

Understanding Atmospheric Chemistry using Graph-Theory, Visualisation and Machine Learning.

Dan Ellis

March 2020

*Veritatem inquirenti, semel in vita de omnibus,
quantum fieri potest, esse dubitandum:*

*In order to seek truth, it is necessary once in the course of our life, to
doubt, as far as possible, of all things.*

- Descartes, Rene, *Principles of Philosophy*

Contents

1 Computational Learning, Visualisation and Clustering:	1
1.1 Introduction	4
1.2 Species of the MCM and ways to represent them.	5
1.2.1 Input generation	5
1.2.2 Manual Categorisation	6
1.2.3 Tokenization	7
1.2.3.1 Species Names	8
1.2.3.2 SMILES strings	8
1.2.4 Graph Inspired	9
1.2.4.1 The species graph (fingerprint)	9
1.2.4.2 Node Embeddings (node2vec)	10
1.2.5 Molecular Fingerprints	11
1.2.5.1 Molecular Quantum Numbers (MQN)	12
1.2.5.2 Molecular ACCess System (MACCS)	12
1.3 Dimensionality Reduction Methods	12
1.3.1 Preparation of the data	13
1.3.2 Principle Component Analysis	14
1.3.2.1 Mathematical explanation of PCA	14
1.3.3 t-Distributed Stochastic Neighbor Embedding (t-SNE)	15
1.3.3.1 Mathematical explanation of t-SNE	16
Step 1	16
Perplexity	16
Step 2	17
1.3.4 PCA vs t-SNE, a quick comparison.	17
1.3.5 The Auto-Encoder (AE)	19
1.3.5.1 Demonstration of non-linear activation functions	20
1.3.6 Node2Vec	21
1.3.6.1 Sentence construction by sampling of a network	22
1.3.6.2 Word2Vec	23
1.3.7 Summary	23
1.4 Visualisation of clustering	23
1.4.1 Viewing the 2D species embeddings	24

1.4.2	Exposing overlapping data	24
1.4.3	Gooey Effect (Gaussian Blur)	24
1.4.4	Four Colours Theorem	24
1.5	Cluster Evaluation	25
1.5.1	Automated selection of clusters	25
1.5.1.1	Clustering (Silhouette) coefficient	27
1.5.2	Feature Extraction	27
1.5.2.1	Random Forrests	27
1.5.2.2	Calculating importance using Random Forrests	28
1.6	Results	29
1.6.1	CLuster distribution	29

Chapter 1

Computational Learning, Visualisation and Clustering:

Learning species structure using unsupervised machine learning.

“So, in the interests of survival, they trained themselves to be agreeing machines instead of thinking machines. All their minds had to do was to discover what other people were thinking, and then they thought that, too.”

- Kurt Vonnegut, *Breakfast of Champions*

1.1 Introduction

Historical significance

The established process of trial and error has always underpinned our survival [Noble, 1957]. Babies are born to rely on a set of sensory reflexes and a framework for physical reasoning [Baillargeon and Carey, 2012], and with these, we develop methods to navigate the influence of change within a physical, and auditory space [Lynch, 2011]. This method of decision making is reflected in our adult lives with ideas and actions being limited in choice by our intuition and experience [Descartes and Lafleur, 1960]. In science, we apply a methodological framework consisting of a continuous assessment of scepticism, educated guessing (hypothesizing) and rigorous practical testing. Specialists accrue years of practical and theoretical knowledge within a narrow field and can identify areas of potential gain and futility. Yet even with all prior knowledge, the discovery of new and untested techniques involve the tortuous traipsing through a sea of uncertainty. Such a methods sometimes prove fruitful, through accidental discoveries of items such as x-rays, penicillin, etc. [Roberts, 1989]; finding novel applications for existing methods such as optical tweezers for chemistry or the abstract field of maths utilised by Einstein [REF], but more often than not end in the constant evolution of a pre-existing project with no clear result.

Theory and Simulation in Science

Until recently much of the experimentation possible was limited by resources, levels of knowledge available technology. With the increase of computation power, we have been able to not only increase our understanding but also run theoretical simulations to guide exploratory efforts with an impact on real-world applications [Oliveira et al., 2006; T. Leube et al., 2018; Morozov, 2016; Yu-ChenLo, 2018]. However, as our ability to record and produce data increases, the need for the scientific method diminishes [Anderson, 2008]. Here the application of ‘big data’ tools and algorithms can provide insights and correlations much more compelling than the predictive capabilities of constantly changing models - “Since all models are wrong the scientist cannot obtain a "correct" one by excessive elaboration” - Box [1976]. As our level of attainable technology increases, so does the complexity of the data collected. Modern data-sets tend to be large, complex and highly multivariate. Although this greatly improves the quality of science that may be extracted from them, the difficulty lies in trying to represent it in such a way that we may successfully access the reliability of the results. Since simple bar and line graphs are no longer applicable, one solution falls within a class of unsupervised machine learning techniques called dimensionality reduction (DR).

Chapter Aims

In ?? we looked at visual representation as a way of understanding complex systems. ?? showed that the chemical properties could be visually inferred from the node-link graph structure of a mechanism. Similarly, ?? and ?? located the presence of important species and clusters of like properties by applying mathematical algorithms to the graph network. As opposed to attempting to visualise complex data, this chapter looks at learning the structure of a chemical species and simplifying it into two dimensions. Here it is possible to extract key features of like-groups through the use of vector clustering, which unlike the graph clustering in ?? works by determining the density between points on a plane.

The chapter begins with the introduction of the chemical system, and the various methods for representing species structure within it (section 1.2). Next, we define the dimensionality reduction methods which shall be used to simplify the aforementioned inputs (??). This is followed by a brief overview of the visualisation methodology (??). Finally, all three sections are combined to produce a set of result and conclusions about the use of DR to identify species structure.

1.2 Species of the MCM and ways to represent them.

The master chemical system (as defined in all previous chapters), represents our foremost knowledge of gas-phase chemistry within the troposphere. It has been shown that due to its creation protocol (??), much of the information about a species structure is encoded within the reaction pathways it can take. This section explores the different methods of representing a species structure, intending to provide a machine built algorithm with the greatest amount of information about each species and its functionality. To do this a range of input types will be evaluated against several dimensionality reduction algorithms to isolate which chemical properties are most ‘picked up’.

1.2.1 Input generation

The MCM provides species information in the form of a species ‘smiles’ (subsubsection 1.2.3.2) and the IUPAC InChi string [Heller et al., 2013]. Within this chapter, we use only the smiles string, which is either manually processed using regular expressions or with the aid of pythons RDKIT package [Landrum et al., 2019]. There are seven different methods for representing the chemistry, each of which are outlined below.

1.2.2 Manual Categorisation

Reactions within the MCM are determined by a set of rules (PROTOCOL SECTION). These are designed to mimic the process a chemist may discover new species and often rely on the bond availability and functionalisation of a species. Since the present functional groups are the benchmark of whether a DR algorithm has successfully separated species structure, it makes sense to run a unit test using the known functional groups of a species as the input.

To generate the functional groups the regular expressions in Table 1.1 are used¹ on the smiles strings (described in subsubsection 1.2.3.2) for each species. In extracting the functional groups we can plot the likeliness a species with a certain group is likely to have another using a chord diagram - Figure 1.1. Since most species are found to contain a multitude of functional groups, the separation of these into ‘tidy’ clustered groups seems unlikely.

PAN	C\\((=O\\)OON\\((=O\\)=O\$ ^\\{0-1}\\]\\N\\+{0,1}\\]\\((=O\\)OOC O=N\\((=O\\)OOC\\((=O\\) C\\((=O\\)O\\]N\\+{0,1}\\]\\((=O\\)\\]O-{0,1}\\]
Carb. Acid	[~O](C\\((=O\\)O\$ ^OC\\((=O\\))
Ester	[\\^O](C\\((=O\\)O\\b OC\\((=O\\))C
Ether	(\\([\\^O=]+\\))*C\\((\\([\\^O=]+\\))*O\\((\\([\\^O=]+\\))*C\\((\\([\\^O=]+\\))*
Per. Acid	c\\((=O\\)O\$ ^OO\\((=O\\)C
Nitrate	O(NO2\\b NOO\\b N\\((=O\\)=O\\]N\\+\\](?:\\[O-\\])\\((=O\\)){2})
Aldehyde	C=O\$ ^O=C
Ketone	C\\((=O\\)C
Alcohol	CO\\b (?=^\\b)(?!^\\b)CO. (?=^\\b)(?!^\\b)OC. \\((O\\) C\\)O(\\b [~O]
Criegee	\\[O-\\]\\[O\\+\\]
Alkoxy rad	\\[[\\/]\\{0,1}CH\\{0,1}\\] \\b[\\^O]\\[O\\.\\{0,1}\\]
Peroxyacyl rad	\\w\\((=O\\)O\\[O\\.\\{0,1}\\]

Table 1.1: CHECKKKKKKK!!!!!! A set of regular expressions that may be used to determine the number of occurrences of a functional group within a SMILES string.

¹To see the structure of each functional group type, go to ??.

