIER algorithm across all $K = 4, \ldots, 20$.

kilometers) is not significant at the monthly time scale for the L15 dataset. At higher temporal scales however, it would appear as though the difference between the 12 kilometers and 24 kilometer resolution is more consequential. This is likely due to the blurring of the added complexity in regions such as the Northern Rocky Mountains and the Pacific Northwest discussed above.

Table 2 allows us to pinpoint individual resolution jumps where large amounts of information was lost. Figures 6 and 7 on the other hand, help discover the spatial and temporal scales that are responsible for information loss. Figures 6(a) (12–24 km) and 6(b) (24–48 km) demonstrate that regardless of temporal scale and K, the expected $\mathcal{NI}$ was approximately the same. However, 6(c) (48–96 km) is significantly lower, with a maximum value near the mean of the other two scale transitions. This indicates that spatial scale $i = 4$ has entered into a regime where the higher frequency features of the L15 dataset have been completely smoothed for more global ones.

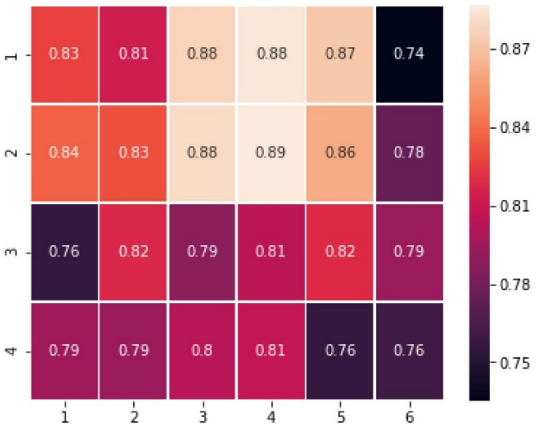
Perhaps more interesting, figure 7 allows us to identify the timescales responsible for the largest amount of change. Note that figures 7(a) (2–4 month), 7(c) (8–16 months) and 7(d) (16–32 months) have similar, right weighted distributions with relatively high expected $\mathcal{NI}$. However, figures 7(b) (4–8 months) and 7 (32–64 months) are different. The transition from 32 to 64 months is significantly more normally distributed with a noticeably lower average $\mathcal{NI}$. The transition from 4 to 8 months is very different, with its $\mathcal{NI}$ values densely centered near the mean significantly lower than the other four plots. What these plots seem to indicate is that, in terms of clustering the L15 dataset, is the critical role seasons and many years play. Indeed, the difference between 2 or 4 months is not very significant in terms of loss of information. However, once you start to jump across seasons (figure 7(b)), new patterns emerge causing a remarkably different clustering. Likewise, the difference between seasons and small number of years (figure 7(b) and (c) is not very significant. However, once you start to enter into a larger number of years (figure 7(e)), climate trends begin to emerge. This climate signal is slower moving compared to the yearly signal, resulting in a noticeably different clustering from an information theoretic standpoint.

4.3.2. MIER discussion

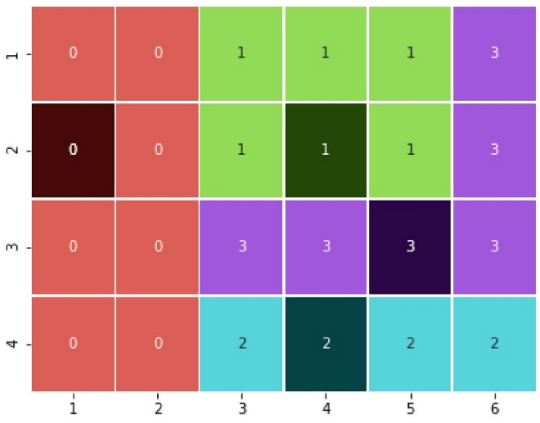
The MIER algorithm massively reduces the size of the large ensemble $\mathcal{L}$ for each choice of K. In all the experiments run, the size of $\mathcal{L}$ was 24, but the reduced ensemble size is between three and four. This illustrates the success of the method in identifying a reduced set of clusterings. Furthermore, the algorithm is successful at picking resolutions that are sufficiently spaced apart. Consequently, the chosen clusters accurately represent the dynamical range of all the 24 clusters in the large ensemble.

