Unsupervised ML SOM and how to choose a Data Science method

UNIVERSITY *of* WASHINGTON

Quick Review of the methods we learned

- Statistical analysis
- Supervised ML
  - Linear regression
  - NN,
  - KNN,
  - Decision Tree
  - SVM
- Unsupervised ML
  - K-Means Clustering
  - PCA
  - ..... Why another one,

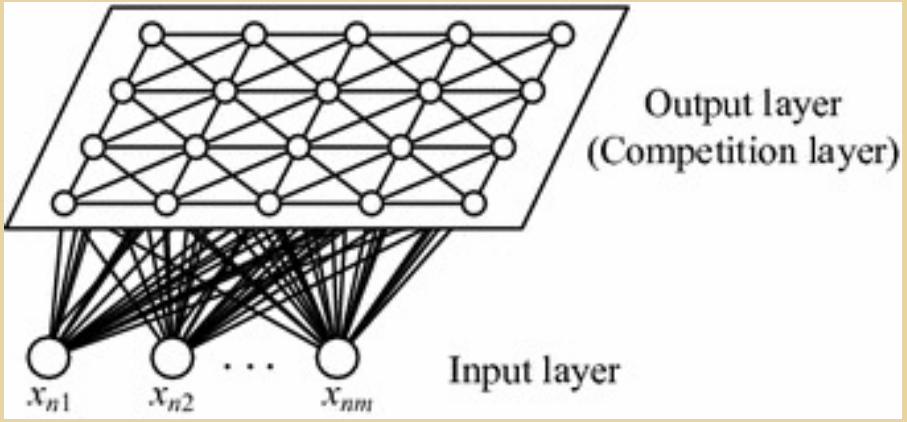
Why another method, SOM

- Demonstrate some data science methods that are not widely used or well known, but also can be very useful for material informatic study
- Introduce a method I have used, and feel is adequate to the uniqueness of many materials study applications.
- Demonstrate how various data science methods can be used together to drive improved results
- Demonstrate a few projects using the same methods so that we can understand a methods from user point of view

What is Self-Organizing Map (SOM)

- An Unsupervised ML method
- Dimensional reduction, enabling powerful visualizations of the data: Advantages of SOM
  - K-Means does clustering, but neither dimensionality reduction nor visualization
  - PCA does dimensionality reduction, enabling visualization to certain level (not applicable if the first 3 principal components won't represent the data well), however, it does not perform clustering. Besides, the visualization does not keep the original topographic information.
- Give some insights into how data is clustered in high dimensions

What is SOM



- You can think of SOM as an artificial neural network with a single neuronal layer, whose neurons are arranged in a two-dimensional matrix.
  - The 2D matrix can been seen as a position map that captures the characteristics of the data
- Merits of SOM
  - Effective in training big datasets
  - Since this is a 2D matrix, visualization of the resulting map is possible
  - kept the topography of the original data,
  - Possible to present the Euclidean distance between data points

Algorithm of SOM

- Normalization of the input data, all features will be distributed more balancely
- Initialization: each (x,y) position in the map is assigned a weight for each input neuron, thus associating a weight vector for each map position.

- Iteration:
  - > Choose a sample from dataset
  - > Calculate Euclidean distance between that sample and each weight vector
  - > The (x,y) position ”closest” to the sample is declared the Best Matching Unit

$$d = \min(||\vec{x} - \vec{w_{ij}}||) = \min(\sqrt{\sum_{t=0}^{n}[\vec{x}(t) - \vec{w}_{ij}(t)]^2})$$

  - > The weights vector for the BMU get adjusted to more closely match the sample. Amount of adjustment (learning) decreases as we go through iterations

$$w_{ij}(t+1) = w_{ij}(t) + \alpha_i(t)[x(t) - w_{ij}(t)], \quad \text{or}$$
$$w_{ij}(t+1) = w_{ij}(t) + \alpha_i(t)\beta_{ij}(t)[x(t) - w_{ij}(t)]$$

$$\sigma(t) = \sigma_0 \cdot \exp(\frac{-t}{\lambda}), \quad \text{where } t = 1, 2, 3 \dots n$$
$$\alpha(t) = \alpha_0 \cdot \exp(\frac{-t}{\lambda}), \quad \text{where } t = 1, 2, 3 \dots n$$

  - > The weights vector for neighbors of the BMU also get adjusted, to a lesser extent. The number of neighbors and how much they get adjusted also depends on hyperparameters and the number of iterations.

$$\beta_{ij}(t) = \exp(\frac{-d^2}{2\sigma^2(t)}), \quad \text{where } t = 1, 2, 3 \dots n$$

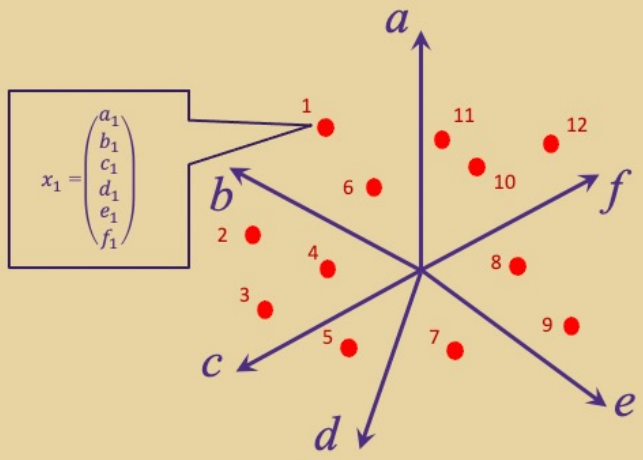
- Convergence:
  - > Max number of iterations
  - > Monitoring of topological error
- Reference: https://link.springer.com/article/10.1007/BF00337288

Self-Organizing Map (SOM)

How does it work?

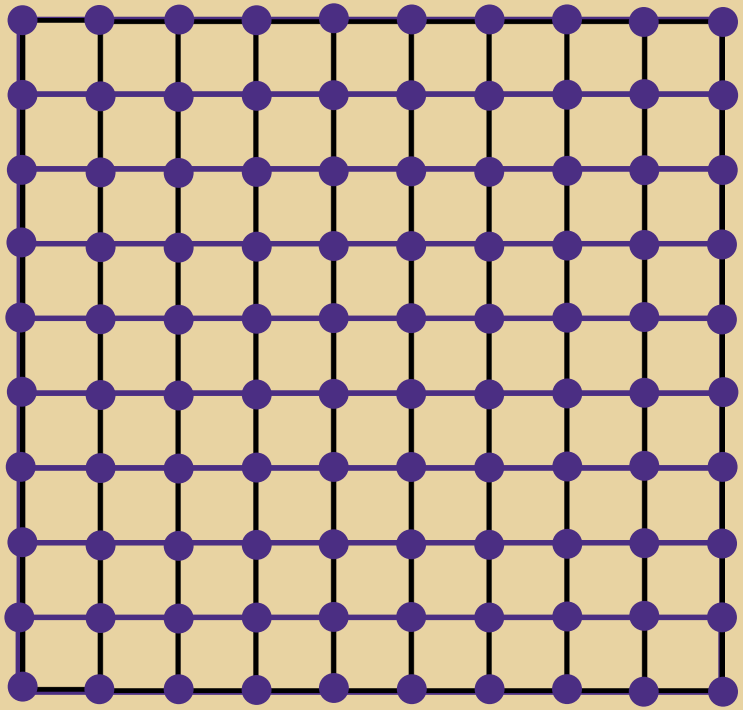
	Density [g/cm]	Glass transition temperature (Tg) [°C]	Yield Stress [MPa]	Young's modulus [GPa]	Tri-axial Strength [MPa]	Cure rate [%/fs]
#1	1.188	145.9	216.6	2.901	174.9	57.38
#2	1.211	210.5	253.6	3.165	177.4	49.27
#3	1.275	198.2	248.2	3.531	200.8	40.80
#4	1.273	183.4	252.9	3.303	206.3	46.05
#5	1.241	158.4	289.2	3.612	223.9	56.73
#6	1.263	208.0	263.5	3.036	215.9	54.42
#7	1.296	204.1	275.6	3.068	239.5	42.14
#8	1.293	217.5	251.9	3.101	214.5	48.62
#9	1.205	228.3	405.7	5.095	301.5	60.49
#10	1.217	269.4	376.9	4.925	297.0	45.55
#11	1.290	276.5	397.8	4.564	285.1	43.58
#12	1.282	307.6	388.8	5.056	265.1	49.62