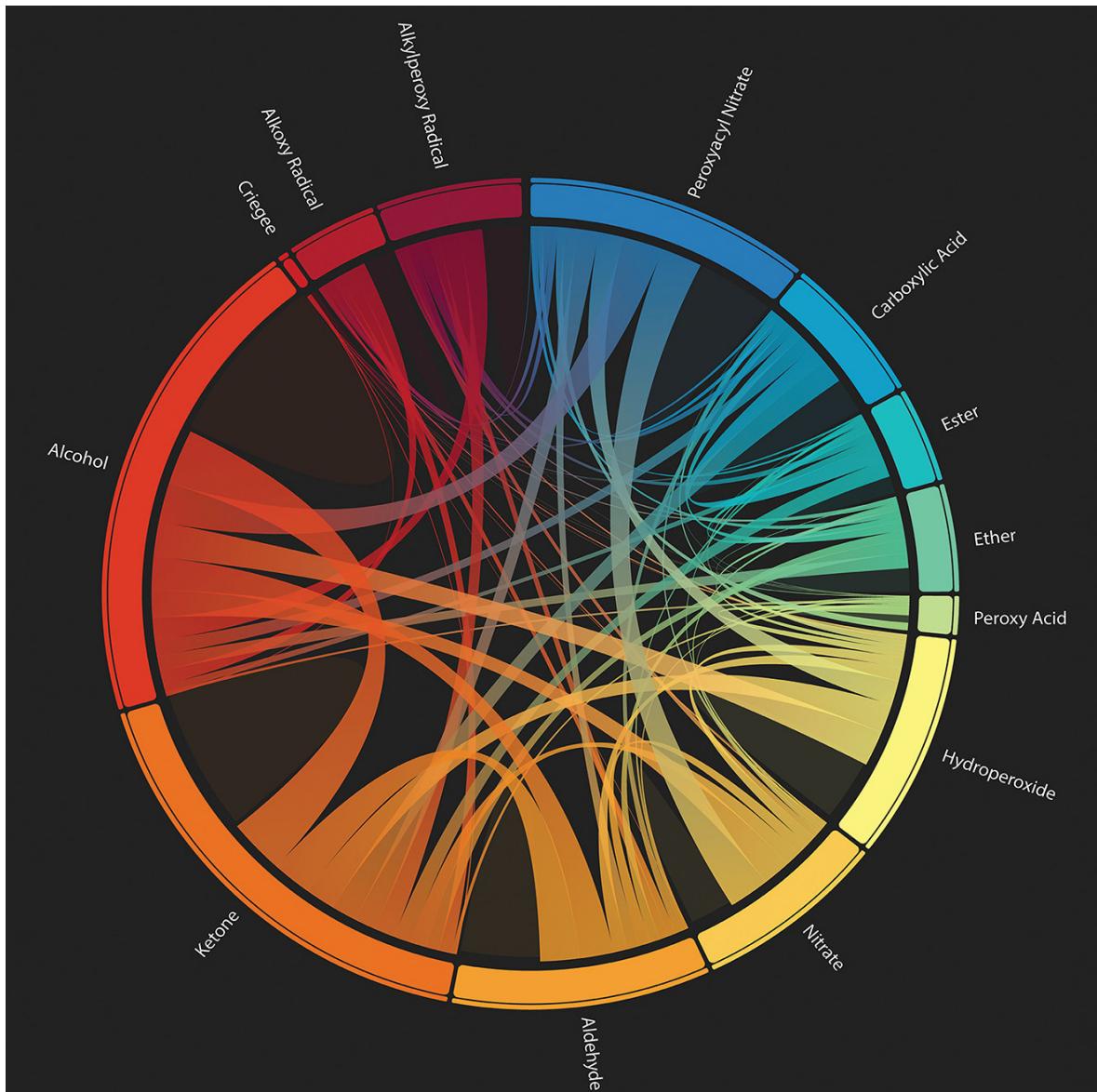


Figure 1.1: **The multifunctionality of the MCM.** A chord diagram showing the functionalisation of a species within the MCM. Arc sizes represent what percentage of all functional groups in the MCM mechanism a group contains. Translucent areas of no outwards links represent species with multiples of a certain functional group, of which Alcohols and Ketones have the most. Source: [Ellis, 2019]

1.2.3 Tokenization

As computer algorithms are unable to understand words or their meaning, we have to first categorise the data into groups. Tokenisation is the conversion of a string into characters and representing them with a numerical equivalent. In doing so a string of characters can be converted into a numerical vector, allowing for its representation in a latent vector space. Within our input selection, we have two sets of inputs we can convert. These are the species names, and their smiles string representation.

1.2.3.1 Species Names

In ?? it was shown that the dedicated species names for species in the CRI mechanism were often representative of their structural properties. This adage also applies for the MCM, where an intuitive naming convention has been selected. This is often derived as part of the construction protocol, where a species names reflect its own, or its precursor's structure (which it will have at least in-part inherited).

Although this is not the most robust method of defining the structure, it allows for an easy test of the algorithms, for which the user can quickly compare the human-readable output.

1.2.3.2 SMILES strings

Smiles ('Simplified Molecular-Input Line-Entry System') provide a human-readable representation of the molecular structure, [Weininger, 1988]. They provide a linear human-readable representation of the chemical structure within a molecule. This makes it easy for us to visually check the structure of a species without any additional work. Besides, their role in generating the molecular fingerprints in subsection 1.2.5 makes it a useful comparison to make when evaluating methods of structure representation.

Construction Methodology of SMILES strings

Smiles strings are constructed in three parts. We begin with a species backbone, then add break cycles and branches producing a smiles string. A visual description of this procedure is given below.

1. The smiles string is built by creating the longest possible chain to form a molecule backbone.

Figure 1.2b

2. This may within itself contain aromatic rings denoted by the lowercase carbons and a number corresponding to the location of each break cycle. Figure 1.2c

3. Finally all the functional groups and branches attached to the main backbone are added. These

are nested within the parenthesis to show that they are not part of the skeletal backbone.

Figure 1.2d

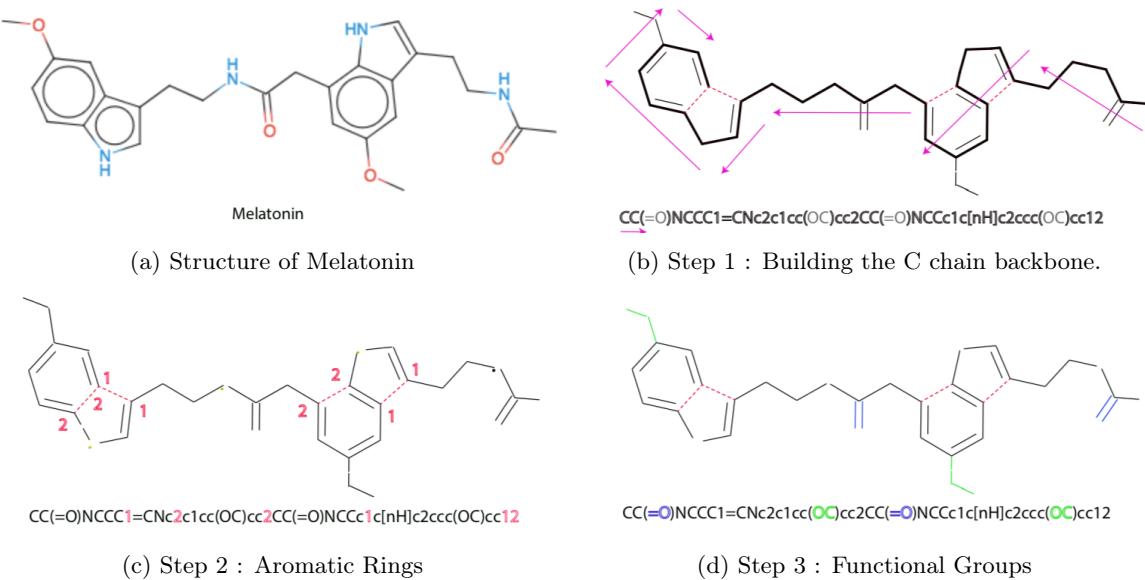


Figure 1.2: **Construction process of a smiles string.** The example compound is Melatonin. Although this does not exist within the atmosphere, it provides a clear example of the smiles string methodology. Figure 1.2a is made using smiles drawer: [Probst and Reymond, 2018]

1.2.4 Graph Inspired

?? - ?? have shown the role of graphs in revealing network properties and structure. Graphs in themselves can simplify relational data into two/three dimensions for visualisation and algorithmic clustering. Continuing this trend we can represent a species structure in the form of a graph (subsubsection 1.2.4.1), as well as converting the structure of a mechanism for dimensionality reduction (subsubsection 1.2.4.2)

1.2.4.1 The species graph (fingerprint)

The structure of a species has long represented using a graph-like layout, ???. It, therefore, follows that other methods for representing the graph structure would also apply. One such method is the use of an adjacency (or relational) matrix to describe the relationships between atoms and bonds in a species. Such a methodology is already used in the construction of bond and z-matrixes, [Aumont et al., 2005; Parsons et al., 2005].

The construction of a structure matrix/graph begins with a chemical species. Here the relationships between atoms (Figure 1.3b) is converted into an adjacency matrix (Figure 1.3c). However since species have different numbers of each atom, a template allowing us to compare different graphs is required. To do this a maximum occurrence table (Figure 1.3a) is created. Here, for example, BCARY C₁₅H₂₄, a sesquiterpene contains the most carbon atoms of any species within the MCM. This universal matrix is now able to contain any possible combination of atoms in a species.

As machine learning algorithms only vectors as an input, it is possible to decompose the 37^2 element adjacency matrix into rows, which can then be joined together, Using this method we create a flat array (vector) of 259 elements (518 bytes) which can be used to represent our species.

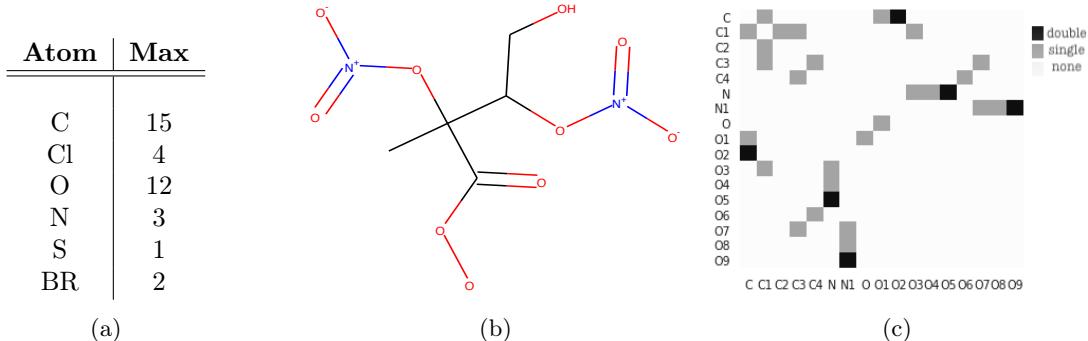


Figure 1.3: **Constructing a graph from species structure.** (a) shows the maximum number of times an atom occurs for any single species in the MCM. (b) depicts the graph-like chemical structure of $\text{INB}_1\text{NBCO}_3$. This is a highly processed species stemming from Isoprene, and this makes for a good example of the bond matrix. Finally, a matrix representing the bonds in $\text{INB}_1\text{NBCO}_3$ is created from the maximum possible occurrence matrix from (a). For simplicity, empty row/column pairs have been removed to produce (c). This matrix will always be symmetrical as the bonds do not have a direction.

1.2.4.2 Node Embeddings (node2vec)

?? and ?? showed that the underlying structure of a chemistry mechanism graph contains information about the species and reactions within it. This is seen in Figure 1.4, where colour represents the ratio of potential oxidation of a species. Here as emitted species become progressively more processed, the number of bonds which may be oxidised diminishes (lighter colours near the centre) until they eventually form carbon dioxide and water.