Qualitatively, this can be seen by comparing figures 5 and 10. The six sample plots in figure 5 are the extreme cases (lowest and highest coarse-graining) and as well as some middle cases. By looking at figure 10 we see that, for instance, the cluster $U^{(1,1,1)}$ belongs to component 0. The representative for component 0 is the cluster $U^{(2,2,1)}$. $U^{(1,1,1)}$ on figure 5 and $U^{(2,2,1)}$ on figure 10 are qualitatively similar. Indeed, $U^{(2,2,1)}$ has structure observed in $U^{(1,1,1)}$ and $U^{(4,4,1)}$, which is another clustering that belongs to the same connected component.

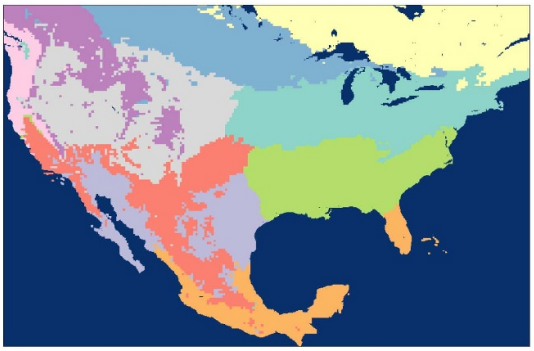
As can be seen from the output of the MIER algorithm, the reduced ensemble can succinctly represent differences across the spatial temporal resolutions. Most of the variance seen between the clusterings at



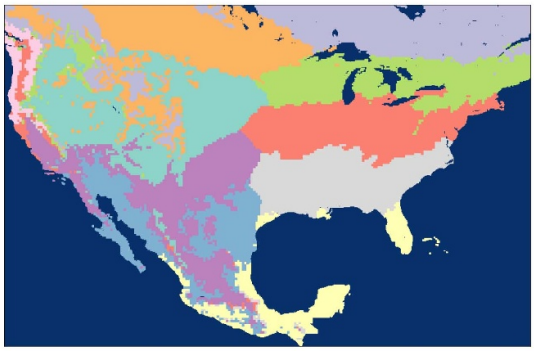
(a) $\mathcal{A}(U^{\vec{\ell}})$



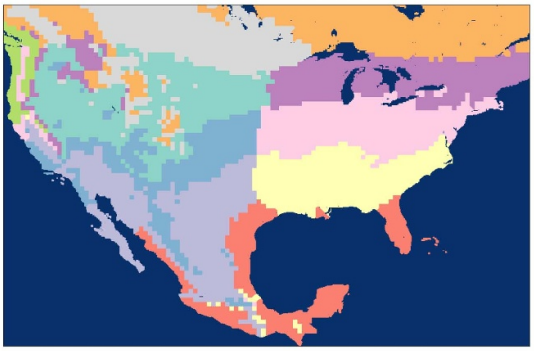
(b) Ratio-cut with optimal clusters highlighted



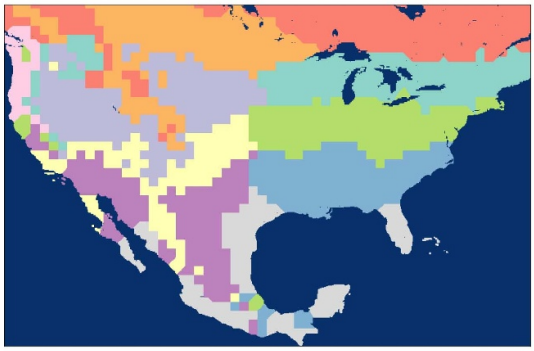
(c) $(\ell_1, \ell_2, \ell_3) = (2, 2, 1)$



(d) $(\ell_1, \ell_2, \ell_3) = (2, 2, 4)$



(e) $(\ell_1, \ell_2, \ell_3) = (3, 3, 5)$



(f) $(\ell_1, \ell_2, \ell_3) = (4, 4, 4)$

Figure 10. Output of the MIER algorithm for $K = 10$. a) $\mathcal{A}(U^{\vec{\ell}})$ for each resolution $\vec{\ell} \in \mathcal{L}$, b) results of the ratio cut algorithm and key resolutions, c–f) plots of key resolutions.

different resolutions is captured within this subset. From a numerical standpoint, the reduced ensemble is robust as well.