$$x_i = \begin{pmatrix} a_i \\ b_i \\ c_i \\ d_i \\ e_i \\ f_i \end{pmatrix}$$

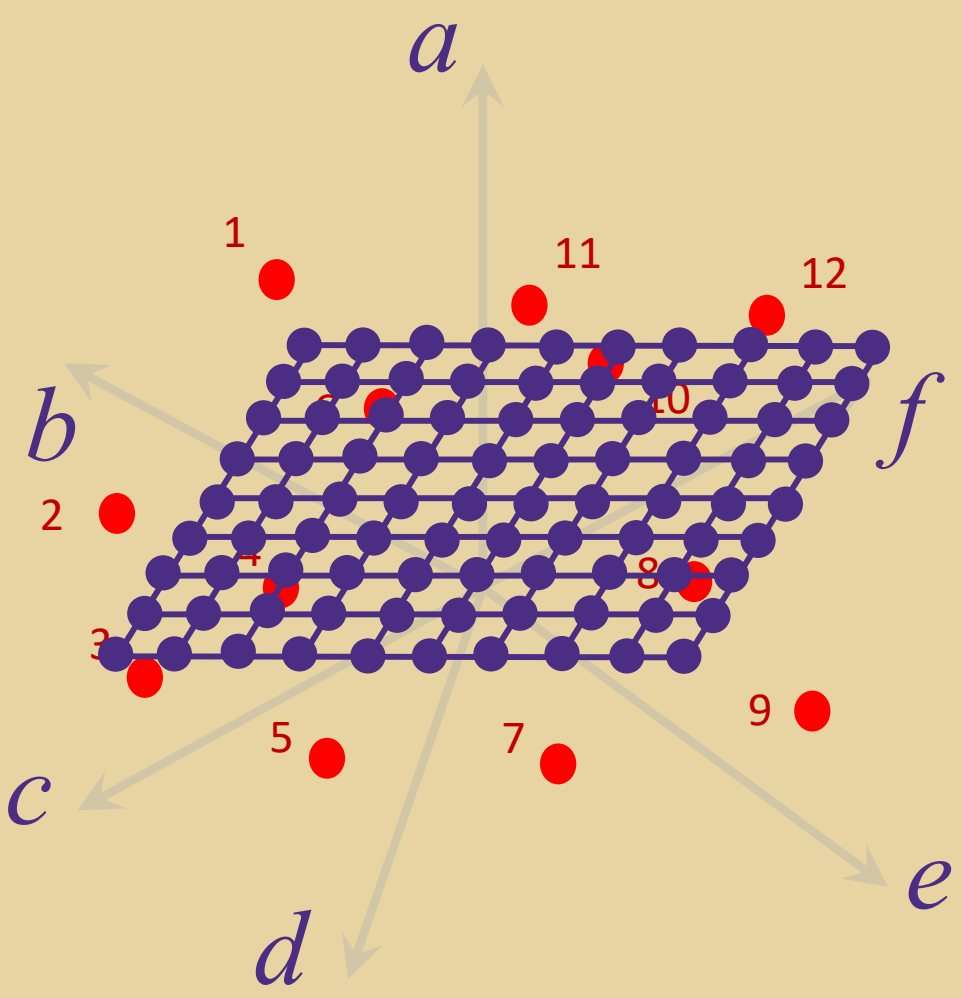


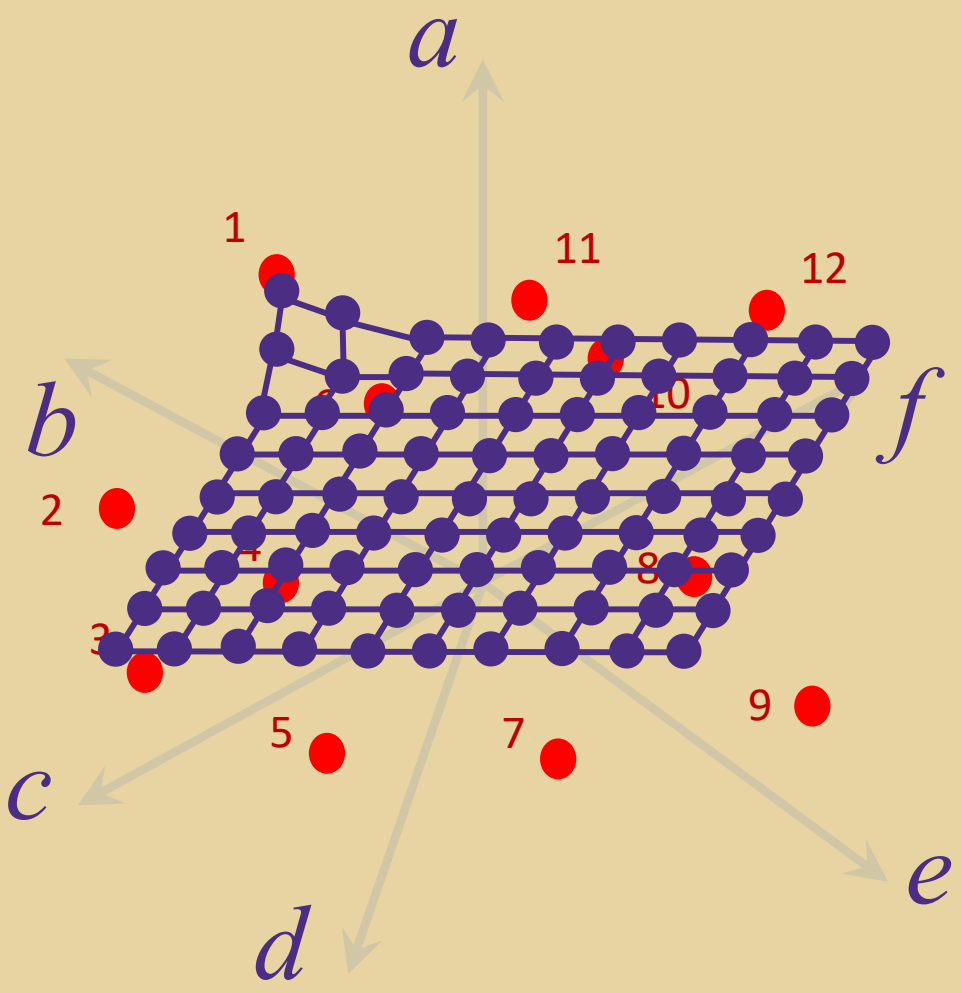
$a \sim f$:Properties

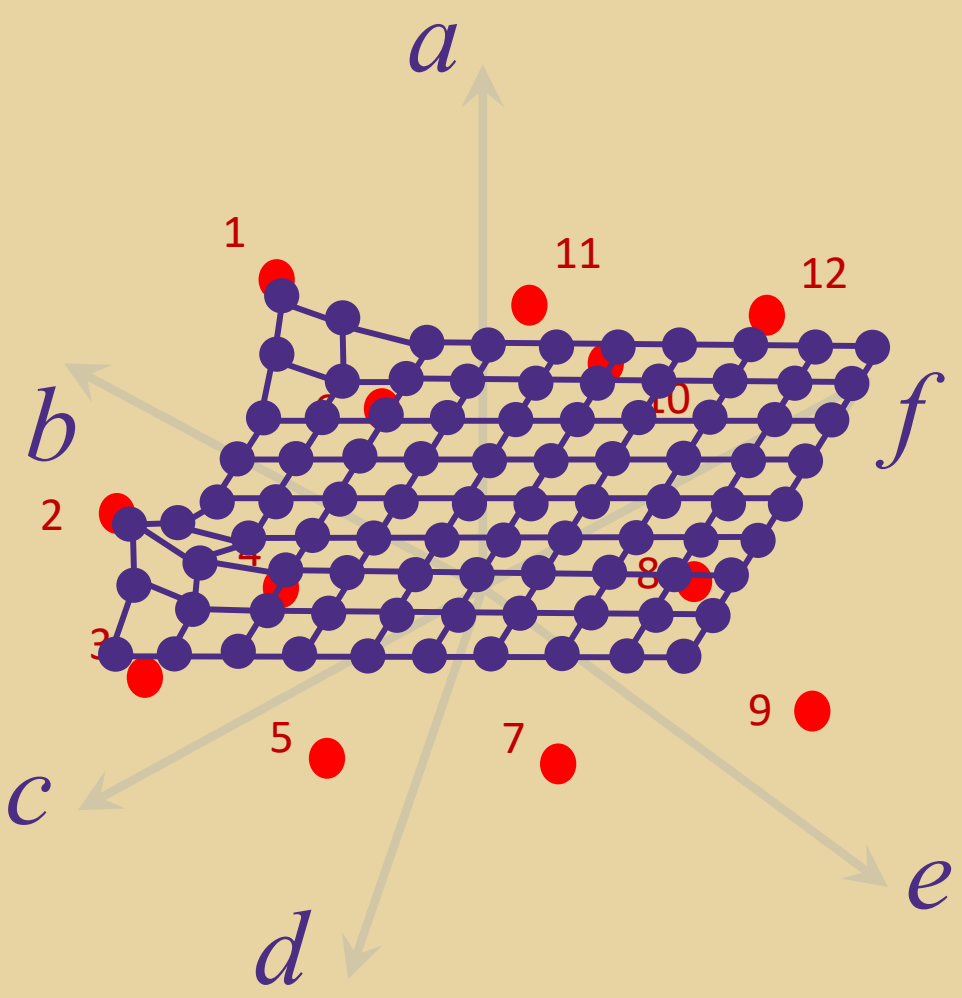
Two Dimensional Mesh structure

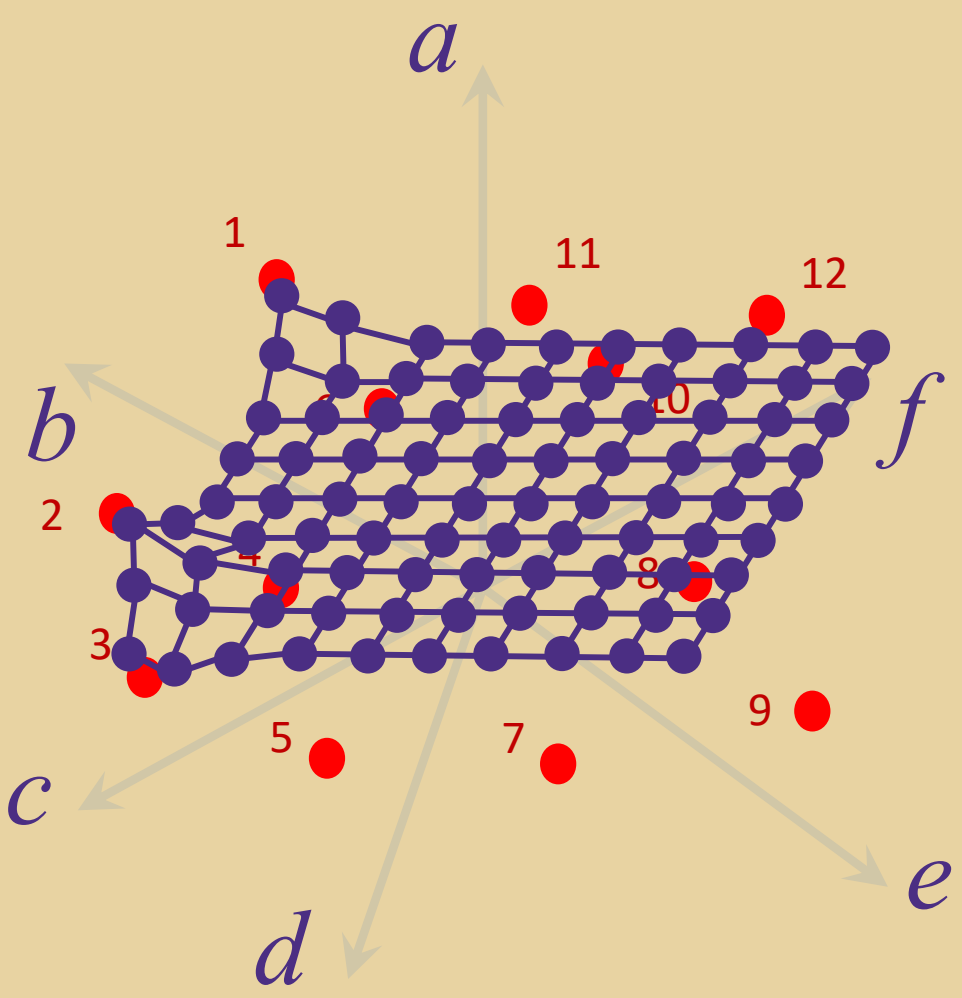


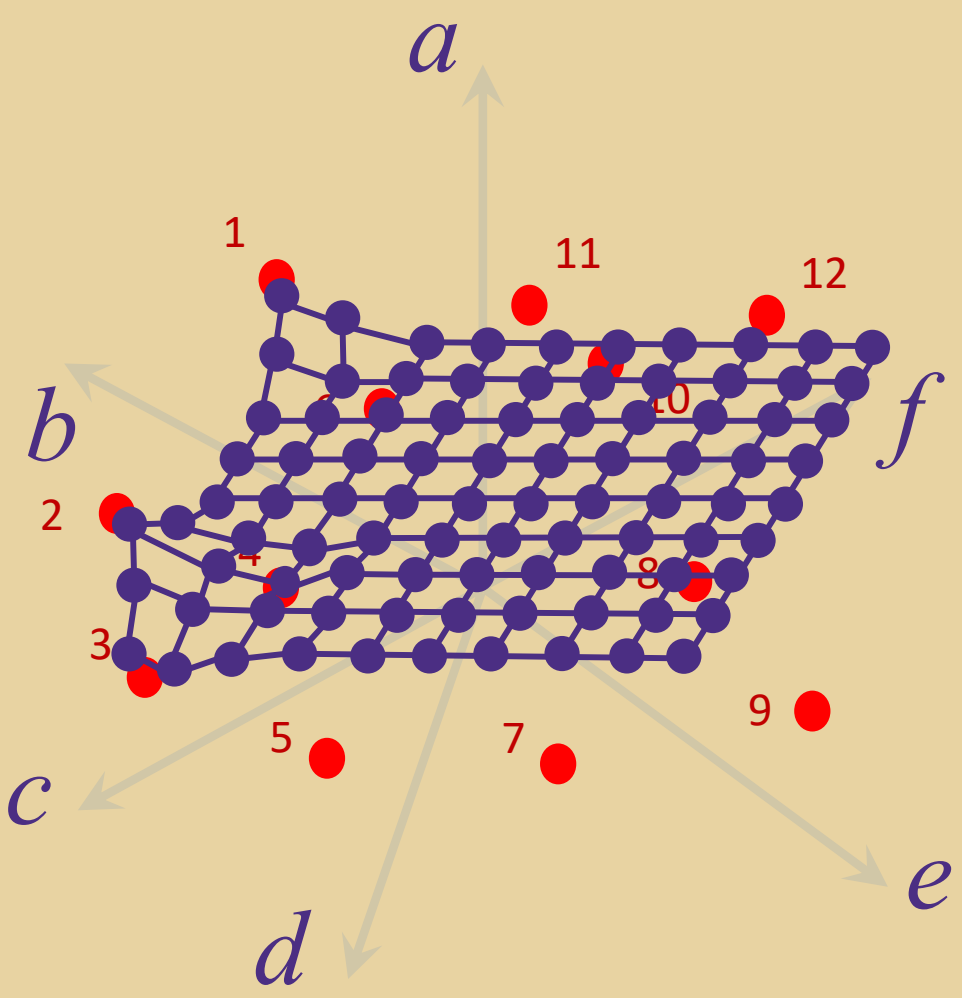
Each connection can deform

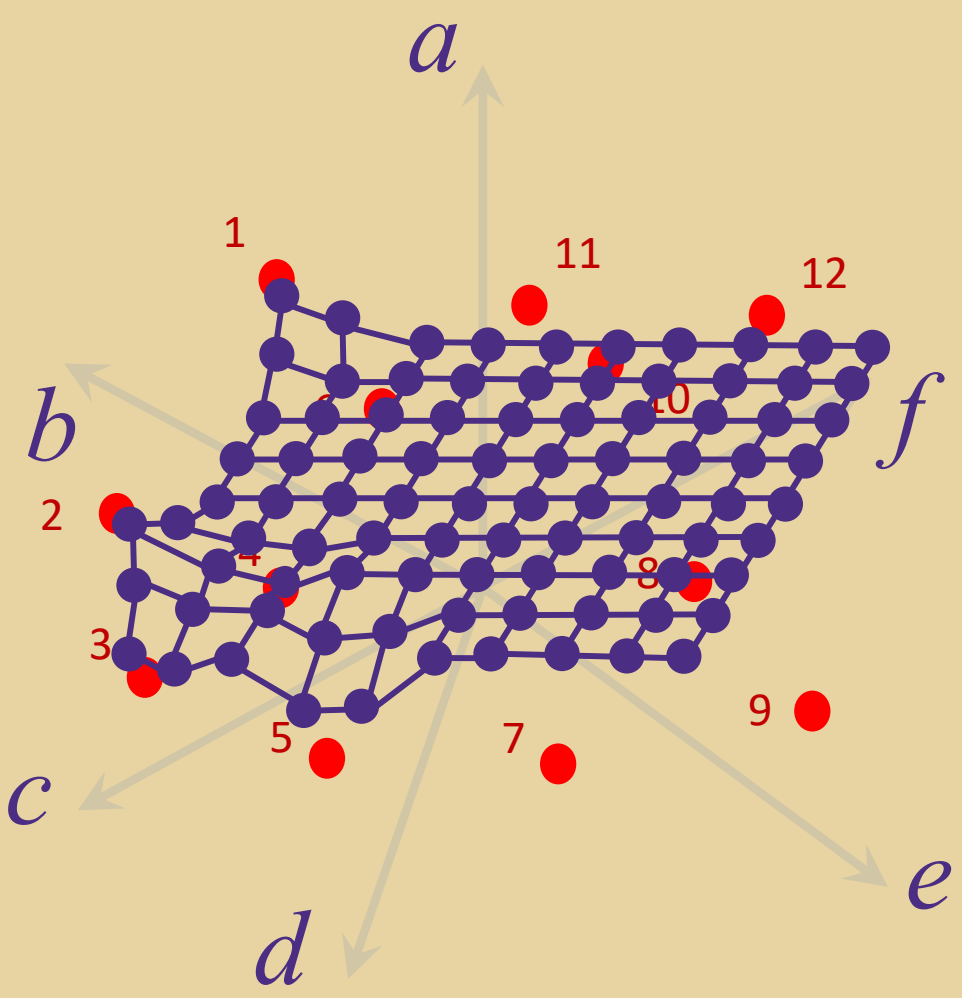

a
b
c
d
e
f
1
11
12
2
9
5
7

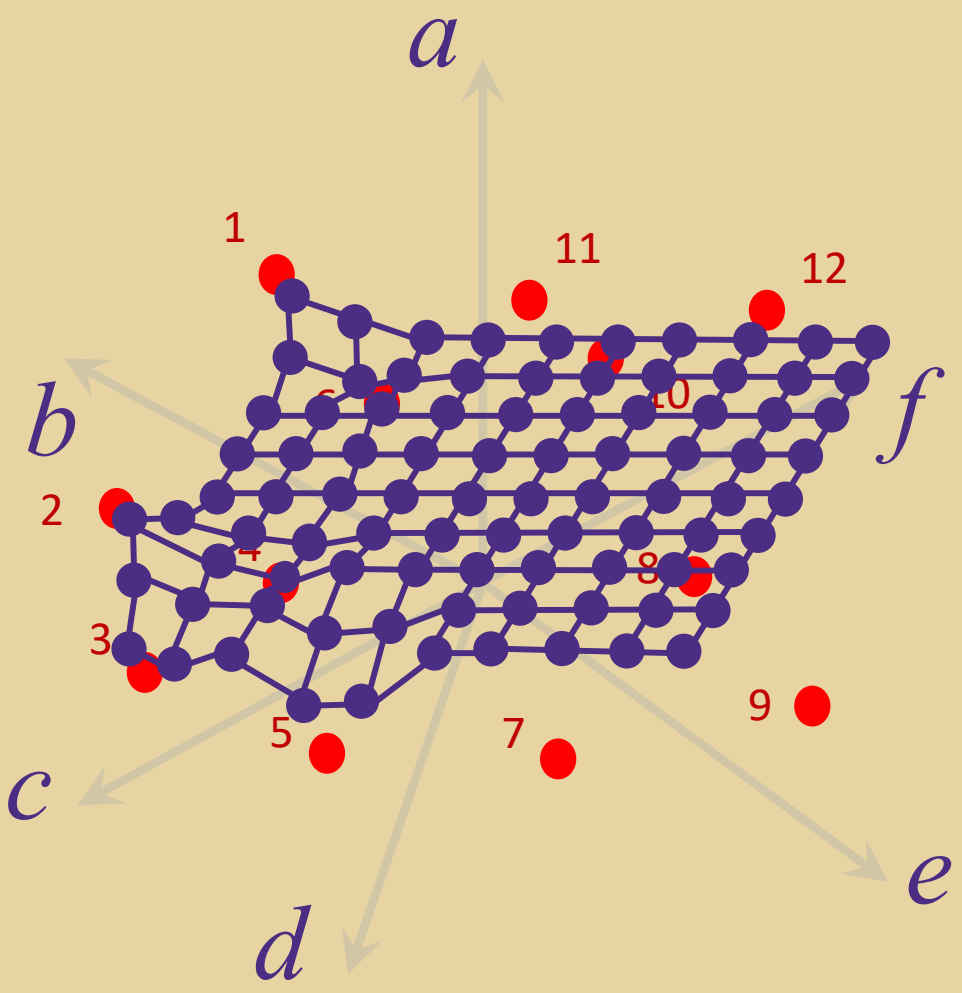

a
b
c
d
e
f
1
2
3
4
5
7
8
9
10
11