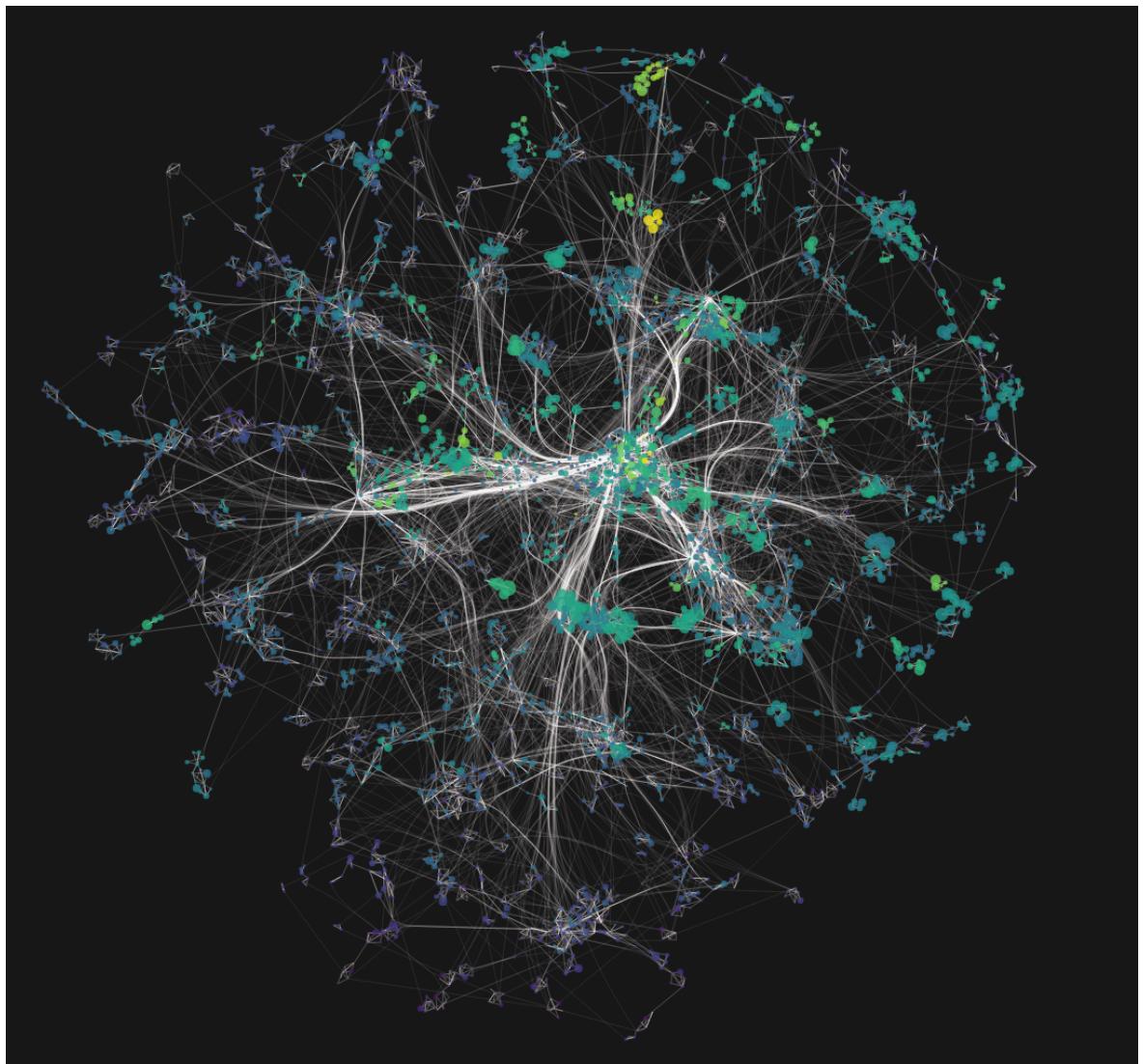


Figure 1.4: **The graph of an MCM subset representing the chemistry within Beijing.** Here colours show the increase of O–C ratio as species are oxidised (lighter). All emitted species ultimately tend towards carbon monoxide which is at the centre of the graph.

This type of structural information can be extracted through the use of a natural language processing package capable of transforming a graph into a vector - node2vec [Grover and Leskovec, 2019]. Since this may also be used for dimensionality reduction, it shall be described in the next section (subsection 1.3.6).

1.2.5 Molecular Fingerprints

In the field of chemical informatics, molecular fingerprints (or structural keys) are used to encode and query structural properties of species. Their binary representation makes them suitable for dimensionality reduction and the exploration of chemical space (a type of property space constructed using pre-determined properties and boundary conditions).

Here species properties are often split into structural and psycho-chemical groups - which has used such as the discovery of natural analogues (which circumvent problems such as intolerances in medicine [Spahn et al., 2017]). Although there exist many different types of molecular fingerprints, the two main ones that will be explored are molecular quantum numbers (MQN) and the molecular access system (MACCS).

1.2.5.1 Molecular Quantum Numbers (MQN)

In chemistry the shape, phase and electron occupancy of an atom may be described through the use of four quantum numbers: the n principle quantum number, I angular momentum quantum number, M_i magnetic quantum number and M_s spin quantum number. The rationalisation of elements based on their structure, and by consequence reactivity, has led to the most iconic tool of the modern-day chemist - the periodic table, where increasing atomic numbers follow the principal quantum number, [Wang and Schwarz, 2009]. In representing a molecule as a set of 42 quantum numbers, MQN fingerprints produce a multi-dimensional mapping of atom, bond, polarity and topology count [Nguyen et al., 2009].

1.2.5.2 Molecular ACCess System (MACCS)

MACCS keys are a 164^2 bit structural keys formulated through answering a series of structure-related questions. Developed by MDL Information Systems [, MDL], their main purpose lies in being a SMILES Arbitrary Target Specification (SMARTS) system for substructure searching. However, their distinct structure key format makes them highly suitable for similarity detection. In many cases, the optimised version of MACCS keys is cited ([Durant et al., 2002]), although most use cases exploit a variation of the undocumented 166bit keys. We use the implementation presented by [Landrum et al., 2019; rdkit, 2019] for all molecular fingerprints in this section.

1.3 Dimensionality Reduction Methods

In the last section, we described several methods in which the chemical structure of a species could be encoded for direct comparison. However since each input is made up of a multitude of elements, it is still not a simple task to determine the differences and similarity between all species in mechanisms. Dimensionality reduction is the process of reducing the number of random variables and only presentind a set of principal values, by mapping a high-dimensional space into a low-dimensional one,

²They are 166-bit keys, although there is no real agreement to what the 44th keys' purpose is, and therefore it is often omitted. Within RDKIT this is denoted by a ? [rdkit, 2019].

[Roweis and Saul, 2000]. This allows us to flatten a multivariate input into the two dimension required for a simple scatter plot.

In this section, we begin by explaining the data preparation required for dimensionality reduction (??) before describing the different possible methods of reducing the dimensions of a dataset.

1.3.1 Preparation of the data

Real-world data is rarely preformatted in such a way that it can be used directly within a computational model. Often values need to be cleaned and corrected to be fit for purpose. In the interest of completeness, the two main methods of data adjustment for machine learning are outlined below. These are normalisation and standardisation.

Normalisation

If the data is without (dimensionless) or of a single unit, it is possible to rescale the data between a range - most commonly 0,1. In doing so it is possible to interpret the importance of value in contrast to the largest recorded value. This gives us a percentage scale spanning the range of the data. Such a range is useful in the definition of colourmaps and describing the importance of value relative to the dataset. To rescale a dataset we shift the minimum value to zero, then divide by the new maximum of the dataset (Note this is equivalent to the range of the unshifted dataset.)

$$n(x_i) = \frac{x_i - \min_x}{\max_x - \min_x} \quad (1.1)$$

Standardisation

If the components we wish to compare are of different units or are expressed with a different scale, normalising them would not produce meaningful data. Instead, it is possible to standardise the data by looking at each points deviation from the mean. This is done by dividing the variation of each point from the mean by the standard deviation of S and results in a value between $\{-1,1\}$, Equation 1.2. In statistics this is known as the ‘z-score’³

$$z(x_i) = \frac{x_i - \mu_x}{S} \quad (1.2)$$

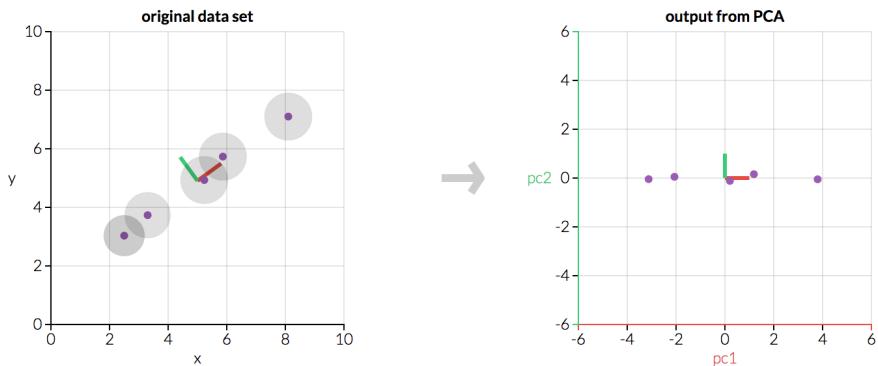
³Possibly because of the American spelling of standardization?

1.3.2 Principle Component Analysis

One of the most well-known dimensionality reduction methods is the determination of the principal components through the use of Principal Component Analysis (PCA). This increases the readability of a dataset by creating a set of new uncorrelated variables which maximise the variance, [Jolliffe and Cadima, 2016].

PCA works on the assumption that components within a dataset are linear combinations of each other. By simplifying these linear combinations, it is possible to identify the components which explain the most variability in a dataset - these are the principal components

A simpler interpretation of this would be to adjust the direction of each axis of the data, such that its projection has the largest variability. In doing so it is possible to determine which components contribute the most to changes in the dataset, [F.R.S., 1901; Hotelling, 1933]. An example of this is seen in Figure 1.5 where the second component of the original data may be removed with little effect on the overall result of the data. Such methods have applications in compression and signal filtering, [Hernandez and Mendez, 2018; Hamadache and Lee, 2017].



PCA is useful for eliminating dimensions. Below, we've plotted the data along a pair of lines: one composed of the x-values and another of the y-values.

If we're going to only see the data along one dimension, though, it might be better to make that dimension the principal component with most variation. We don't lose much by dropping PC2 since it contributes the least to the variation in the data set.



Figure 1.5: **Determining the Principal Component of a sample dataset.** It can be seen that in a change in axis to follow the first principal component (right), it is possible to explain most of the variation in the sample dataset (left). Source: [Powell, 2020]

1.3.2.1 Mathematical explanation of PCA

Note: The basic statistics/mathematics required to understand this section is shown in ???. Please read this if you are not familiar with any of the terms below.

The mathematics behind PCA consists of first calculating the covariance matrix. This is a $n \times n$ matrix

outlining how strongly each variable changes with every other. Using this we can calculate both the eigenvalues and eigenvectors of the matrix ⁴. This may be done using a computational package such as numpy or scipy [Oliphant, 2006; Jones et al., 01].

We can now sort the eigenvector columns by influence using their eigenvalues. This way a feature data-set can be produced by removing vectors of low influence. The final feature dataset can now be transposed and multiplied by the transpose of the original dataset. This produces an output dataset containing each principal component of the desired dimension.

1.3.3 t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is an algorithm designed with visualisation in mind [Maaten and Hinton, 2008]. Rather than representing the data through a series of linear transformations, t-SNE uses local relationships to create a low-dimensional mapping, much in the same way as a fully connected force graph, Figure 1.6. This allows the ability to capture non-linear structures in the data which cannot be done through linear mapping methods (e.g. PCA).

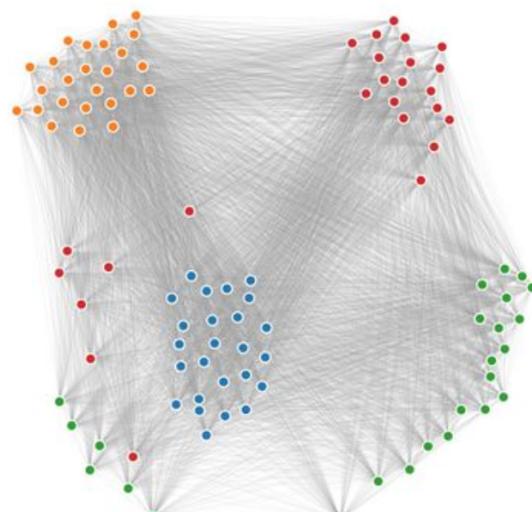


Figure 1.6: **Representing the t-SNE algorithm as a fully connected force graph.** Here each node is attached to every other node. Nodes with a strong relationship are pulled closer together than those with a weaker one.