The success of the MIER algorithm can also be identified from an information theoretic standpoint. As can been seen from figure 10, the expected normalized mutual information between any representative and the other clusters in its component of the graph is usually rather large. Note that while mutual information graph $\mathcal{G}$ is built from all pairs of interactions $\mathcal{NI}(U^{\ell}, U^{\ell'})$, $\ell, \ell' \in \mathcal{L}$, the graph cut is made along adjacent resolutions. This was common amongst all the runs of the MIER algorithm for $K = 4, \ldots, 20$. Furthermore, the max value for $\mathcal{A}(U^{\vec{\ell}})$ over each component was almost always greater than the average value obtained by *adjacent* mutual information. This is noteworthy, since adjacent resolutions will always have a higher mutual information than those spread apart. This indicates that the MIER algorithm has selected components that were very information theoretically similar. Thus, picking the best cluster from each component via $\text{Argmax}_{\vec{\ell} \in \mathcal{L}_j} \mathcal{A}(U^{\vec{\ell}})$, we have found a small set of clusters that minimize the information loss across the large ensemble.

It is also worth noting that the most important resolutions found in figure 10 are in some sense, expected. Indeed, comparing figure 9 and figure 10, three of the four found resolutions are among the five most consistently discovered across all K. As discussed above, each run of MIER for $K = 4, \ldots, 20$ found either three or four representative clusters. In general, all but one of these would appear in the five consistent resolutions, with one falling to a different resolution. This indicates that for the L15 dataset, there are important resolutions for clustering independent of the cluster number K.

5. Conclusion

We have shown that scale of data is a non-negligible feature with regards to clustering. The normalized mutual information drop off at different scale lengths illustrates certain resolutions are producing novel clusters. Consequently, in addition to running several clustering algorithms, it is also important to include several coarse-grain clusterings into your cluster ensemble. To avoid ballooning the size of the ensemble, its crucial to not consider every possible coarse-graining, but rather a small subset that largely represents every possible resolution. The MIER algorithm has shown to be a good method to prune the size of the CGC ensemble. This capability to produce an ensemble of classifications representing the diversity of scales provides a direct pathway to better understand clustering sensitivities, illustrating a continued need to assess and mitigate uncertainties resultant from hyperparameter selection.

As discussed in section 3.1, the computational complexity to run many CGC's is generally no more expensive than naive K-means. However, the resultant large ensemble it is more difficult to analyze than a single clustering. By design, MIER algorithm effectively selects a diverse small subset from the large ensemble. However, as discussed in figure 1, the additional clusterings from the CGC and MIER framework should be imported into a consensus clustering algorithm. Further study is needed to assess the confidence across the cluster ensembles within this classification approach.

Acknowledgments

This research was supported as part of the Energy Exascale Earth System Model (E3SM) project, funded by the U S Department of Energy, Office of Science, Office of Biological and Environmental Research as well as LANL laboratory directed research and development (LDRD) grant 20190020DR and the Center for Non-Linear Studies. High-performance computing time was conducted at Los Alamos Nat. Lab. Institutional Computing, US DOE NNSA (DE-AC52-06NA25396).

Appendix A. Continuous wavelet transform

Representing functions via a decomposition into simpler functions is classic. Perhaps the most famous is the Fourier transform and Fourier series. Recall that given $f \in L^2[0, 2\pi]$, the set of square integrable (finite energy) functions and $\omega \in \mathbb{R}$, the Fourier transform

$$\hat{f}(\omega) = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-i\omega x} dx$$

measures the 'amount' of the frequency ω the signal f contains. The Fourier series of $f \in L^2[0, 2\pi]$ is a decomposition of f into a (potentially infinite) L^2-sum of trigonometric functions. Concretely, the functions $f_N(x) = \sum_{n=-N}^{N} \hat{f}(n) e^{inx}$ converge to f in $L^2[0, 2\pi]$ [37].