12

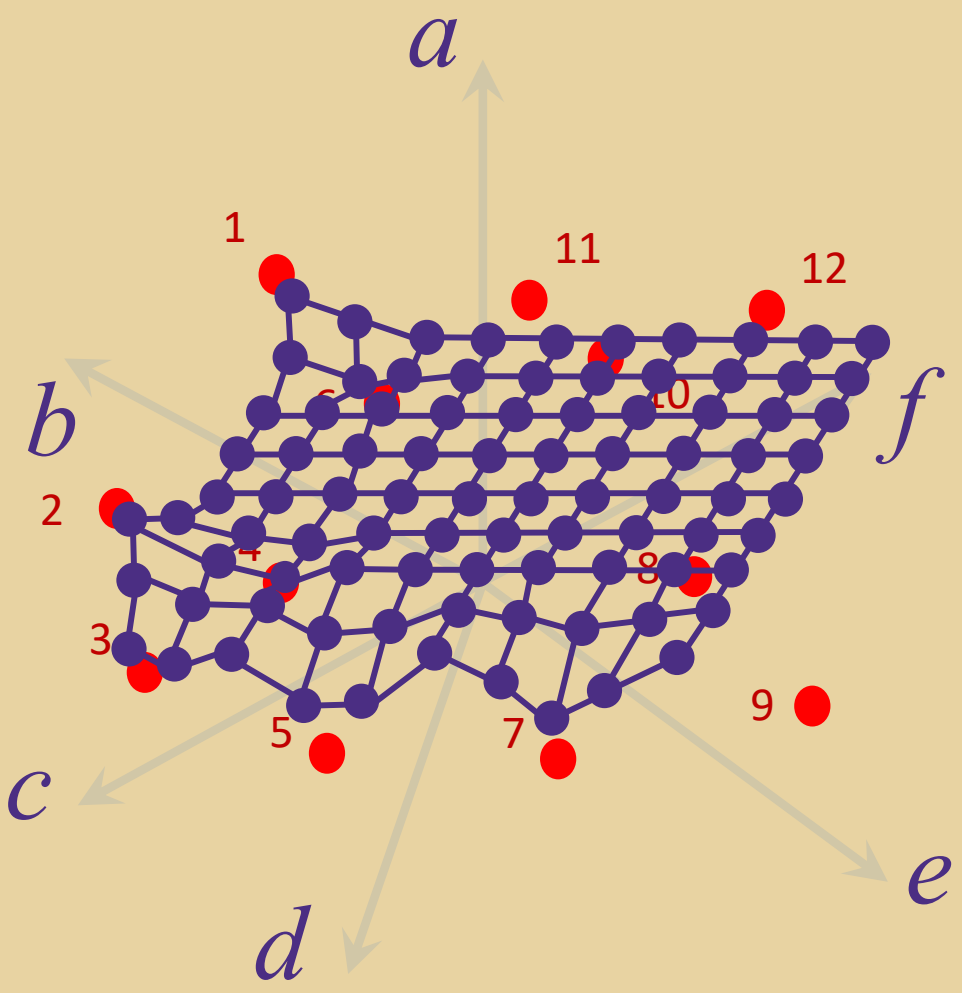

a
b
c
d
e
f
1
11
12
2
9
5
7

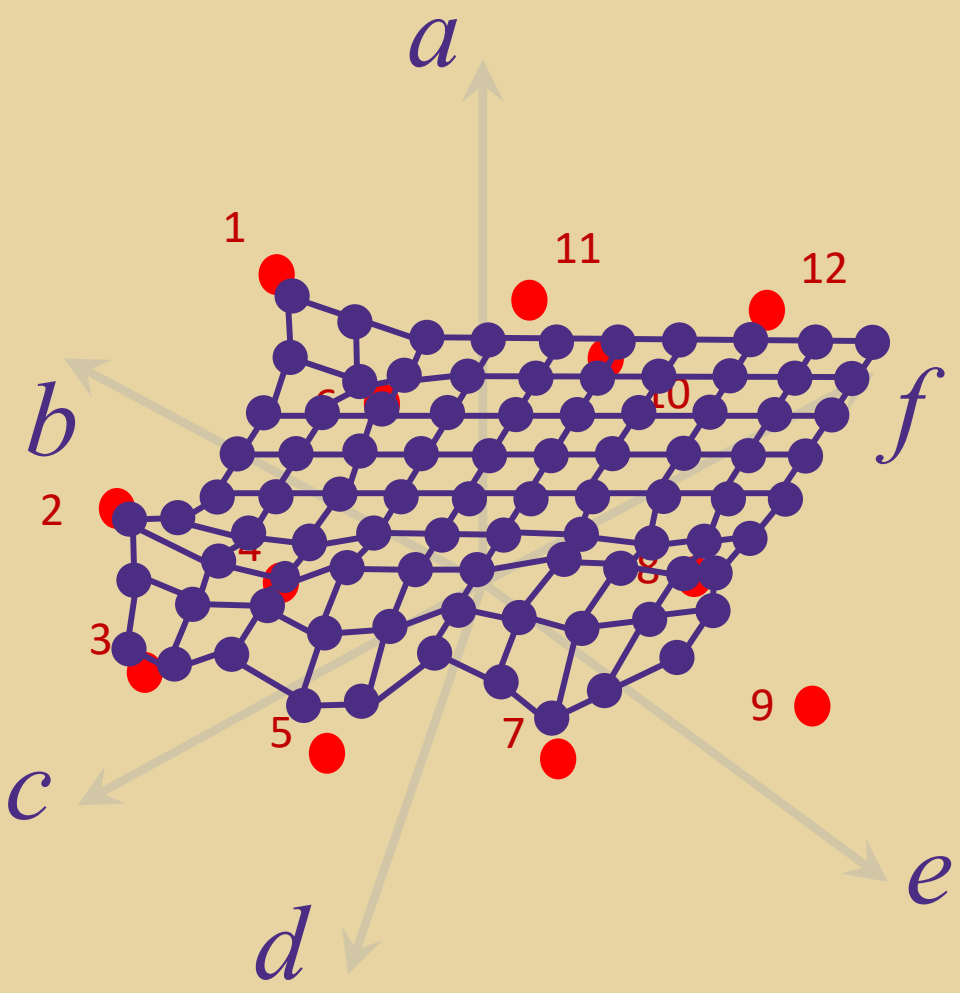

a
b
c
d
e
f
1
2
3
4
5
7
8
9
10
11
12

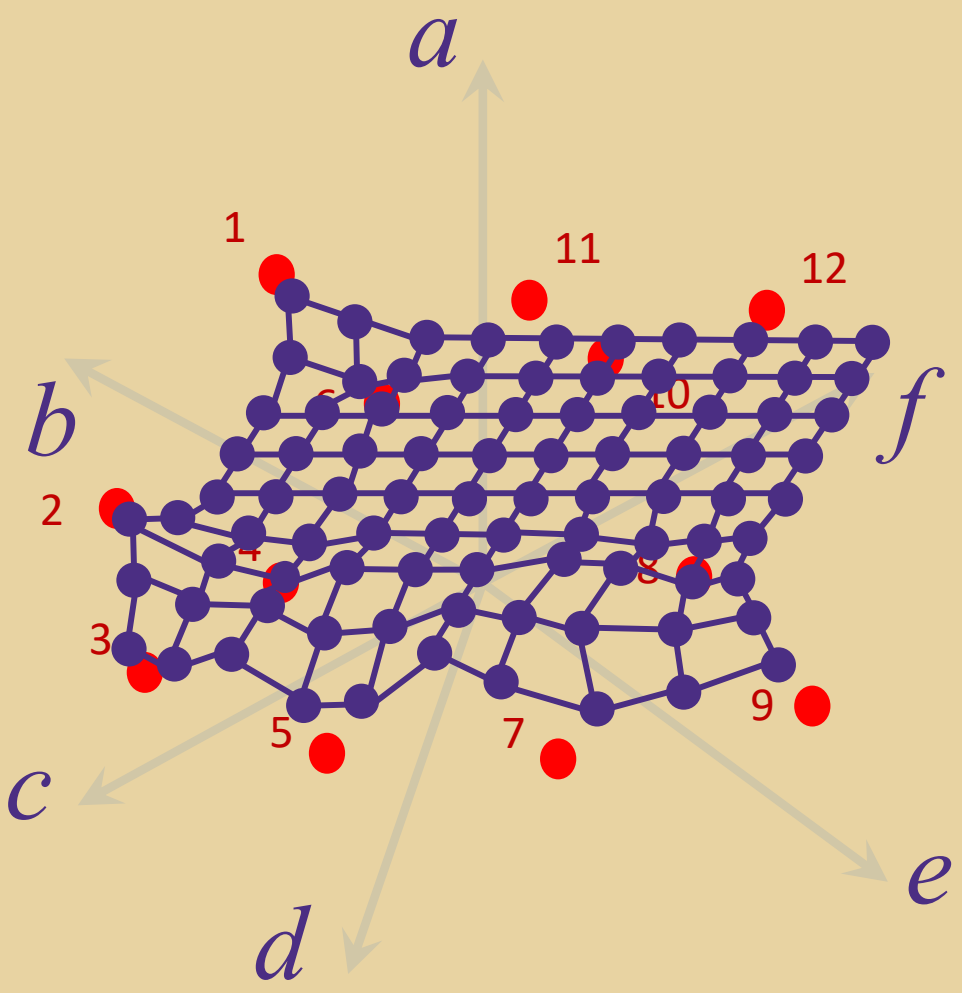

a
b
c
d
e
f
1
2
3
4
5
7
8
9
10
11
12

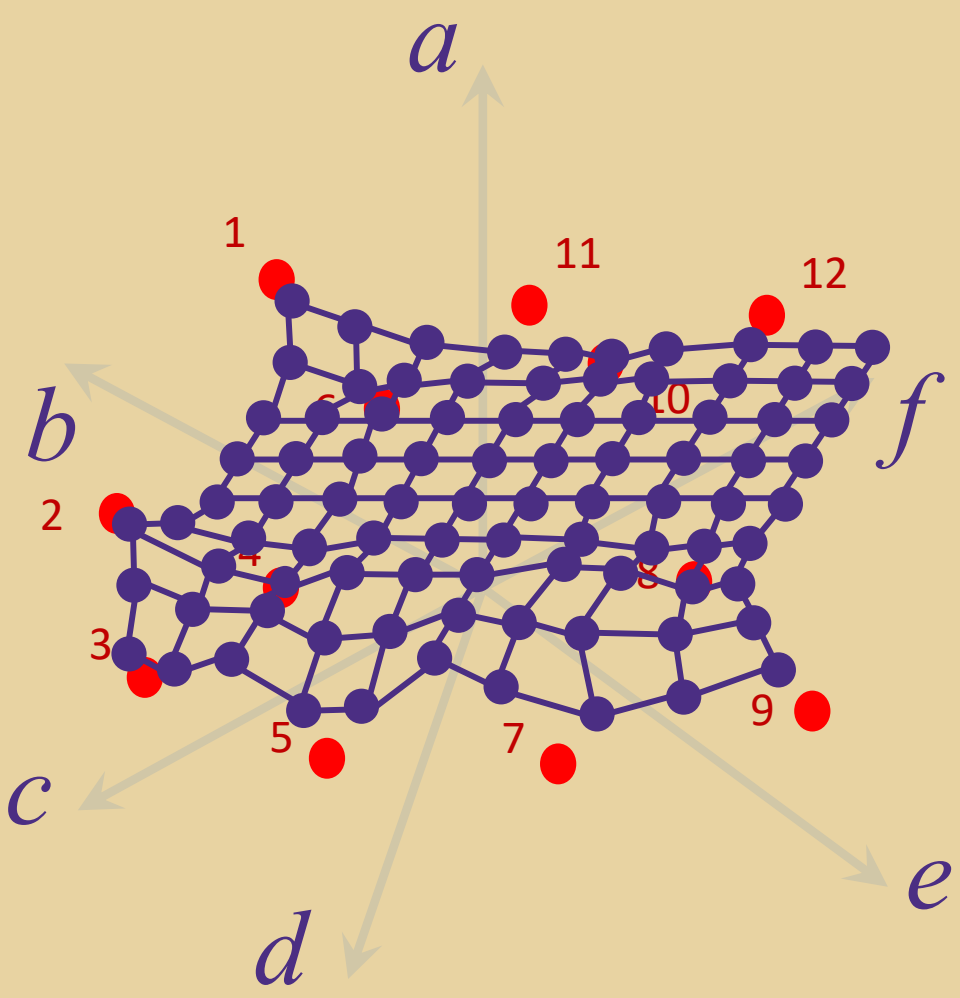

a
b
c
d
e
f
1
2
3
4
5
7
8
9
10
11
12

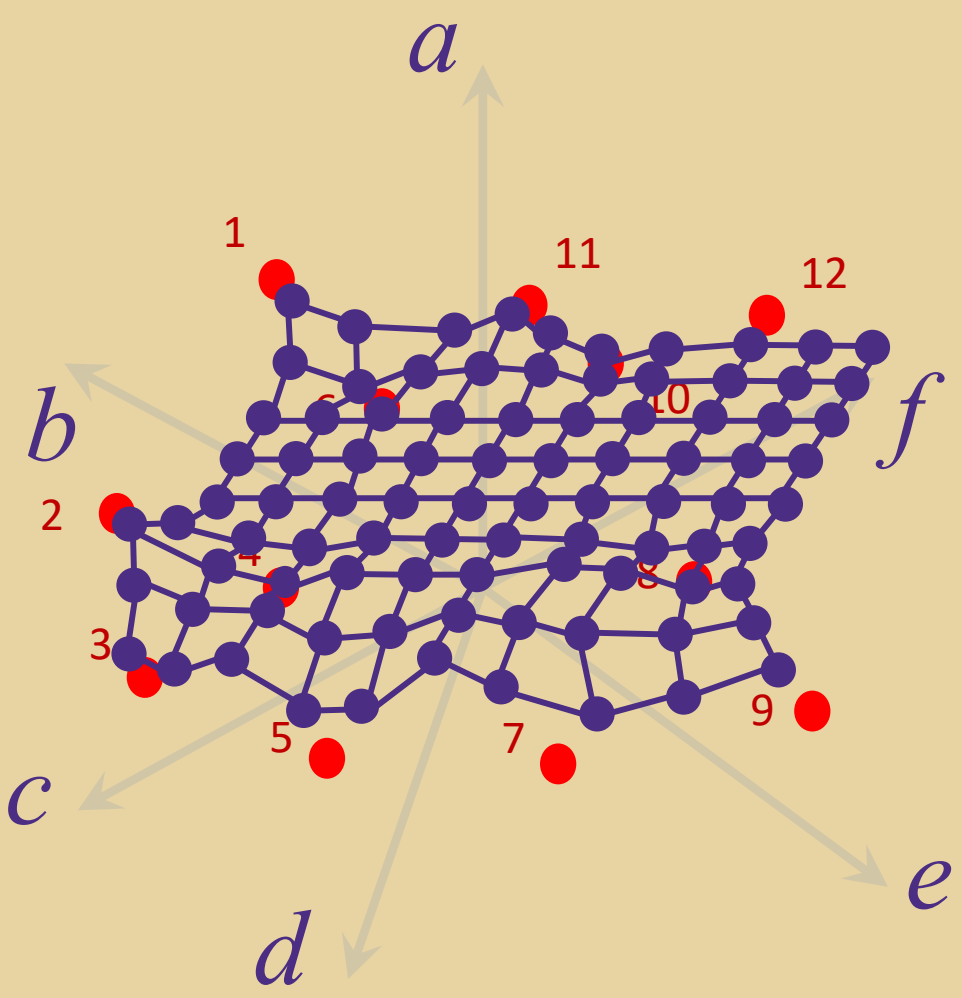

a
b
c
d
e
f
1
2
3
4
5
6
7
8
9
10
11
12

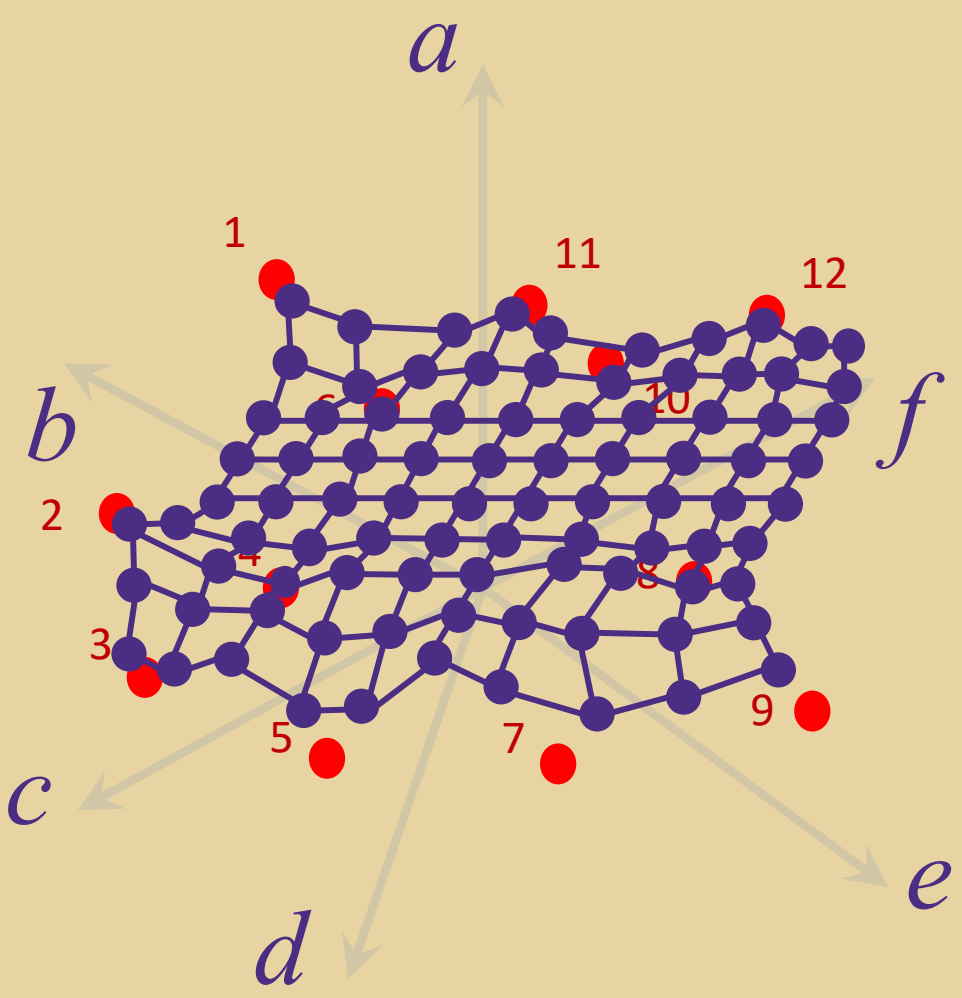

a
b
c
d
e
f
1
2
3
4
5
6
7
8
9
10
11
12


a
b
c
d
e
f
1
2
3
4
5
6
7
8
9
10
11