The algorithm itself can be broken down into two parts,

1. Create a probability distribution which dictates relationships between neighbouring points
2. Recreate a lower-dimensional space following the probability distribution established in 1.

⁴These need to be unit vectors, although most packages already do this out of the box.

and is described in subsubsection 1.3.3.1. The main reason t-SNE produces good results is that it can handle the ‘**crowding problem**’ very well. The crowding problem is a product of the ‘curse of dimensionality’. This is because, in high dimensional space, the surface of a sphere will grow much quicker than one in a lower dimension space. This means higher dimension spaces will have more points at a medium distance from a certain point, Figure 1.7. When we map our data into a lower dimension, data will try to gather at its medium distance, resulting in a more ‘squashed’, and thus crowded, output.

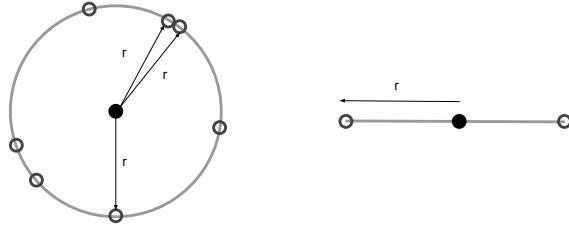


Figure 1.7: An example of how the curse of dimensionality affects the mapping of points a certain distance from each other.

1.3.3.1 Mathematical explanation of t-SNE

In the original paper [Maaten and Hinton, 2008], the algorithm is described using the etymologic dissection of its name.

Step 1

First we begin with Stochastic Neighbour Embedding (SNE) - the distribution across neighbouring datapoints in our high dimension space. This is done by converting the high dimensional Euclidian distances between points into conditional probabilities representing their similarity:

$$p_{ij} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq l} \exp(-\|x_k - x_l\|^2 / 2\sigma_i^2)} \quad (1.3)$$

Here $p_{i|j}$ is the conditional probability that x_i may pick x_j as a neighbour. This is proportional to the probability density of a Gaussian σ_i centered at x_i .

Perplexity Since we want the number of neighbours of each point to be similar in number and prevent a single point from having a disproportionate influence on the entire system we introduce a hyperparameter named *perplexity*. This works by ensuring that σ_i is small for points in densely populated areas and large for spare ones. This means that the perplexity of a system can be thought of as a scale of the number of neighbours considered for any one point. Generally, values between 5

and 50 are considered to give good results, with larger perplexities taking global features into account, and by consequence smaller ones, local features.

Step 2

Now a probability distribution describing the relationship between points has been formulated, we wish to express this as a low dimensional mapping of our inputs X in terms of our output dimensions Y . Naturally, we would want to make the low dimensional mapping represent a similar (Gaussian) distribution as in Step 1. However, it often causes issues presented by the ‘overcrowding problem’ subsection 1.3.3. This is because the gaussian has a ‘short tail’, and thus nearby points are likely to be pushed together. A solution to this is the student t-distribution which has a longer tail⁵:

$$q_{i|j} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{-1}} \quad (1.4)$$

Note: The definition and explanation of the Student t-distribution is given in ??.

The optimisation of this equation can be achieved through the use of *gradient decent*⁶ on the Kullback-Leibler divergence ?? between distributions p and q . Here the gradient is used to apply an attractive and repulsive force on the items⁷.

1.3.4 PCA vs t-SNE, a quick comparison.

Of the two algorithms PCA has been around for much longer, and its uses are well established within the scientific community. It is fast, simple and easy to use and very intuitive. The PCA algorithm works by creating a lower-dimensional embedding which best preserves the overall variance of the dataset. Clusters created from the algorithm are grouped in ways, such that they preserve the greatest variance of the data.

The main drawback of PCA is that it is a linear projection. If our data happened to be in a ‘swiss roll’ (spiral) pattern, we would not be able to ‘unroll’ it. This is because PCA works by viewing the data from different perspectives, much like casting a shadow from various directions. With such an example, there is no one way we can do this that unfurls the spiral.

⁵The distribution employed is a t-distribution with only one degree of freedom. This is identical to the Cauchy distribution

⁶**Gradient Decent** This is an optimisation algorithm used to minimise a function by iteratively moving in the direction of the steepest descent. This can be used to find local minima and is defined by the negative of the gradient. Its primary usage in machine learning is the updating of parameters (coefficients in linear regression and weight in neural networks).

⁷A positive gradient signifies attraction, whilst a negative one corresponds to repulsion.

t-SNE, on the other hand, is a relatively new method, [Maaten and Hinton, 2008]. Its greatest asset is that it is not limited by linear projections. Although more computationally intensive for large datasets, t-SNE produces visibly cleaner results. Unlike in PCA, t-SNE cannot be trained on additional data at a later point, however, the output clusters are more visually distinct (they have less overlap). Much like in a force graph, the output from t-SNE is scale-invariant. This means that whilst the location of clusters in a PCA reduced representation have an attributable quality, those produced by t-SNE will not necessarily contain the same information.

To test the differences between the two methods, a box model run representative of the chemistry within Beijing was used. The aim was to classify the diurnal profiles of each species concentration (much like the cosine similarity in ??). Diurnal profiles were extracted on the third day of a spun up model initialised with initial conditions representative of the chemistry within the Beijing environment (??). These are then standardized and converted into temporal vectors for use in the algorithms.

Figure 1.8 shows the output of both dimensionality reduction algorithms on the dataset. Different colours represent the location of clusters of similar diurnal profiles. A greater dispersion between clusters and species overlap is seen within the PCA output, Figure 1.8a. This makes it harder to distinguish species from each other or other groups around them. Since distance between clusters within t-SNE does not hold the same mathematical meaning as PCA, the algorithm is able to provide a better distribution of points, creating clearer and more defined clusters, Figure 1.8b. The concentration profile shapes for each coloured cluster are shown in Figure 1.8c.

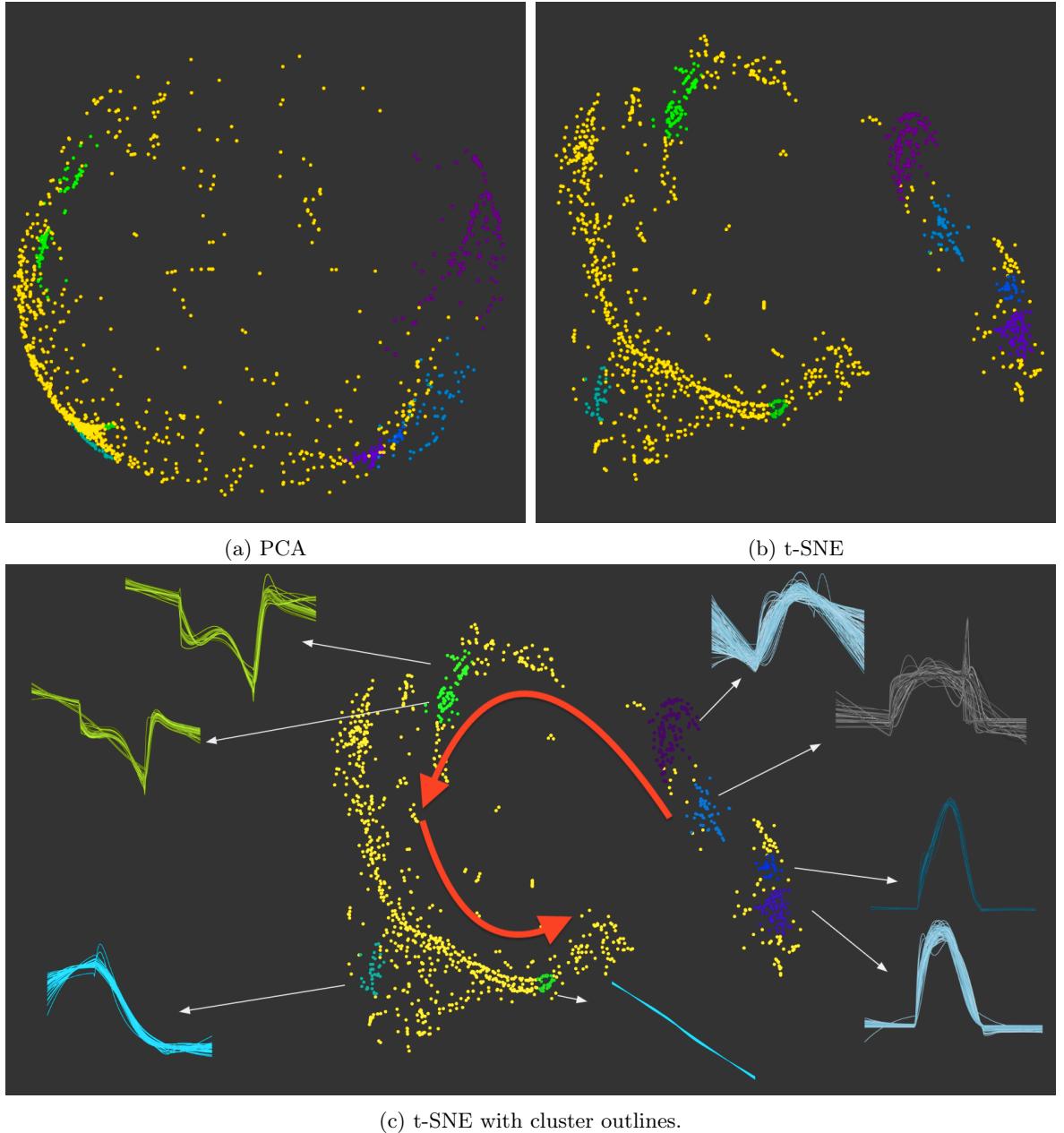


Figure 1.8: Showing the difference between PCA and t-SNE clustering. These figures show the clustering of a set of standardized concentration profiles (c) across two styles of dimensionality reduction: PCA (a) and t-SNE (b).

1.3.5 The Auto-Encoder (AE)

Auto-encoders are a subclass of neural networks which are primarily used for dimensionality reduction. Rather than predicting a numerical output, AutoEncoders focus on the construction and deconstruction of data through the use of an encoder and decoder pair. The encoder takes an n-dimensional input and applies a compression, reducing it to the number of dimensions in the bottleneck layer. This reduced dataset is then reconstructed within the decoder. Such a process not only allows for an easy understanding of the error of the reduced data but can also be used in the filtration of noisy or

pixelated data [Leite et al., 2018; Dataman, 2019] and as an input to more complex machine learning models.