The Fourier transform only contains this frequency information, it does not know 'where' this frequency occurs. By taking a window of the domain, one can better localize frequencies. However, there is a Heisenberg's uncertainty type inequality that limits how well one can simultaneously resolve both the frequency and spatial values of a function [38]. The wavelet transform seeks to solve this issue by utilizing a scaling window, which is shifted along the signal. At each position, the spectrum is calculated. This process is repeated with varying window lengths, resulting in a collection of time–frequency representations of the signal and multiple resolutions i.e. power at different frequencies and scales.

Wavelet transforms differ from the Fourier transform in that there is a choice of wavelet function, or kernel, one uses to compute the weights. Rather than integrating against $e^{-i\omega x}$ to measure the amount of frequency ω the function f has, shifts and dilation's of a 'window' kernel function are used. The *mother*

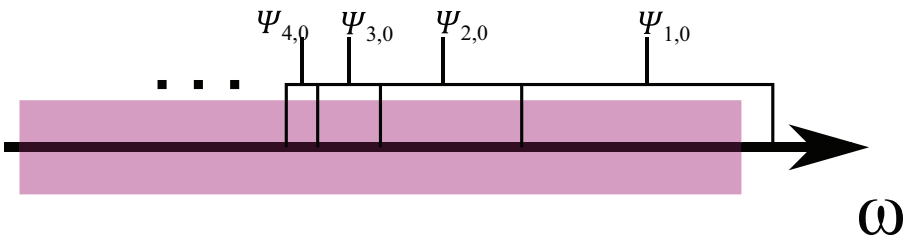


Figure C1. Diagram demonstrating how the different dilations cover the frequency spectrum of f. The bar in red indicates the frequency domain of f, and the boxes for each $\psi_{\ell,0}$ show what portion of the frequency spectrum the wavelet captures.

wavelet is a function $\Psi \in L^2([a,b])$ that satisfies additional regularity conditions [39]. Perhaps the most famous example is Haar wavelet, given by

$$\Psi(x) = \begin{cases} 1 & 0 \leq x < \frac{1}{2} \\ -1 & \frac{1}{2} \leq x < 1 \\ 0 & \text{else.} \end{cases}$$

Wavelets are generated by scaling and translated the mother wavelet. Given a scale factor s and translation factor τ, we denote the wavelet scaled by s and shifted by τ as

$$\Psi_{s,\tau}(x) = \frac{1}{\sqrt{s}}\Psi\left(\frac{x-\tau}{s}\right).$$

Just as the Fourier transform at ω measures the amount of frequency Ω in f by convolution with $e^{-i\omega x}$, the wavelet transform measure the 'amount' of the signal at scale s and translation τ by convolution. The (continuous) wavelet transform of f at s, τ is

$$\hat{f}(s,\tau) = \int_0^{2\pi} f(x)\overline{\Psi_{s,\tau}(x)}\, dx$$

where the bar denotes complex conjugate. In our work, we will only be working with real valued wavelet functions, so the conjugate is not necessary. Thus, the wavelet transform $\hat{f}(s,\tau)$ will always be real valued.

Appendix B. Discrete wavelets

By taking discrete steps in the frequency space, the Fourier transform was leveraged to create the Fourier series. Similarly, by fixing a scale and translation and taking integral steps, one can create a wavelet series representation of f. Usually, one fixes $s = 2$, and $\tau = 1$ (so-called dyadic sampling of scale/translation space), and modifies the wavelet transform to

$$\Psi_{\ell,k}(x) = \frac{1}{\sqrt{2^\ell}}\Psi\left(\frac{x-2^\ell}{2^\ell}\right),$$

where $\ell, k \in \mathbb{Z}$. The $\Psi_{\ell,k}$ are known as the discrete wavelets. If the discrete wavelets satisfy a very natural bounding condition called the *frame bounds*, then the initial signal can be reconstructed perfectly with (potentially infinite) L^2-sum of $\Psi_{\ell,k}$ [39]. That is,

$$f = \sum_{\ell=-\infty}^{\infty}\sum_{k=-\infty}^{\infty}\hat{f}(\ell,k)\Psi_{\ell,k}$$

where convergence is in L^2.