12


a
b
c
d
e
f
1
2
3
4
5
6
7
8
9
10
11
12

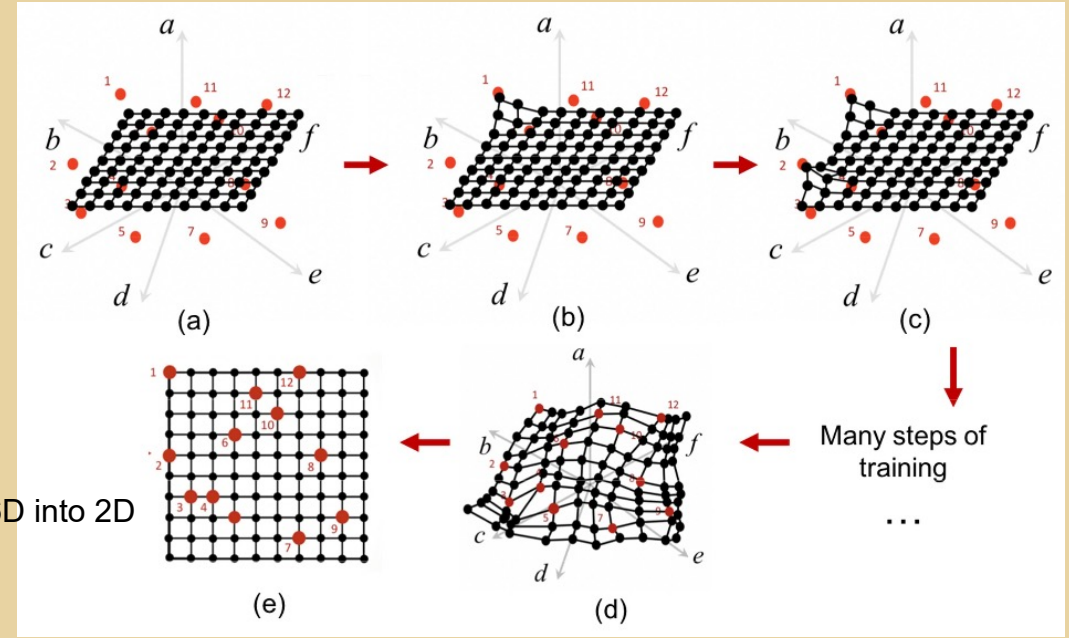

a
b
c
d
e
f
1
2
3
5
7
9
10
11
12


a
b
c
d
e
f
1
2
3
4
5
6
7
8
9
10
11
12


a
b
c
d
e
f
1
2
3
5
7
9
10
11
12

Self-Organizing Map (SOM) Algorithm

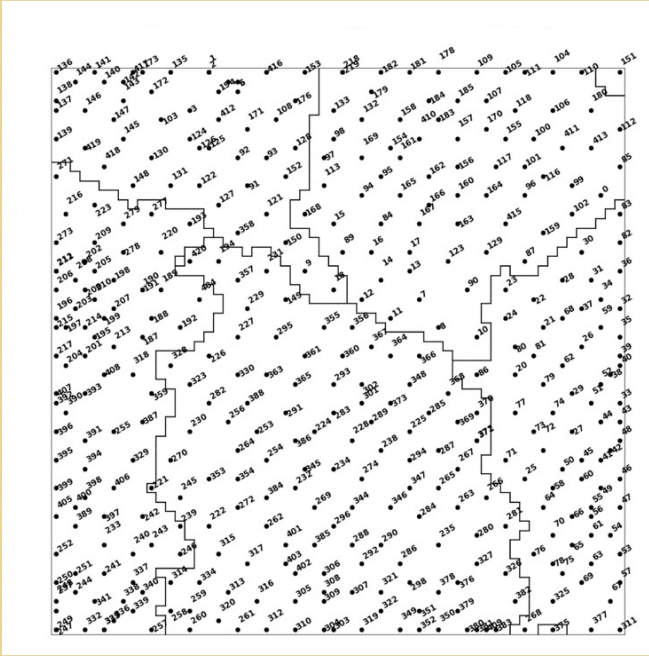
- **Dragging Nodes**
- **"Flattening a crumpled paper"**



represents 6D into 2D

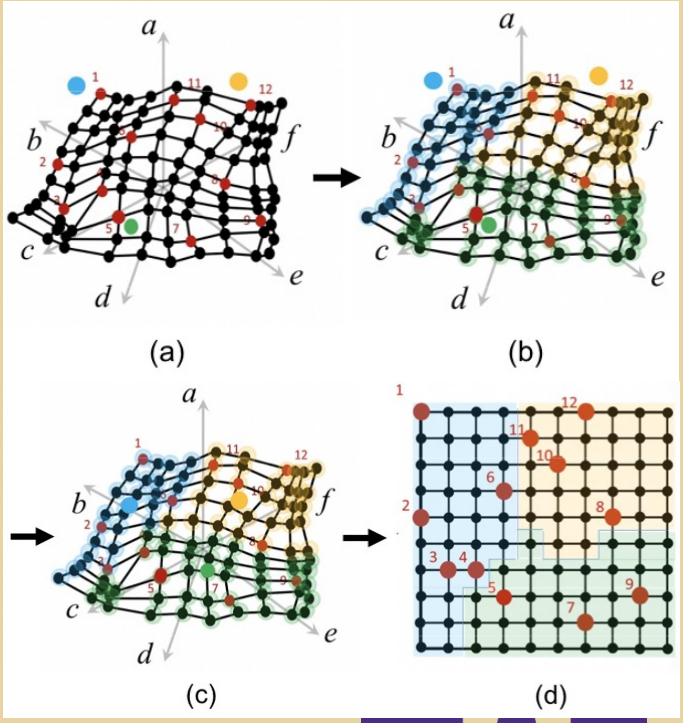
U-matrix and how to use it to get insights for clustering