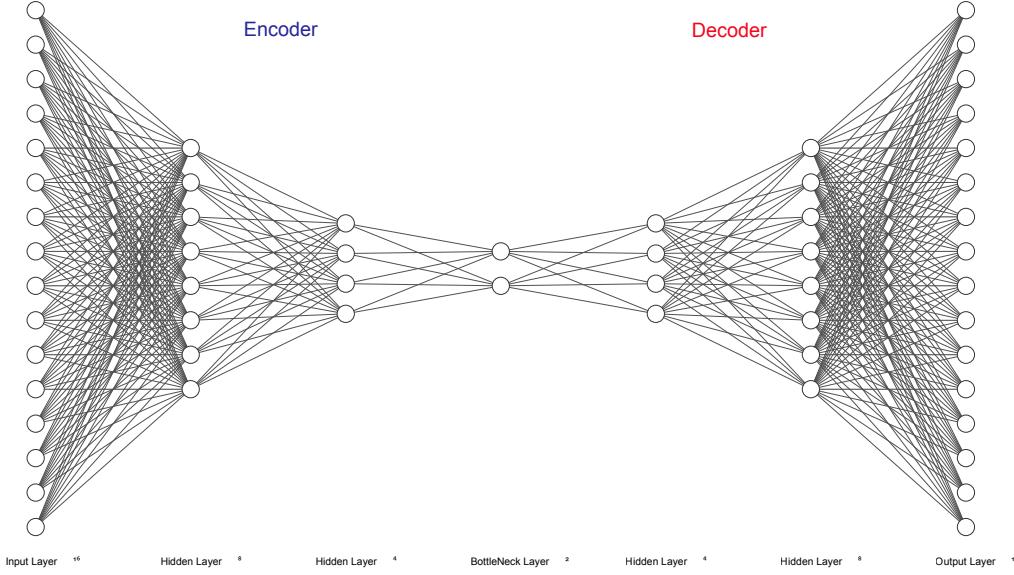


Figure 1.9: An example autoencoder structure which reduces a 16 dimensional input to 2. Draw with the aid of [Krizhevsky et al., 2012]

There are two features of an autoencoder that make it powerful. The first is the ability to sample your latent space using the decoder. This means we can establish features that correspond to gaps between our data points - which can have its application if the data used is sparse or incomplete. Next comes the inherent non-linearity of the model. As an autoencoder is just a neural network, the amount of information passed through each link between layers is governed by an activation function. Should this activation function be linear, the reduced dimension will be much akin to a PCA decomposition. However unlike PCA, in reducing the number of dimensions, we do not discard any data, but rather combine it - as per the nature of the links of the network. In trying to model/fit non-linearity, we have a range of activation functions to choose from. These are explained in ??.

1.3.5.1 Demonstration of non-linear activation functions

To demonstrate the effect of these we take a sample isopleth of Methane and Ozone, reduce it to two dimensions. This is then reconstructed back into three dimensions using the DR algorithms. Figure 1.10 shows the difference between the original dataset (Figure ??) and that of the PCA (Figure ??) and AutoEncoder (Figure ??) reconstructions. Here we see a loss in the non-linearity of the original data for the PCA reconstruction. Much of this can still be preserved when using a non-linear (\tanh) activation function for the AutoEncoder. Use of a linear activation function, however, produces a

similar result to the PCA algorithm.

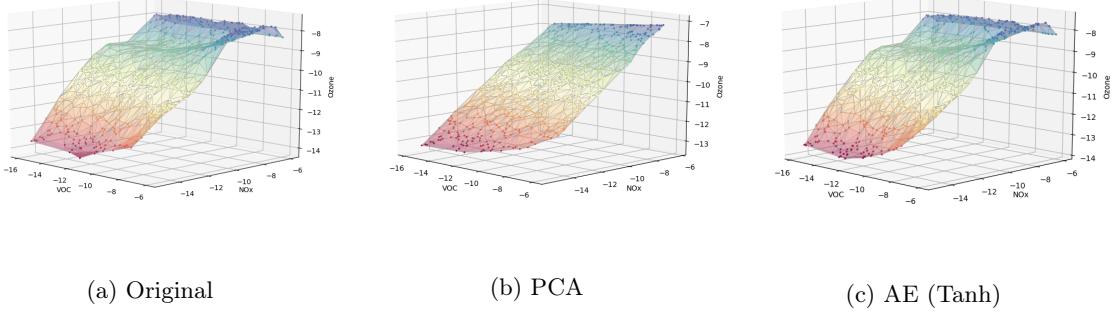


Figure 1.10: Comparing the result of the 2D encoding and decoding of an Ozone-NOx-Methane isopleth. The original data (a) is reduced to two dimensions and then reconstructed back into 3D. This is done with Principal Component Analysis (b) and an AutoEncoder (c). The original isopleth is created using 300 simulations of different initial conditions: NOx (variable), Methane (variable) and Ozone (constant). These were designed using a latin hypercube and converted into a surface plot using Delaunay triangulation.

1.3.6 Node2Vec

Finally Node2Vec is an embedding algorithm designed to generate vector representations of the nodes in an *undirected* and *unweighted* network. Although it can be used to reduce a complex network into a 2D vector (dimensionality reduction), for this experiment we shall only use it to generate a fingerprint for a species position within a mechanism network graph - and then apply this as an input to the DR methods above. This method of input creation has been found more computationally efficient, by circumventing the need for expensive composition, in producing better predictions on network-related tasks compared to more classical methods such as PCA [Grover and Leskovec, 2019].

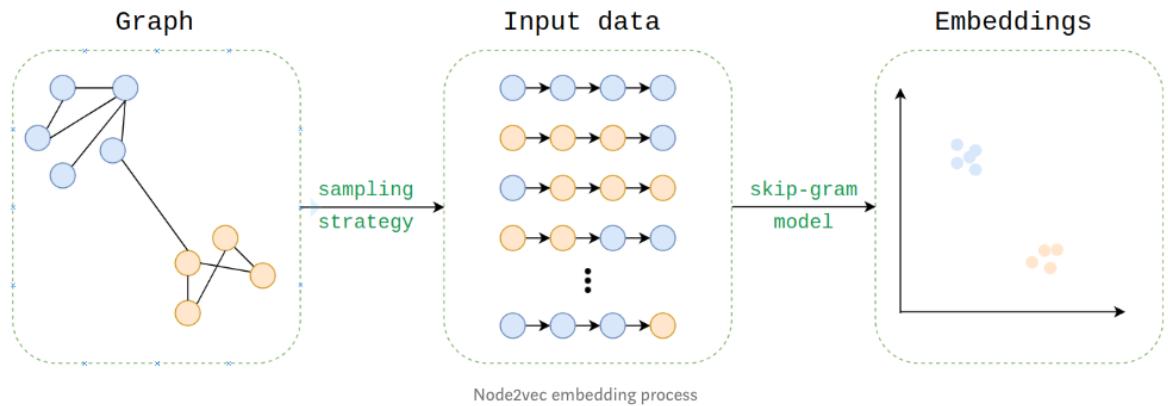


Figure 1.11: The process of converting a graph into a vector using Node2Vec. Source:[Cohen, 2018]

The process of converting the graph structure (Figure 1.11) into a numerical vector node embedding starts by taking a series of 2nd order random walks. These describe the neighbourhood of a node in the

form of a set of random walk paths, much in the same way words are dependant on their neighbours within a sentence: Equation 1.5.



This methodology allowed for the use of word2vec algorithm, converting the walk into a vector (subsection 1.3.6.2)

1.3.6.1 Sentence construction by sampling of a network

The probability and path depend both on a set of arguments and a random seed provided to the model. The return and input parameters (p & q) determine how fast we explore the network and our probability to leave the neighbourhood, Figure 1.12. In a network, where the previous path is from t to v , we may calculate the probability of returning to t as $1/p$, going to a mutual node connected between t and v as 1, and viewing a new node as $1/q$. If $q > 1$ we have a high probability to end up at nodes close to t , and with $q < 1$ we are likely to explore other nodes. Additionally if we chose $p > \max q, 1$ we are less likely to return to an already visited node ($p < \min q, 1$ is likely to generate a backwards step). Since we wish to generate a ‘local’ view, but do not wish to return to t we select $q \geq 1$ and $p > q$ our parameters as $p = 2.0, q = 1.1$. In the case of a weighted graph (something that we are *not* exploring within this chapter) the resultant *alpha* value calculated is further multiplied by the edge weight.

To run the simulation we use the python 2 code provided by the original paper [Grover and Leskovec, 2019] with a set of 50000 random walks, each of length 9. The reasoning behind this is that we have a large graph, with a power-law like structure (within only a couple of steps we are connected to every other species within the network - that is if we do not include the inorganics, which in this case we are).

NOTE: This process takes over a week to compute, and then the binary file containing all walks in character form approaches 10 GB, for the complete MCM. ...in serial...

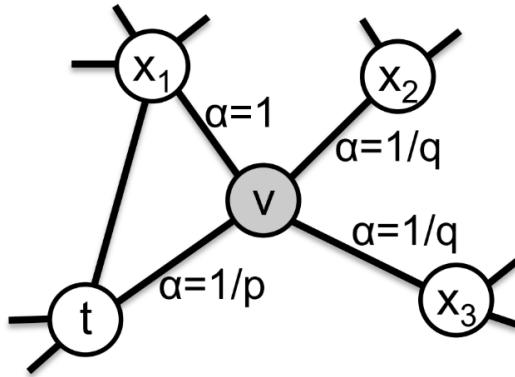


Figure 1.12: Calculation of the random walk path. Source:[Grover and Leskovec, 2019]

1.3.6.2 Word2Vec

Once we have constructed our random path ‘sentences’ (e.g. Equation 1.5), we can make use of Google’s word2vec algorithm, [Mikolov et al., 2013]. This is similar to an auto-encoder in many regards, but rather than learning word embeddings through reconstruction, it looks at the words (or species in our case) which neighbour each other in our corpus. This form of representation has found many uses beyond the realm of natural language processing. Some of these are objects, people, code, tiles, genes and graphs [Lynch, 2011; People2Vec, 2019; Alon et al., 2019; Jean et al., 2018; Du et al., 2019; ?].

1.3.7 Summary

There exist several methods of reducing a complex dataset into a smaller one. PCA is the simplest method to understand but is constrained to linear decompositions. AutoEncoders can have both a linear and non-linear response, based on the activation functions that they use, and t-SNE applies a non-linear grouping which mimics a complete force-directed graph.

Having defined each method, we next explain how they shall be evaluated (section 1.4), before applying them to the MCM in Equation 1.5.

1.4 Visualisation of clustering

In assessing the validity of clustered space, we require a level of exploratory data analysis. To reveal features of interest we plot the reduced 2D dataset and apply interactivity coupled with a selection of visualisation techniques described below. This section outlines the different visualisation methods which have been used.

1.4.1 Viewing the 2D species embeddings

Since the different DR algorithms return data on various scales, comparison between the outputs is not straightforward. To overcome this all outputs in both the x and y directions are normalised (scaled between $\{0,1\}$), before being plotted as a scatterplot.

1.4.2 Exposing overlapping data

To be able to interactively highlight data, a user must first be able to select it. This proved difficult if many of the nodes of a tight-knit cluster are overlapping. As an initial test, node sizes may be reduced, to check if this solves the problem, however, this may often leave points smaller than is possible to easily select. The other solution which was used is to create a force-directed graph where each point is strongly attracted to their initial position. Here we can apply collision detection, whilst still preserving the overall grouping of nodes within a cluster - a technique that was seen in ??.