Appendix C. Discrete wavelet transform—wavelet filter bank

Clearly in practice, one cannot compute the infinite number of $\hat{f}(\ell,k)$. Luckily in practice, one would not have to anyways. At any given scale, the number of translations is bounded by the length of the signal interval. Therefore, it suffices to limit the range of scales one needs to compute. From Fourier analysis, we know that time compression by factor of 2 will stretch and shift the frequency spectrum by a factor of 2. If,

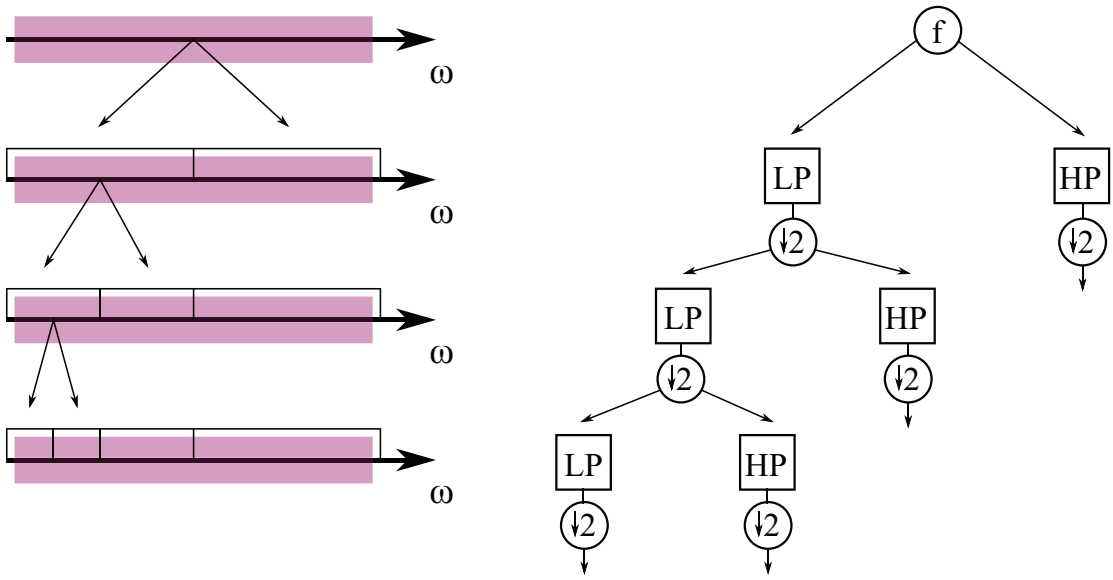


Figure C2. Diagram of a 3-layer wavelet filter bank for a signal f. The frequency domain of f is split each iteration via a low pass/high pass filter. After each filter, the resultant signal is down-sampled.

for example, $\Psi_{1,0}$ covers the upper bound of the frequency spectrum of f, then further dilations will begin to cover the whole frequency spectrum of f. See figure C1.

Because the window width for the frequency spectrum is halved at each iteration, it is not possible to cover zero with a finite number of iterations. For this reason, after a finite number of steps, the residual low passed signal is collected. The low frequency function is called the *scaling function*, denoted Φ. For the Haar wavelet,

$$\Phi(x) = \begin{cases} 1 & 0 \leq x < 1 \\ 0 & \text{else.} \end{cases}$$

This process of splitting the frequency domain via low-pass and band-pass filters is an example of a *filter bank* [40].

In practice, the filter bank is created by selecting a scale level ℓ, and iteratively computing the wavelet coefficients up to the chosen scale. For example, if $\ell = 2$, convolution with the wavelets $\Psi_{1,0}$, $\Psi_{1,1}$ will cover the top half of the frequency spectrum for f providing a high pass filter. Taking the scaling function from the first iteration, convolution with $\Psi_{2,0}, \Psi_{2,1}, \Psi_{2,2}, \Psi_{2,3}$ will cover the next quarter of the frequency spectrum for f. The residual is the low passed signal. See figure C2.