- After training, the nodes in the 2D key map are not evenly distributed. The adjacent data point might not be similar to each other in the higher dimension space.
- U-matrix use the concept of the heatmap to illustrate the distance in Euclidean space



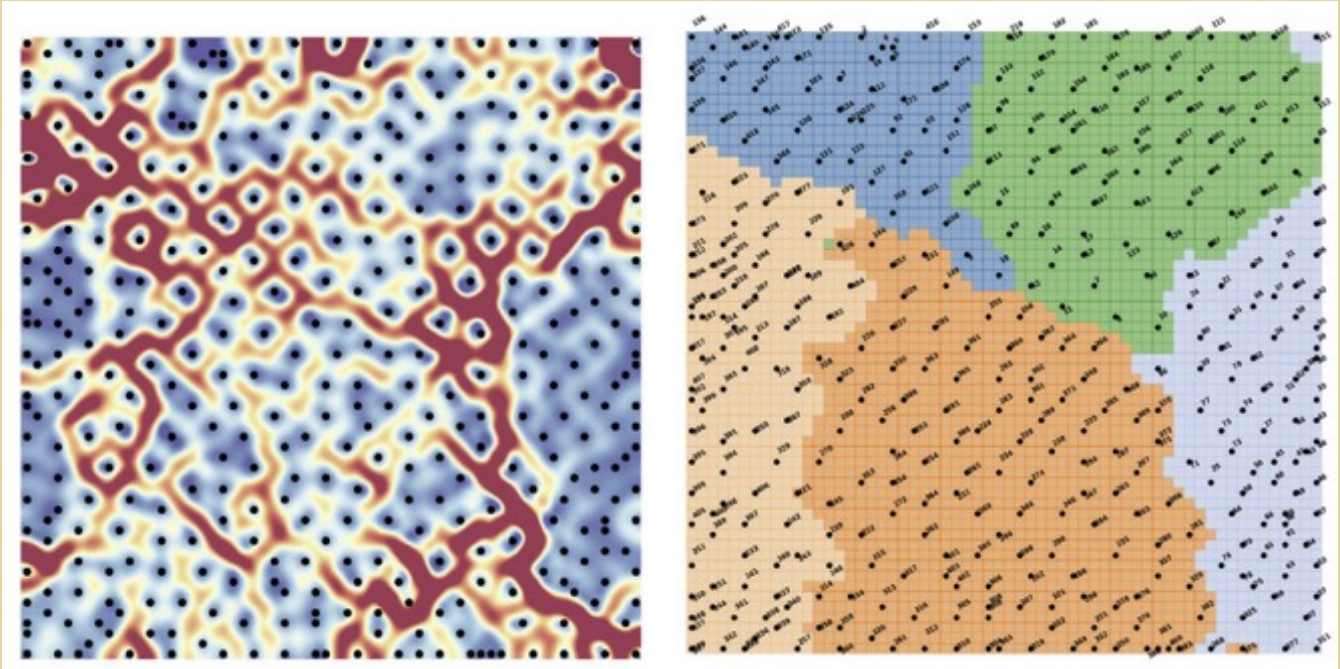
Using SOM in conjunction with other methods

- Since this is a dimensionality reduction method, for smaller dataset, you can initialize your SOM map using the first 2 Principal components, essentially the 2D PCA map
- K-means can also be run on the same dataset, and corresponding clusters can be visualized on SOM map.



K-Means clustering and U-Matrix

They can be compared to validate the results!



Blue: close to each other; Red: far from each other

- **SOM can provide a means to visualize K-Means!**
- **If the boundary matches well, then the training is successful**

Different Implementations of SOM

- SOM is just an algorithm, there are many packages you can use that implement it
- We will introduce
  - An augmented version of SOMPY, a version our group has contributions on
  - MiniSOM