1.4.3 Gooey Effect (Gaussian Blur)

Taking a quote from Reinhardt [1975]: “*The more stuff in it, the busier the work of art, the worse it is. More is less. Less is more.*” and combining it with the work from ??, we realise that showing each species, when observing overall clusters just add unnecessary clutter to the images. Instead, since we are only interested in the clusters as a unit, a ‘gooey effect’ filter can be applied. This works by merging nearby points into a single water-like blob using a gaussian blur⁸. Here since each point is allocated a colour, if a colour gradient exists, then there are multiple clusters occupying the same place. The aim of this is to reduce the cognitive load on the end-user by reducing the number of distinct objects that they need to take in.

1.4.4 Four Colours Theorem

When plotted, the number of clusters detected often exceeds the number of categorical colours available. In cartography, it has been noted that the colouring of neighbouring polygons should at most take 4 colours. This is the origin of the four colours theorem, Appel and Haken [1976], of which a greedy implementation has been applied.

The aim of this is to show item boundaries (for instance countries, or in our case clusters), whilst reducing ambiguity (if say two neighbours have the same colour). The algorithm I adapted uses the Delaunay tessellation scripts contained within DataDrivenDocuments.js (d3js) Bostock [2012]. This

⁸Here a gaussian blur of standard deviation 3.7 and a colour matrix [1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 37 -5] is used.

partitions our plane into polygon-regions with boundaries at the furthest distance from each point (Voronoi cells) Watson [1981]. First, we chose a random cell and assign it a colour. Next, all its neighbours are recursively iterated, giving them the lowest possible colour in a list, which does not match any of their neighbours. Although such a greedy approach does not produce an optimum result it allows for the colouring of data with ≤ 5 distinct colours, as is shown in Figure 1.13.

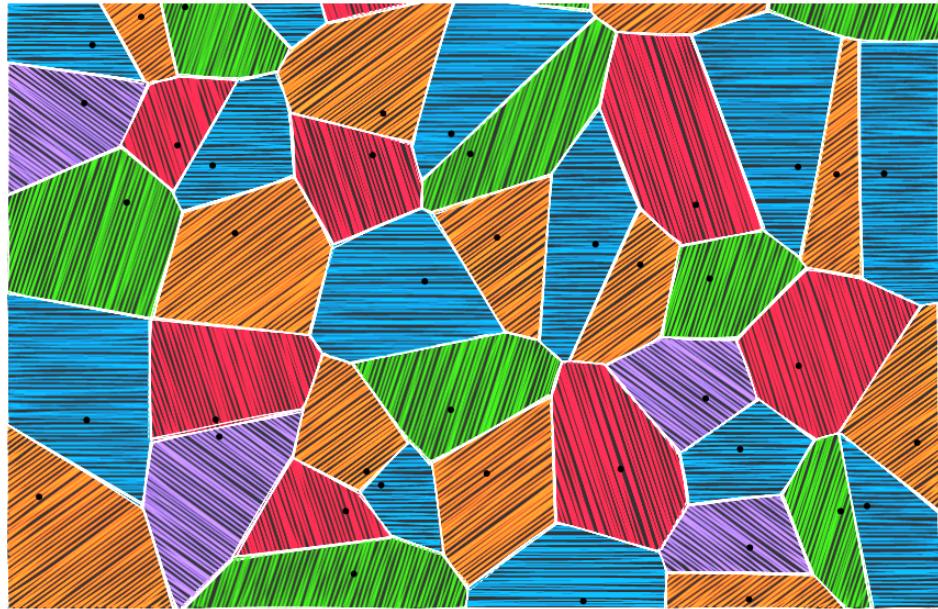


Figure 1.13: **An example 4 colour matching** This uses the first implementation of the algorithm mentioned in subsection 1.4.4. The greedy approach does not often find the optimum solution, which may result in 5 colours instead. Observable Notebook : Daniel Ellis [2019]

Having defined all the visualisation techniques we move on to explain the clustering algorithms which are used, and how ‘goodness of fit’ may be measured in the clustering context.

1.5 Cluster Evaluation

The previous section discussed methods of visualising the reduced data for use with interactive exploratory data analysis. In this section we look at the use of vector clustering algorithms⁹ (subsection 1.5.1) to highlight groups in a 2D dataset, as well an automated method of assessing the quality of the clusters selected (subsubsection 1.5.1.1) and feature extraction (subsection 1.5.2).

1.5.1 Automated selection of clusters

When it comes to clustering data points in a dataset, there exist a range of methods which may accomplish a task, Figure 1.14. Most often the k-means, [MacQueen, 1967], is used as it is fast and

⁹Vector clustering is the grouping of data based on their proximity or density to other nearby points

simple to understand. However, its linear method of partitioning cannot capture the splits between non-linear relationships of real data. The other problem is that an estimate for the number of expected clusters is required - something that is often unknown without prior understanding of the data. When this is the case often it is easier to manually select the nodes with interactivity.

In contrast, density-based clustering techniques such as GMM ([Pedregosa et al., 2011a]) or DBSCAN ([Ester et al., 1996]) tend to be better at locating non-linear trends in the data. The DBSCAN algorithm assesses the distribution of data across a certain location. This allows clusters with a high density of datapoints to be located without the need for a predefined number of clusters. Another method: OPTICS (Ordering Points To Identify the Clustering Structure), [Ankerst et al., 1999], shall be used¹⁰. This is an adaptation of the DBSCAN algorithm which does not require the specification of a minimum distance between points (for the density estimate)- instead, we specify a gradient for the distribution and the minimum number of points for a cluster to be classified.

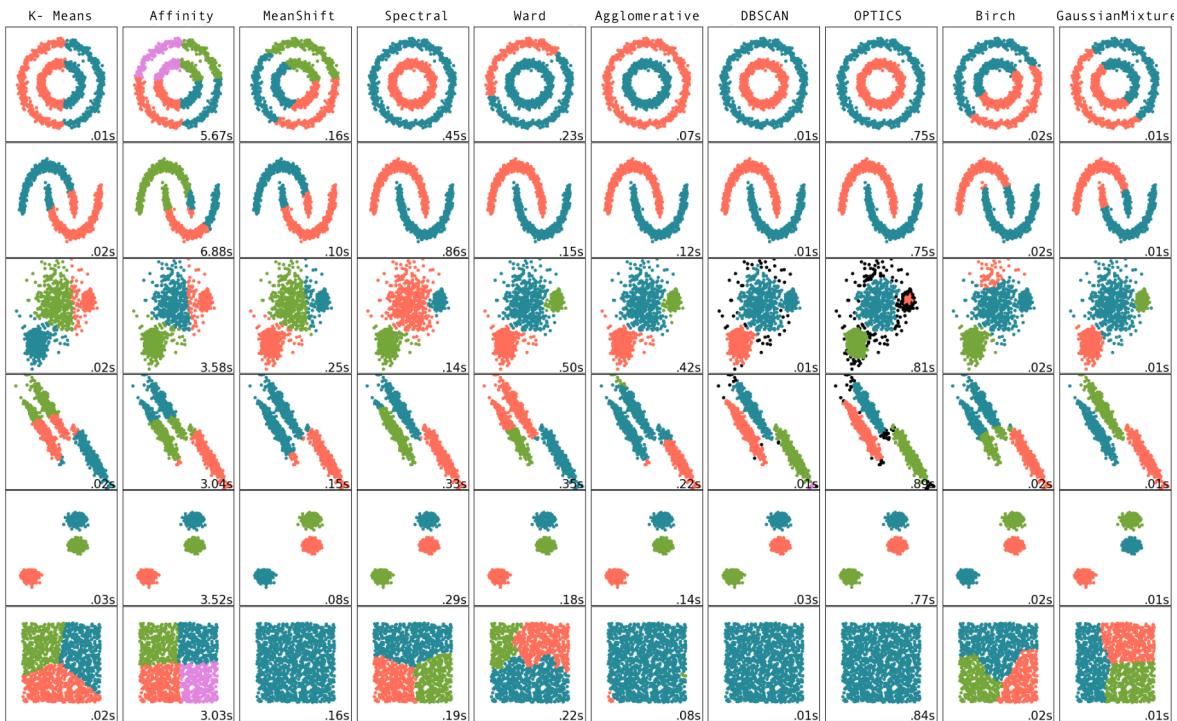


Figure 1.14: **A comparison of different clustering methods on a toy dataset.** The plot shows the performance of several vector clustering algorithms in Scikit-Learn. Cluster algorithms are represented across the horizontal axis and several types of datasets are across the vertical. Clustered groups are coloured. Source: sklearn [2019]

When deciding which algorithms to use, each algorithms' ability to partition non-linear data is considered. The first two rows of Figure 1.14 show data which cannot be partitioned linearly, here spectral, DBSCAN and optics are the only clustering algorithms to identify both correctly. It is for this reason

¹⁰If using Python 2, the library for this needs to be extracted from the sci-kit-learn library for python 3 package and altered to run with the previous version. (See copy in attached code.)

that we shall look at these for the remainder of the chapter.

In selecting a value for the results section, several clustering algorithms, with a wide range of input parameters are run. From these, the values with the best silhouette coefficient (subsubsection 1.5.1.1) are taken.

1.5.1.1 Clustering (Silhouette) coefficient

The silhouette measure is a tool used for assessing the validity of a set of clusters. Here each cluster is represented as a silhouette, based on the comparison of its tightness and separation. To calculate the silhouette coefficient we look at the intra-cluster a and the mean inter-cluster¹¹ distance b . The silhouette cluster can then be described using ??:

$$s(i) = \frac{b(i) - a(i)}{\max a(i), b(i)} \quad (1.6)$$

This gives a value $-1 \leq s(i) \leq 1$. Values near zero suggest overlapping clusters, 1 - dense, well-separated clusters and negative values indicate that a sample may have been incorrectly classified. In using this method we can get an overview of how well individual objects lie within their assigned cluster.

1.5.2 Feature Extraction

Upon establishing a set of DR datasets, and their groups (the clusters of species they contain), it is important to evaluate what input features they represent. Rather than doing this manually we make use of Random Forests.

1.5.2.1 Random Forrests

Random forests, [Breiman, 2001], are a subset of ML algorithms called ensemble learning. This means that they train a large number of decision trees, each on a random subset of the original features. A decision tree is a tree formed from a series of conditionals¹², much like a perceptron network (??) with binary activation functions. Random forests introduce a level of additional randomness by selecting only a subset on which to create each decision tree. This may introduce a higher bias, but lowers the overall model variance which creates a better (more robust) model. Such methods have been applied to replacing the computationally expensive process of chemistry integration of GEOS-Chem

¹¹Inside and between different clusters.

¹²Questions with a True/False answer

(a global 3D model of tropospheric chemistry) [Keller and Evans, 2019] and the prediction of global sea-surface iodine based on observations coupled with sea-surface temperature, depth, and salinity, [Sherwen et al., 2019].

1.5.2.2 Calculating importance using Random Forrests

Since random forests are in essence a collection of decision trees, it is possible to generate a ‘decision tree aggregate’ to visualise the ensemble structure of the random forest, [Ellis and Sherwen, 2019] (Figure 1.15). Alternatively, if all that is required is the relative importance of each feature, the RandomForestClassifier from Pedregosa et al. [2011b] provides a quick and easy way of understanding which features matter, [Géron, 2017]. This works by aggregating the weighted nodes which use a certain feature using the number of samples and then scales the result to 1. We use this method to access the overall importance of features within each DR output and identify the differences between clusters.