A computationally efficient implementation of this filter bank method is easily implemented for a discrete 1-D signal f of length N. Here, the convolution and downsampling can be encoded into a single vector-matrix multiplication $g = Hf$. The output $g = [g_l|g_h]$ will be a vector approximately the same length as f. Two vectors g_l and g_h that make up g consist of the low-pass and high-pass portions of f, respectively [41].

For multidimensional signals such as a tensor, there are several ways to compute the discrete wavelet transform. The most popular is the separable method, whereby the 1-D DWT is applied sequentially along each axis. For example, given a 3-way tensor data of size $N_1 \times N_2 \times N_3$, the 1-D DWT filterbank is first applied along axis 1. This splits the data into two chunks of size roughly $\frac{N_1}{2} \times N_2 \times N_3$. The 1-D DWT filterbank can then be applied again to the next axis, again reducing the size of the data [42]. See figure C3 for a diagrammatic explanation.

The complexity of the 1-D DWT can be shown to be $\mathcal{O}(N)$ [30]. The complexity for higher-order wavelet transforms will depend on the choice of wavelets, but a worst-case upper bound for a separable DWT of a third-order tensor is $\mathcal{O}(N_1 N_2 N_3)$. For a comprehensive overview of wavelets, see [43].

Appendix D. Example of discrete wavelet transform

Consider the 1-D signal $f = [1, 2, 3, 4, 5, 6, 7, 8]$. Suppose we want to compute the third-level Haar wavelet transform of f. The matrix for the Haar wavelet transform is given by

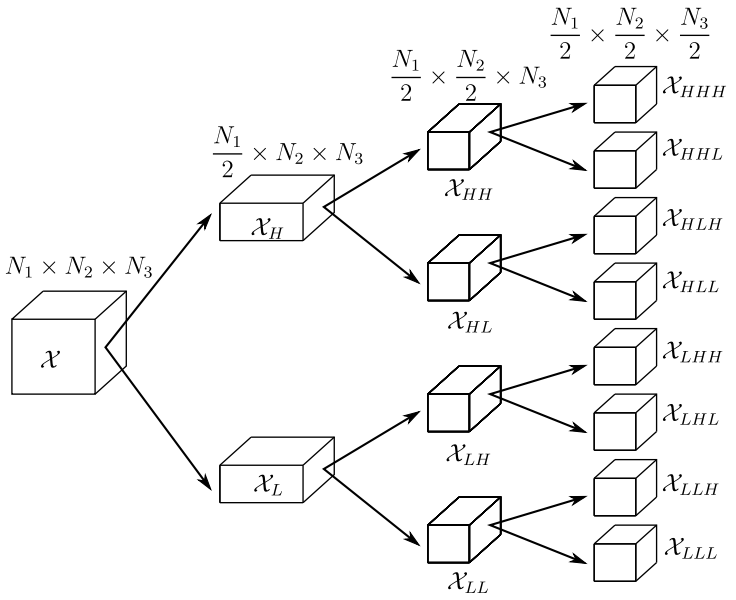


Figure C3. Diagram of (separable) DWT applied to a 3-tensor of data. After each application of a 1-D wavelet transform, the size of the tensor is reduced by a factor of two.

$$H = \frac{1}{\sqrt{8}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{pmatrix} \begin{matrix} \Phi \\ \Psi_1 \\ \Psi_{2,0} \\ \Psi_{2,1} \\ \Psi_{3,0} \\ \Psi_{3,1} \\ \Psi_{3,2} \\ \Psi_{3,3} \end{matrix}$$

Here, we have written the Φ and $\Psi_{\ell,k}$ next to the rows that the represent. In order to compute the DWT, we compute Hf. Doing the matrix multiplication yields

$$Hf = \frac{1}{\sqrt{8}} \left(\begin{array}{c} 36 \\ \hline -16 \\ \hline -2^{\frac{3}{2}} \\ -2^{\frac{3}{2}} \\ \hline -2 \\ -2 \\ -2 \\ -2 \end{array} \right)$$

where we have added separation lines to indicate the different detail coefficients from the different coarse-grainings. The top level is the approximation third level's approximation coefficient.