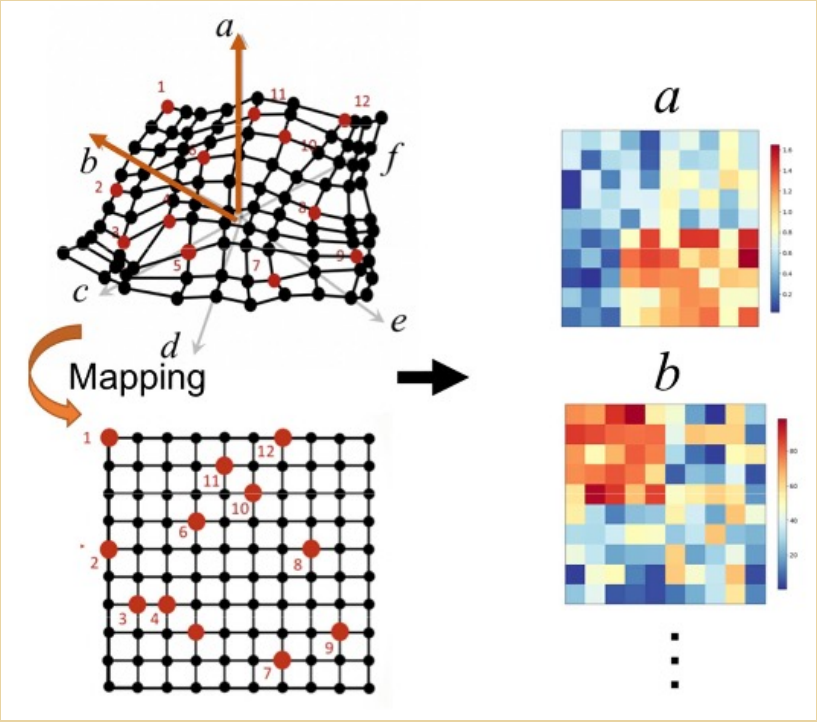
The uniqueness and functions of augmented SOMPY

https://github.com/DataScienceUWMSE/SOM

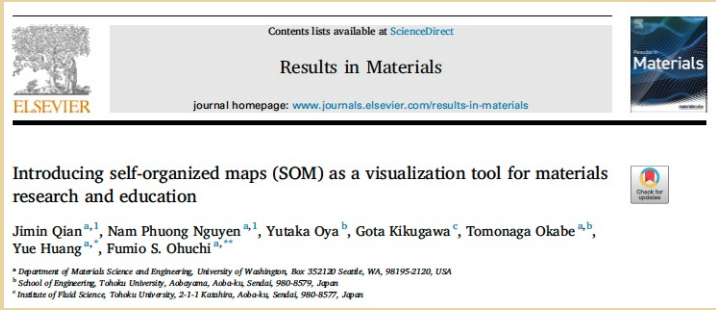
- **Utilizes PCA for initialization, and include K-Means Clustering overlay**
- **“Heat maps” provide a way to visualize each feature after training**
- **Projection function helps users find additional correlations or patterns among features, including for categorical data**

“heatmap” concept

- Map each node’s weight onto the 2D map
- Number of heat maps equals to number of input variables



Example of utilizing the heatmap on materials research


Contents lists available at ScienceDirect

Results in Materials

journal homepage: www.journals.elsevier.com/results-in-materials

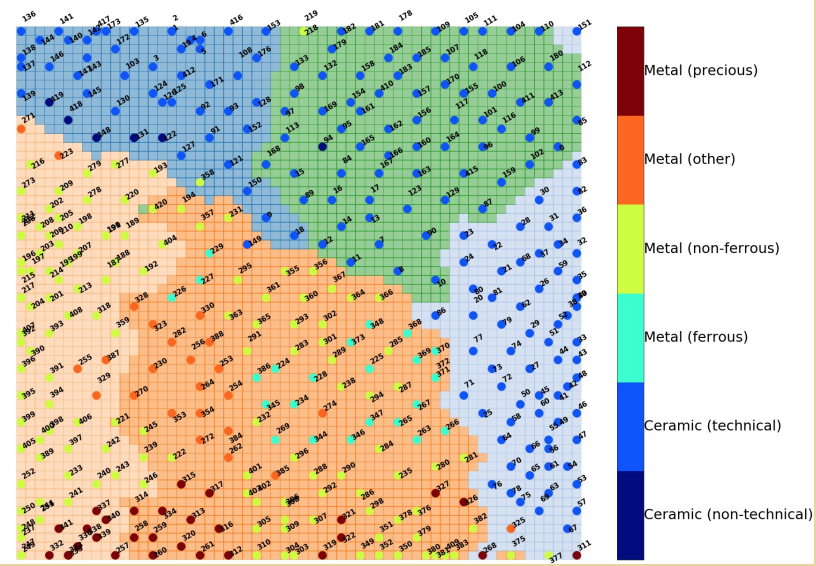
Introducing self-organized maps (SOM) as a visualization tool for materials research and education

Jimin Qian [a,1], Nam Phuong Nguyen [a,1], Yutaka Oya [b], Gota Kikugawa [c], Tomonaga Okabe [a,b], Yue Huang [a,*], Fumio S. Ohuchi [a,**]

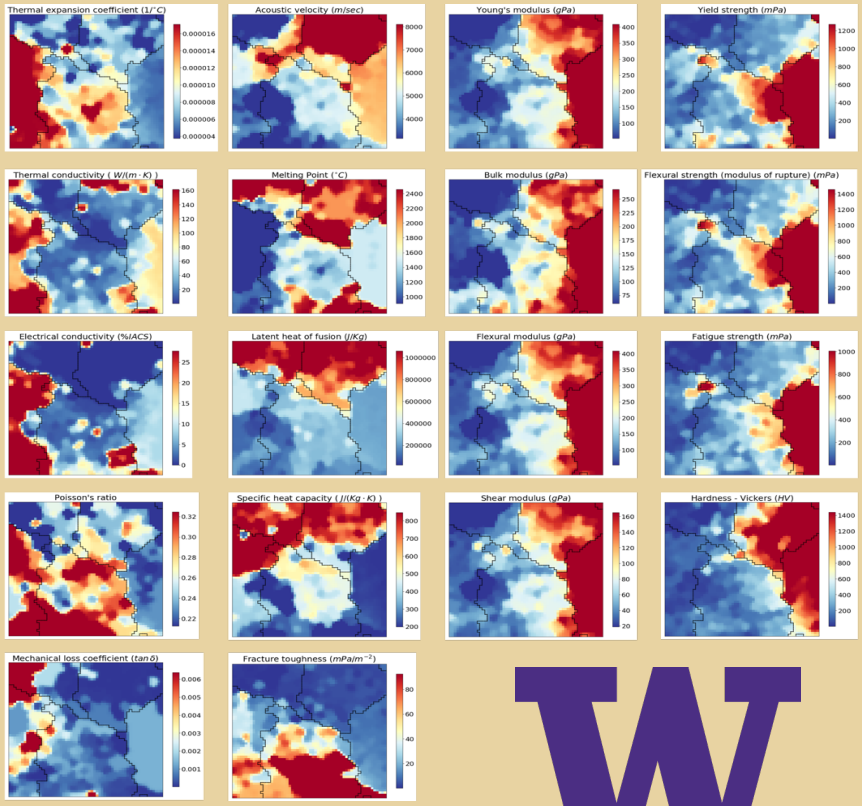
[a] Department of Materials Science and Engineering, University of Washington, Box 352120 Seattle, WA, 98195-2120, USA
[b] School of Engineering, Tohoku University, Aobayama, Aoba-ku, Sendai, 980-8579, Japan
[c] Institute of Fluid Science, Tohoku University, 2-1-1 Katahira, Aoba-ku, Sendai, 980-8577, Japan

Example 1 Granta Data Set: Experimental Commercial Materials Property Dataset

- **Training data set contains 398 commercial materials and 21 numerical properties**

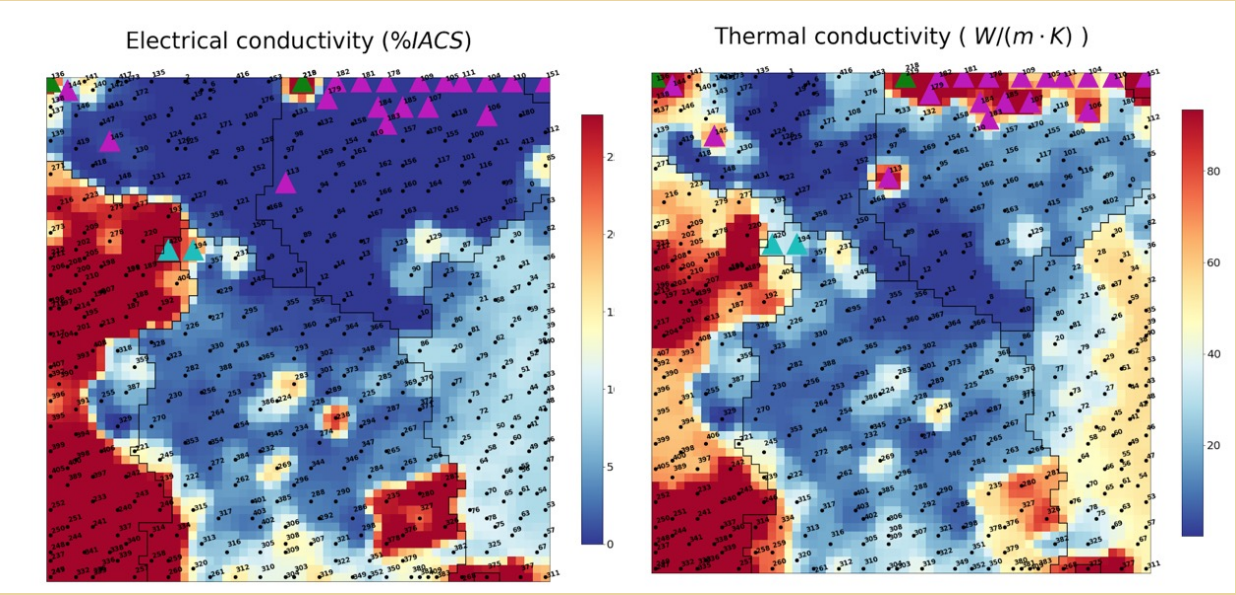


To find some pattern, is there any correlation between each properties?