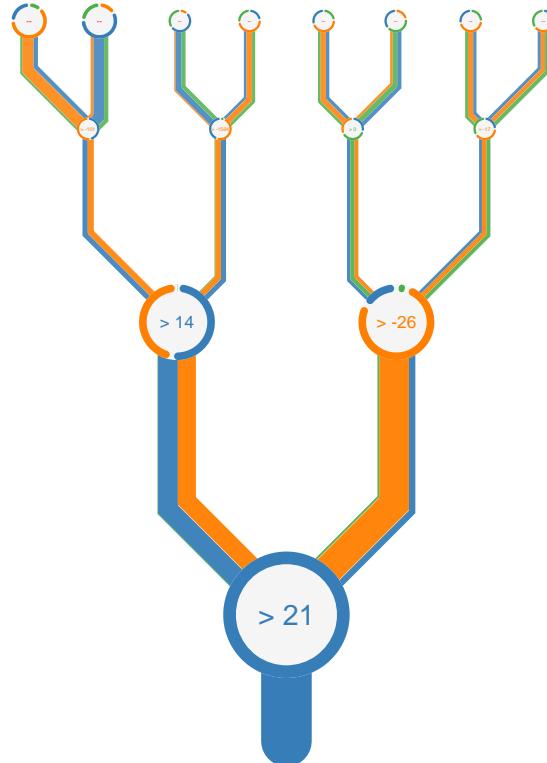


Figure 1.15: A decision tree aggregate from a random forest plotted with the Epiphyte version of the TreeSurgeon program [Ellis and Sherwen, 2019]. The data originates from Sherwen et al. [2019] and the imporance of Tempearature (blue), Depth (orange) and Chlorophyll *a* (green).

NOTE: The only downside is that Random Forrests are in themselves ML techniques which also need to be evaluated. To do this, as they are simply being used as indicators of cluster properties which we

are to explore further, we can initiate a collection of 300 random Forrest classifiers, from which we take the median.

1.6 Results

1.6.1 CLuster distribution

Start with the visual comparison and compare it with the silhouette values.

Principle Component Analysis

DR	input	silhouette	groups
PCA	fngroups	0.9122	141
PCA	protocol	0.8761	149
PCA	node2vec	0.8569	3
PCA	maccs	0.6563	2
PCA	mqn	0.4041	8
PCA	smiles	0.3648	6
PCA	fingerprints	0.3529	6
PCA	spec	0.3364	6

Table 1.2: The inputs to the PCA dimensionality reduction algorithm sorted by the best obtained silhouette coefficient.

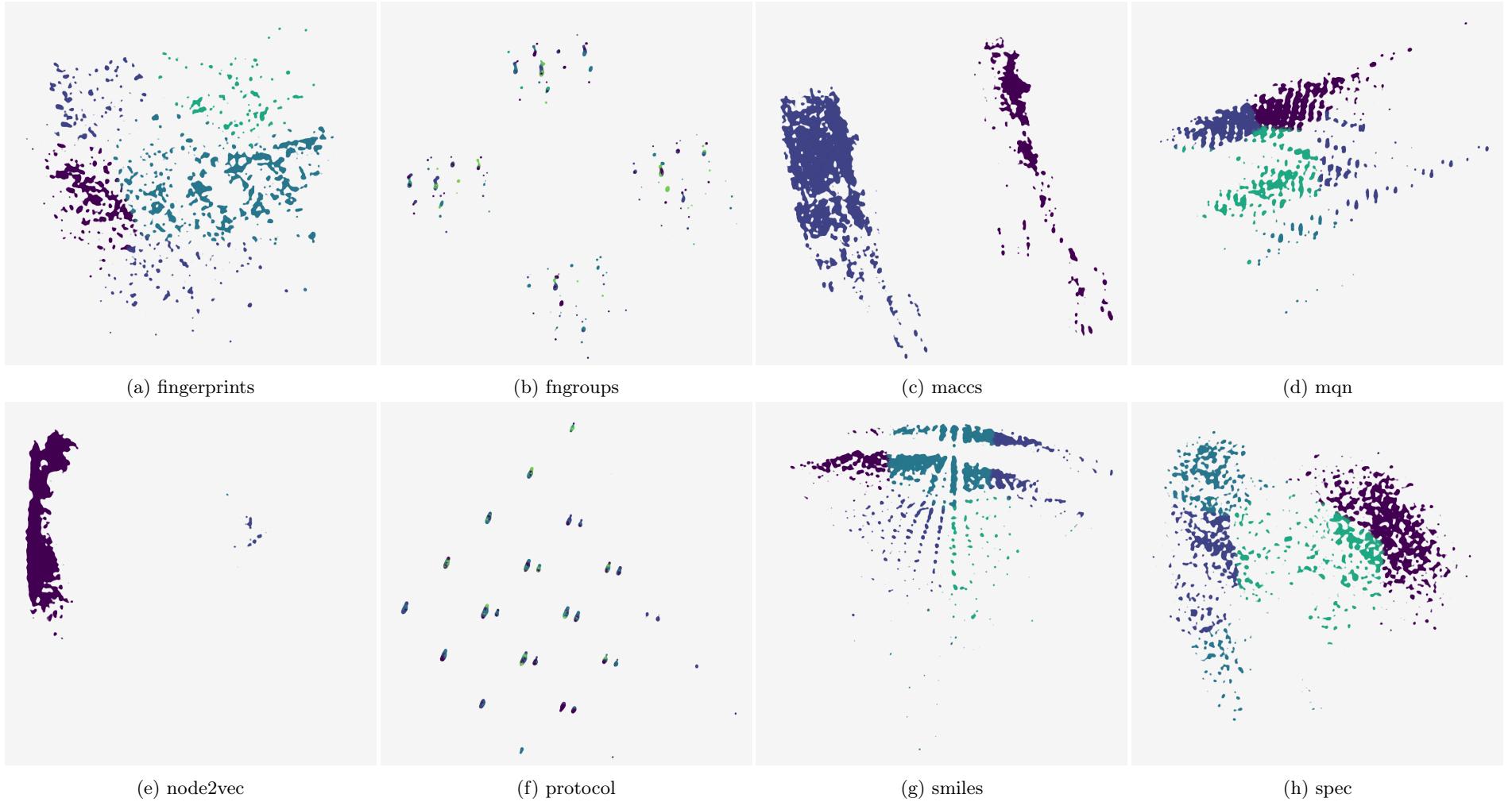


Figure 1.16: **Comparing clusters for all inputs after a reduction to 2 dimensions using Principle Component analysis.** Each graphs has undergone several clustering algorithms under a range of parameters. The result with the best silhouette coefficient have been chosen. Colours follow the greedy 4 colour theorem and are there only to indicate contrast between cluster boundaries.

Auto Encoder Encoding

DR	input	silhouette	groups
AE	fngroups	0.9249	140
AE	protocol	0.8992	27
AE	smiles	0.6897	5
AE	mqn	0.6572	12
AE	maccs	0.6241	3
AE	node2vec	0.5476	5
AE	spec	0.4238	3
AE	fingerprints	0.3189	8

Table 1.3: The inputs to the AutoEncoder dimensionality reduction algorithm sorted by the best obtained silhouette coefficient.



Figure 1.17: **Comparing clusters for all inputs after a reduction to 2 dimensions using an AutoEncoder.** Each graph has undergone several clustering algorithms under a range of parameters. The result with the best silhouette coefficient have been chosen. Colours follow the greedy 4 colour theorem and are there only to indicate contrast between cluster boundaries.

t-Distributed Stochastic Neighbor Embedding

DR	input	silhouette	groups
t-SNE	fngroups	0.7458	106
t-SNE	protocol	0.5688	51
t-SNE	smiles	0.4808	6
t-SNE	node2vec	0.4359	6
t-SNE	maccs	0.4295	3
t-SNE	spec	0.3781	35
t-SNE	mqn	0.3684	8
t-SNE	fingerprints	0.3539	6

Table 1.4: The inputs to the t-SNE dimensionality reduction algorithm sorted by the best obtained silhouette coefficient.

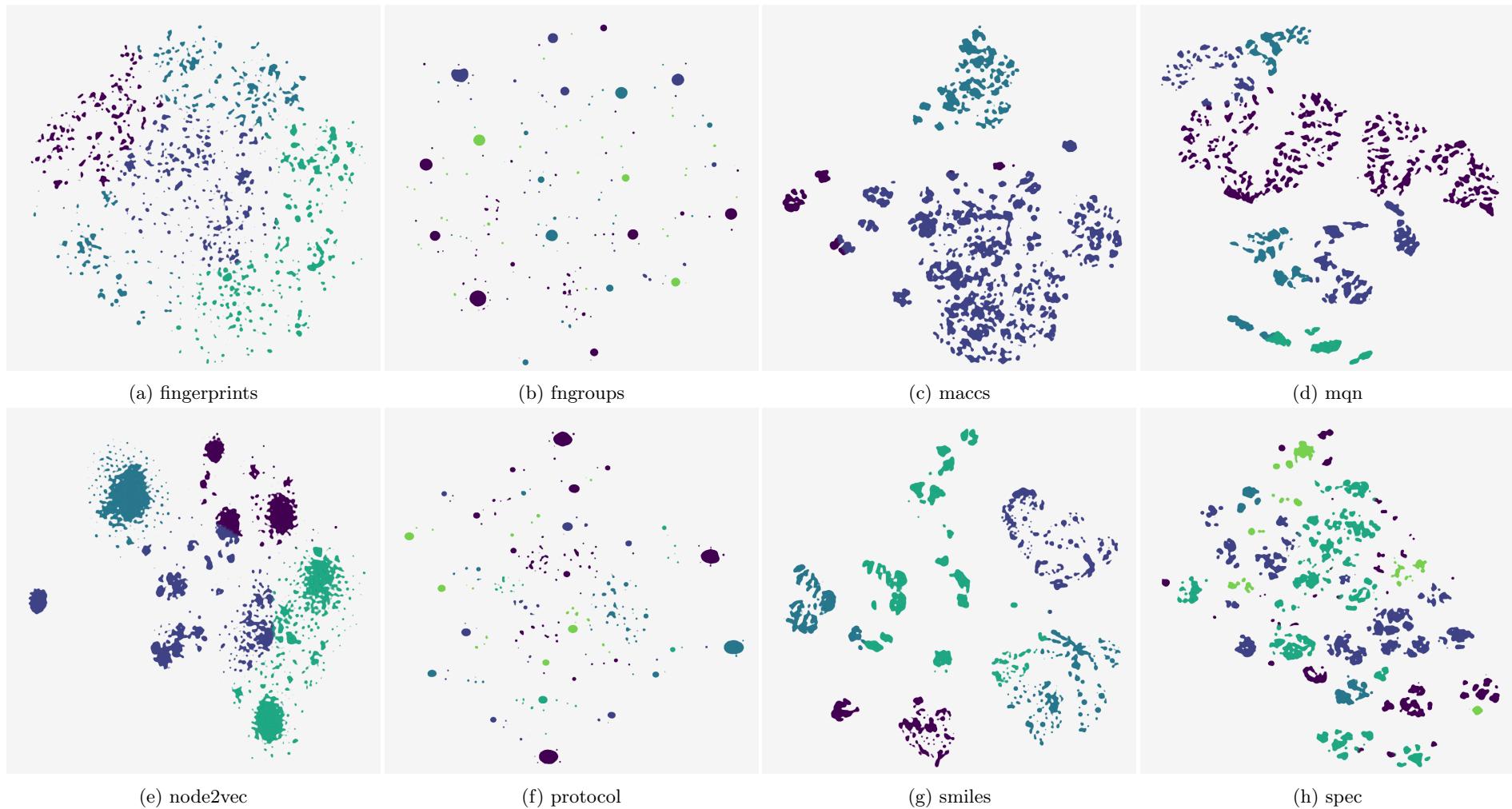


Figure 1.18: **Comparing clusters for all inputs after a reduction to 2 dimensions using t-SNE.** Each graphs has undergone several clustering algorithms under a range of parameters. The result with the best silhouette coefficient have been chosen. Colours follow the greedy 4 colour theorem and are there only to indicate contrast between cluster boundaries.