ORCID iDs

Derek DeSantis https://orcid.org/0000-0001-7648-7343
Phillip J Wolfram https://orcid.org/0000-0001-5971-4241
Katrina Bennett https://orcid.org/0000-0003-2433-8607
Boian Alexandrov https://orcid.org/0000-0001-8636-4603

References

[1] Cichocki A, Zdunek R, Phan A H and Amari S-ichi 2009 *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation* (New York: Wiley)
[2] De Bacco C, Power E A, Larremore D B and Moore C 2017 Community detection, link prediction and layer interdependence in multilayer networks *Phys. Rev.* E **95** 042317
[3] Kolda T G and Bader B W 2009 Tensor decompositions and applications *SIAM Rev.* **51** 455–500
[4] Alexandrov B S, Stanev V G, Vesselinov V V and Rasmussen K Ø 2019 Nonnegative tensor decomposition with custom clustering for microphase separation of block copolymers *Stat. Anal. Data Mining: The ASA Data Sci. J.* **12** 302–10
[5] Lopez C A, Vesselinov V V, Gnanakaran S and Alexandrov B S 2019 Unsupervised machine learning for analysis of phase separation in ternary lipid mixture *J. Chem. Theory Computat.* **15** 6343–57
[6] Schein A, Zhou M, Blei D M and Wallach H 2016 Bayesian Poisson Tucker decomposition for learning the structure of international relations arXiv:1606.01855
[7] Stanev V, Vesselinov V V, Gilad Kusne A, Antoszewski G, Takeuchi I and Alexandrov B S 2018 Unsupervised phase mapping of x-ray diffraction data by nonnegative matrix factorization integrated with custom clustering *npj Computat. Mater.* **4** 1–10
[8] Vesselinov V V, Mudunuru M K, Karra S, O'Malley D and Alexandrov B S 2019 Unsupervised machine learning based on non-negative tensor factorization for analyzing reactive-mixing *J. Comput. Phys.* **395** 85–104
[9] Lee D D and Sebastian Seung H 1999 Learning the parts of objects by non-negative matrix factorization *Nature* **401** 788–91
[10] Fahad A, Alshatri N, Tari Z, Alamri A, Khalil I, Zomaya A Y, Foufou S and Bouras A 2014 A survey of clustering algorithms for big data: Taxonomy and empirical analysis *IEEE Trans. Emerging Topics Comput.* **2** 267–79
[11] Cao X, Wei X, Han Y, Yang Y and Lin D 2013 Robust tensor clustering with non-greedy maximization *Twenty-Third Int. Conf. on Artificial Intelligence*
[12] Jegelka S, Sra S and Banerjee A 2009 Approximation algorithms for tensor clustering *Int. Conf. on Algorithmic Learning Theory* Springer pp 368–383
[13] Ding C, Xiaofeng H, Simon H D and Jin R 2008 *On the Equivalence of Nonnegative Matrix Factorization and k-Means-Spectral Clustering*
[14] Huang H, Ding C, Luo D and Li T 2008 Simultaneous tensor subspace selection and clustering: the equivalence of high order SVD and k-means clustering *Proc. of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* page 327–335
[15] Alexandrov B S and Vesselinov V V 2014 Blind source separation for groundwater pressure analysis based on nonnegative matrix factorization *Water Resour. Res.* **50** 7332–47
[16] Zhang Q, Berry M W, Lamb B T and Samuel T 2009 A parallel nonnegative tensor factorization algorithm for mining global climate data *Int. Conf. on Computational Science* Springer pp 405–415
[17] Kottek M, Grieser Jurgen, Beck C, Rudolf B and Rubel F 2006 World map of the Köppen-Geiger climate classification updated *Meteorologische Zeitschrift* **15** 259–63
[18] Thornthwaite C W *et al* 1948 An approach toward a rational classification of climate *Geographical Rev.* **38** 55–94