Example of utilizing the heatmap on materials research

Example 1 Granta Data Set: Experimental Commercial Materials Property Dataset (continue)



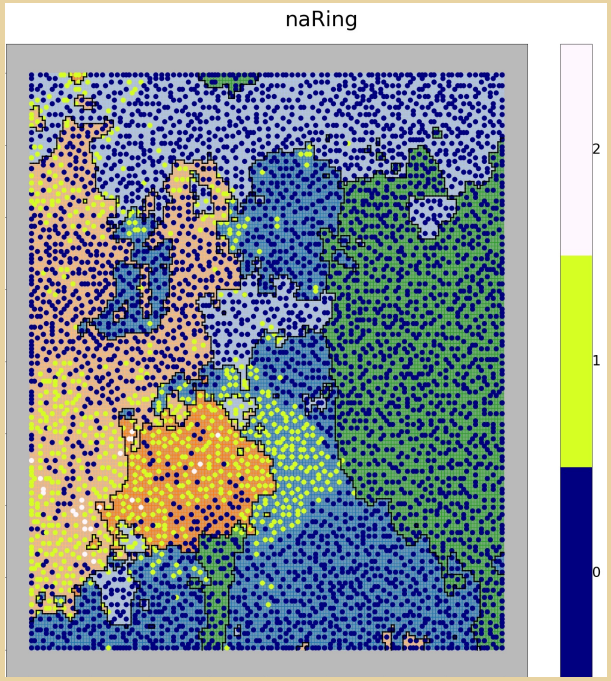
Group 1: Low electrical conductivity and High thermal conductivity

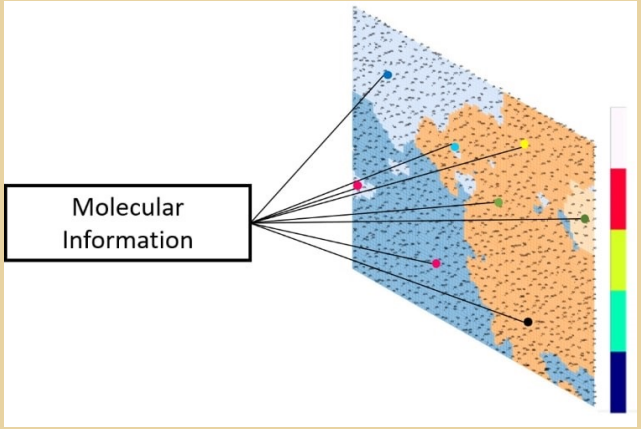
Group 2: High electrical conductivity and High thermal conductivity

Group 3: High electrical conductivity and Low thermal conductivity

Project information concept

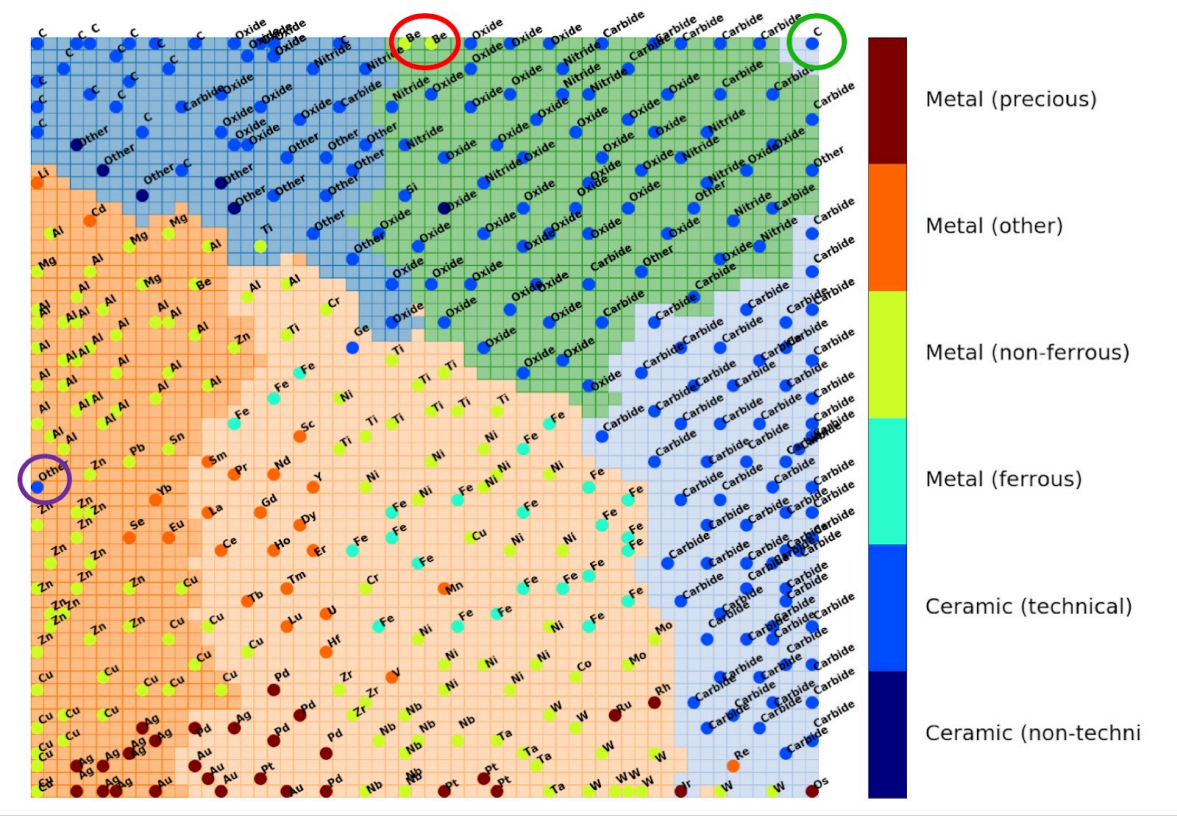
- Overlay one specific data property onto SOM, can use even categorical values
- Easily identify patterns





Example of utilizing the project function on materials research

Example 1 Granta Data Set: Experimental Commercial Materials Property Dataset (continue), finding the outliers' uniqueness

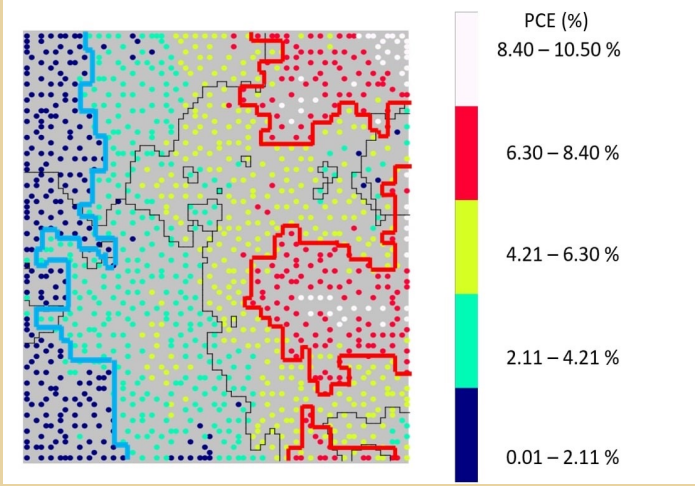


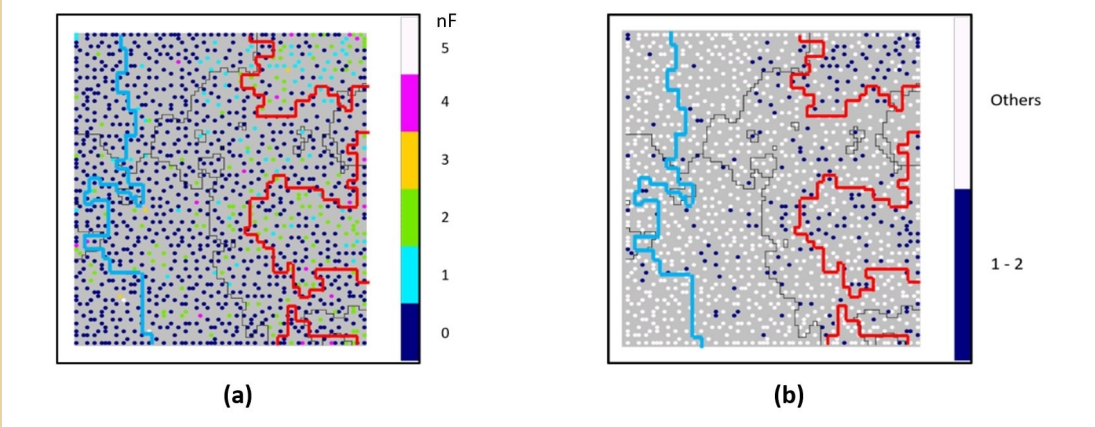
Example of utilizing the projection function on materials research

Example 2 OPV materials study using an experimental dataset

Reference Y.Huang, J. Phys. Chem. C 2020, 124, 12871−12882

> **Dataset includes 1203 donor polymers of Donor-Acceptor pairs, with properties related to the proficiency of the charge transfer.**





Molecular Descriptors

Python package of Molecular Descriptor

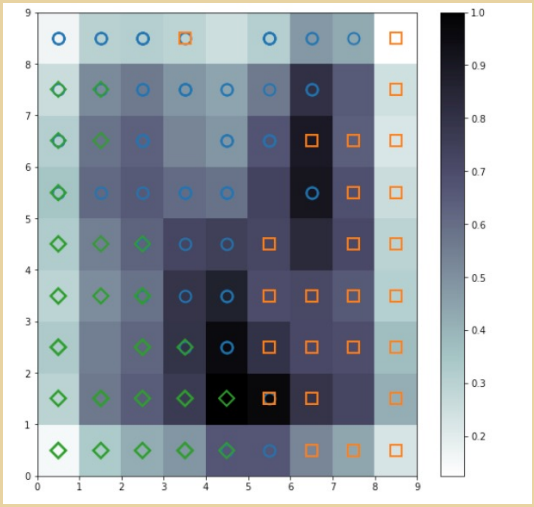
- **There are Python tools to extract molecular structural or geometrical information from notation of molecule, such as SMILES (Simplified molecular-input line-entry system)**
- **We will introduce Mordred, (covered in the Hands-on session)**

The advantage of using MiniSOM

- SOMPY is not as easy to use as the other packages introduced in this class.
  - The Augmented SOMPY has contribution from a few Materials Science researchers in our group, including your TA Jimin, Qian
- MiniSOM is relatively easier to use, well documented and constantly maintained, and have the basic implementation of the SOM algorithm

What MiniSOM provides

- It has :
  - The core implementation of SOM
  - Visualization
  - U-Matrix (“distance map” in MiniSOM)
  - Project certain feature onto SOM



- Doesn’t have:
  - PCA initialization
  - Cannot generate heatmap for each features
  - K-Means clustering,

Hyperparameters of SOM

- Length of input vectors (the number of properties)
- Map size, the most important one
- Map topology – rectangular or hexagonal
  - Important in defining the notion of "neighbors"
- Sigma – spread of the neighborhood function
- Learning Rate – initial learning rate, decreases with the number of iterations
- Decay function – defines how much learning rate and sigma decrease with the number of iterations
- Neighborhood function – defines how much neighbors of the BMU get impacted at each iteration (eg gaussian, bubble,...)
- Activation distance function (eg Euclidean distance)
- Initialization method – random or PCA

Hands-on session and HW for this week