Bibliography

- Alon, U., Zilberstein, M., Levy, O., and Yahav, E. (2019). Code2Vec: Learning Distributed Representations Of Code.
- Anderson, C. (2008). The End Of Theory: The Data Deluge Makes The Scientific Method Obsolete. *online*.
- Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). Optics: Ordering points to identify the clustering structure. *SIGMOD Rec.*, 28(2):49–60.
- Appel, K. and Haken, W. (1976). Every planar map is four colorable. *Bull. Amer. Math. Soc.*, 82(5):711–712.
- Aumont, B., Szopa, S., and Madronich, S. (2005). Modelling the evolution of organic carbon during its gas-phase tropospheric oxidation: Development of an explicit model based on a self generating approach. *Atmospheric Chemistry and Physics*, 5(9):2497–2517.
- Baillargeon, R. and Carey, S. (2012). Core cognition and beyond: The acquisition of physical and numerical knowledge. *Early childhood development and later outcome*.
- Bostock, M. (2012). D3.js - data-driven documents.
- Box, G. E. P. (1976). Science And Statistics. *Journal of the American Statistical Association*, 71(356):791–799.
- Breiman, L. (2001). Random Forests. *Machine learning*, 45(1):5–32.
- Cohen, E. (2018). Node2Vec: Embeddings For Graph Data. <https://towardsdatascience.com/node2vec-embeddings-for-graph-data-32a866340fef>. Accessed: 2019-10-22.
- Daniel Ellis (2019). D3-Fourcolour Voronoi. <https://observablehq.com/@wolfiehx/d3-fourcolour-voronoi>. Accessed: 2019-7-6.
- Dataman (2019). Convolutional Autoencoders For Image Noise Reduction. <https://towardsdatascience.com/convolutional-autoencoders-for-image-noise-reduction-32fce9fc1763>. Accessed: 2020-2-26.
- Descartes, R. and Lafleur, L. J. (1960). *Meditations On First Philosophy*. Bobbs-Merrill New York.
- Du, J., Jia, P., Dai, Y., Tao, C., Zhao, Z., and Zhi, D. (2019). Gene2Vec: Distributed Representation Of Genes Based On Co-Expression. *BMC genomics*, 20(Suppl 1):82.

- Durant, J. L., Leland, B. A., Henry, D. R., and Nourse, J. G. (2002). Reoptimization Of Mdl Keys For Use In Drug Discovery. *Journal of chemical information and computer sciences*, 42(6):1273–1280.
- Ellis, D. (2019). Chemical Kinetic Interactions Cover Image. <https://s100.copyright.com/AppDispatchServlet?startPage=i&publisherName=Wiley&publication=kin&contentID=10.1002%2Fkin.21180&endPage=i&title=Cover+Image%2C+Volume+50%2C+Issue+6>. Accessed: 2019-6-6.
- Ellis, D. and Sherwen, T. (2019). Wolfiex/treesurgeon: Wollemia.
- Ester, M., peter Kriegel, H., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press.
- F.R.S., K. P. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.
- Géron, A. (2017). *Hands-On Machine Learning With Scikit-Learn And Tensorflow: Concepts, Tools, And Techniques To Build Intelligent Systems*. O'Reilly Media.
- Grover, A. and Leskovec, J. (2019). Node2vec: Scalable feature learning for networks. Accessed: 2019-10-21.
- Hamadache, M. and Lee, D. (2017). Principal Component Analysis Based Signal-To-Noise Ratio Improvement For Inchoate Faulty Signals: Application To Ball Bearing Fault Detection. *International journal of control, automation, and systems*, 15(2):506–517.
- Heller, S., McNaught, A., Stein, S., Tchekhovskoi, D., and Pletnev, I. (2013). Inchi - The Worldwide Chemical Structure Identifier Standard. *Journal of cheminformatics*, 5(1):7.
- Hernandez, W. and Mendez, A. (2018). Application Of Principal Component Analysis To Image Compression. In Göksel, T., editor, *Statistics - Growing Data Sets and Growing Demand for Statistics*. InTech.
- Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441.
- Jean, N., Wang, S., Samar, A., Azzari, G., Lobell, D., and Ermon, S. (2018). Tile2Vec: Unsupervised Representation Learning For Spatially Distributed Data.
- Jolliffe, I. T. and Cadima, J. (2016). Principal Component Analysis: A Review And Recent Developments. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 374(2065):20150202.

- Jones, E., Oliphant, T., Peterson, P., et al. (2001–). Scipy: Open source scientific tools for Python. <http://www.scipy.org/>.
- Keller, C. A. and Evans, M. J. (2019). Application of random forest regression to the calculation of gas-phase chemistry within the geos-chem chemistry model v10. *Geoscientific Model Development*, 12(3):1209–1225.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet Classification With Deep Convolutional Neural Networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.
- Landrum, G., Tosco, P., Kelley, B., sriniker, gedeck, NadineSchneider, Vianello, R., Dalke, A., Cole, B., AlexanderSavelyev, Turk, S., Ric, Swain, M., Vaucher, A., N, D., Wójcikowski, M., Pahl, A., JP, strets123, JLVarjo, O’Boyle, N., Berenger, F., Fuller, P., Jensen, J. H., Sforna, G., DoliathGavid, Cosgrove, D., Nowotka, M., Leswing, K., and van Santen, J. (2019). Rdkit 2019-03-2 (q1 2019) release.
- Leite, N. M. N., Pereira, E. T., Gurjão, E. C., and Veloso, L. R. (2018). Deep convolutional autoencoder for eeg noise filtering. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 2605–2612.
- Lynch, H. (2011). *Infant Places, Spaces And Objects: Exploring The Physical In Learning Environments For Infants Under Two*. PhD thesis.
- Maaten, L. v. d. and Hinton, G. (2008). Visualizing Data Using T-Sne. *Journal of machine learning research: JMLR*, 9(Nov):2579–2605.
- MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif. University of California Press.
- (MDL), M. I. S. (1984). Maccs-ii.
- Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation Of Word Representations In Vector Space.
- Morozov, A. (2016). Modelling biological evolution: Linking mathematical theories with empirical realities. *Journal of Theoretical Biology*, 405:1 – 4. Advances in Modelling Biological Evolution: Linking Mathematical Theories with Empirical Realities.
- Nguyen, K. T., Blum, L. C., van Deursen, R., and Reymond, J.-L. (2009). Classification Of Organic Molecules By Molecular Quantum Numbers. *ChemMedChem*, 4(11):1803–1805.

- Noble, C. E. (1957). Human Trial-And-Error Learning. *Psychological reports*, 3(2):377–398.
- Oliphant, T. (2006). Guide to numpy.
- Oliveira, B., Pereira, F., de Ara ojo, R., and Ramos, M. (2006). The hydrogen bond strength: New proposals to evaluate the intermolecular interaction using dft calculations and the aim theory. *Chemical Physics Letters*, 427(1):181 – 184.
- Parsons, J., Holmes, J. B., Rojas, J. M., Tsai, J., and Strauss, C. E. M. (2005). Practical Conversion From Torsion Space To Cartesian Space For In Silico Protein Synthesis. *Journal of computational chemistry*, 26(10):1063–1068.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011a). Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011b). Scikit-Learn: Machine Learning In Python . *Journal of Machine Learning Research*, 12:2825–2830.
- People2Vec (2019). People2Vec. <http://people2vec.org/>. Accessed: 2019-10-22.
- Powell, V. (2020). Principal Component Analysis Explained Visually. <http://setosa.io/ev/principal-component-analysis/>. Accessed: 2020-2-25.
- Probst, D. and Reymond, J.-L. (2018). Smilesdrawer: Parsing And Drawing Smiles-Encoded Molecular Structures Using Client-Side Javascript. *Journal of chemical information and modeling*, 58(1):1–7.
- rdkit (2019). Rdkit.
- Reinhardt, A. (1975). *Art-As-Art: The Selected Writings Of Ad Reinhardt*. Documents of 20th-century art. Viking Press.
- Roberts, R. (1989). *Serendipity: Accidental Discoveries In Science*. Wiley Science Editions. Wiley.
- Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.
- Sherwen, T., Chance, R. J., Tinel, L., Ellis, D., Evans, M. J., and Carpenter, L. J. (2019). A machine-learning-based global sea-surface iodide distribution. *Earth System Science Data*, 11(3):1239–1262.

- sklearn (2019). Comparing Different Clustering Algorithms On Toy Datasets — Scikit-Learn 0.21.3 Documentation. https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html. Accessed: 2019-10-21.
- Spahn, V., Del Vecchio, G., Labuz, D., Rodriguez-Gaztelumendi, A., Massaly, N., Temp, J., Durmaz, V., Sabri, P., Reidelbach, M., Machelska, H., Weber, M., and Stein, C. (2017). A Nontoxic Pain Killer Designed By Modeling Of Pathological Receptor Conformations. *Science*, 355(6328):966–969.
- T. Leube, B., Inglis, K., J. Carrington, E., Sharp, P., Shin, F., R. Neale, A., Manning, T., Pitcher, M., J. Hardwick, L., Dyer, M., Blanc, F., Claridge, J., and J. Rosseinsky, M. (2018). Lithium transport in li 4.4 m 0.4 m Å 0.6 s 4 (m = al 3+ , ga 3+ and m Å= ge 4+ , sn 4+): Combined crystallographic, conductivity, solid state nmr and computational studies. *Chemistry of Materials*, 30.
- Wang, S.-G. and Schwarz, W. H. E. (2009). Icon Of Chemistry: The Periodic System Of Chemical Elements In The New Century. *Angewandte Chemie*, 48(19):3404–3415.
- Watson, D. F. (1981). Computing The N-Dimensional Delaunay Tessellation With Application To Voronoi Polytopes*. *The Computer Journal*, 24(2):167–172.
- Weininger, D. (1988). Smiles, A Chemical Language And Information System. 1. Introduction To Methodology And Encoding Rules. *Journal of chemical information and computer sciences*, 28(1):31–36.
- Yu-ChenLo (2018). Machine learning in chemoinformatics and drug discovery. *Drug Discovery Today*, 23(8):1538 – 1546.