[19] Bishop C M 2006 *Pattern Recognition and Machine Learning* (Berlin: Springer Science+ Business Media)
[20] Zscheischler J, Mahecha M D and Harmeling S 2012 Climate classifications: the value of unsupervised clustering *Procedia Computer Sci.* **9** 897–906
[21] Netzel P and Stepinski T 2016 On using a clustering approach for global climate classification *J. Clim.* **29** 3387–401
[22] Mahajan M, Nimbhorkar P and Varadarajan K 2009 The planar k-means problem is np-hard *Int. Workshop on Algorithms and Computation* Springer pp 274–285
[23] Yu L and Zhou H H 2016 Statistical and computational guarantees of Lloyd's algorithm and its variants arXiv: 1612.02099
[24] Nguyen N and Caruana R 2007 Consensus clusterings *Seventh IEEE Int. Conf. on Data Mining (ICDM 2007)* IEEE pp 607–612
[25] Caruana R, Niculescu-Mizil A, Crew G and Ksikes A 2004 Ensemble selection from libraries of models *Proc. of the Twenty-First International Conference on Machine Learning* p 18
[26] Fern X Z and Lin W 2008 Cluster ensemble selection *Stat. Anal. Data Mining: The ASA Data Sci. J.* **1** 128–41
[27] Hadjitodorov S T, Kuncheva L I and Todorova L P 2006 Moderate diversity for better cluster ensembles *Information Fusion* **7** 264–75
[28] Kuncheva L I and Hadjitodorov S T 2004 Using diversity in cluster ensembles *2014 IEEE Int. Conf. on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)* vol 2 IEEE pp 1214–1219
[29] Amodei D, Olah C, Steinhardt J, Christiano P, Schulman J, and Mané D 2016 Concrete problems in AI safety arXiv: 1606.06565
[30] Shukla K K and Tiwari A K 2013 *Efficient Algorithms for Discrete Wavelet Transform: With Applications to Denoising and Fuzzy Inference Systems* (Berlin: Springer Science & Business Media)
[31] Ng A Y, Jordan M I and Weiss Y 2002 On spectral clustering: Analysis and an algorithm *Advances in Neural Information Processing Systems* pp 849–856
[32] Ulrike V L 2007 A tutorial on spectral clustering *Stat. Comput.* **17** 395–416
[33] Wagner D and Wagner F 1993 Between min cut and graph bisection *Int. Symp. on Mathematical Foundations of Computer Science* Springer pp 744–750
[34] Dom B E 2002 An information-theoretic external cluster-validity measure *Proc. of the Eighteenth Conference on Uncertainty in Artificial Intelligence* Morgan Kaufmann Publishers Inc pp 137–145
[35] Vinh N X, Epps J and Bailey J 2010 Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance *J. Machine Learning Res.* **11** 2837–54
[36] Livneh B, Bohn T J, Pierce D W, Munoz-Arriola F, Nijssen B, Vose R, Cayan D R and Brekke L 2015 A spatially comprehensive, hydrometeorological data set for Mexico, the US and Southern Canada 1950–2013 *Scientific Data* **2** 150042

[37] Rudin W 1962 *Fourier analysis on groups* vol 121967 (New York: Wiley Online Library)
[38] Folland G B and Sitaram A 1997 The uncertainty principle: a mathematical survey *J. Fourier Anal. Appl.* **3** 207–38
[39] Daubechies I 1992 *Ten Lectures on Wavelets* vol 61 (Philadelphia, PA: SIAM)
[40] Mallat S G 1989 A theory for multiresolution signal decomposition: the wavelet representation *IEEE Trans. Pattern Anal. Mach. Intell.* **11** 674–93
[41] Thyagarajan K S 2011 Discrete Wavelet Transform IEEE Piscataway, NJ
[42] Chun-Lin L 2010 *A Tutorial of the Wavelet Transform* (Taiwan: NTUEE)
[43] Jensen A and Anders la C-H 2001 *Ripples in Mathematics: the Discrete Wavelet Transform* (Berlin: Springer Science & Business Media)