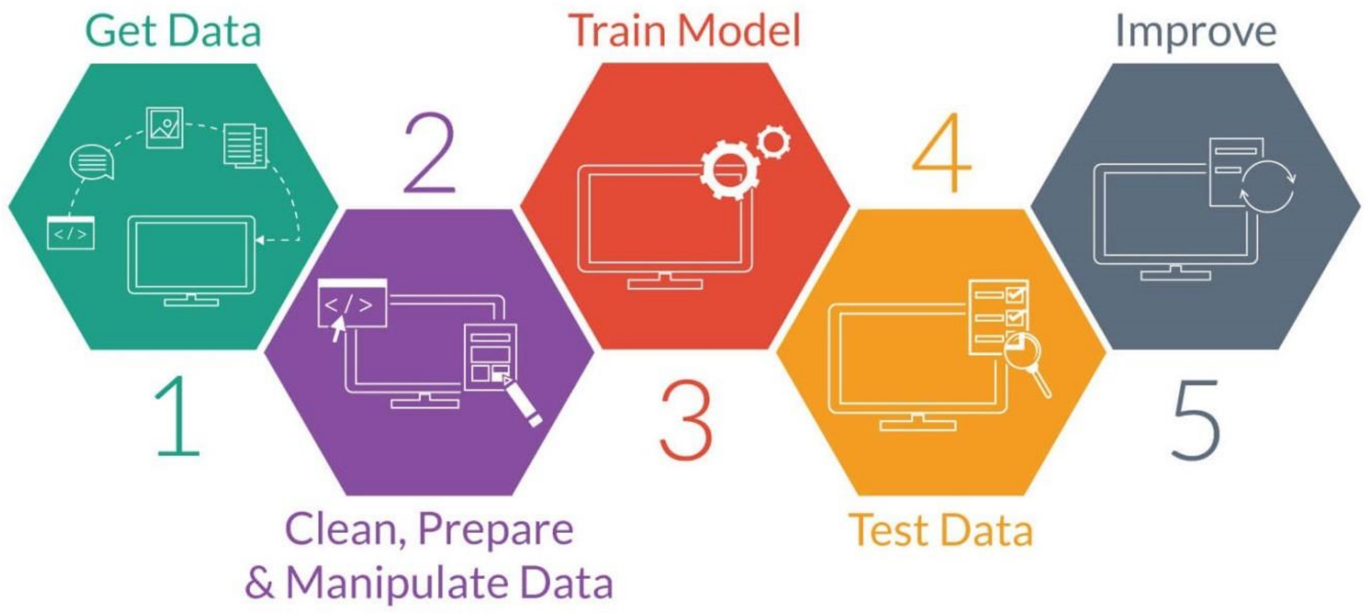
機器學習於材料資訊的應用
Machine Learning on Material Informatics

陳南佑(NAN-YOW CHEN)

nanyow@narlabs.org.tw

楊安正(AN-CHENG YANG)

acyang@narlabs.org.tw



使用軟體產生資料

檔案處理

特徵萃取

建立網路

分群演算法

用測試資料檢驗演算法

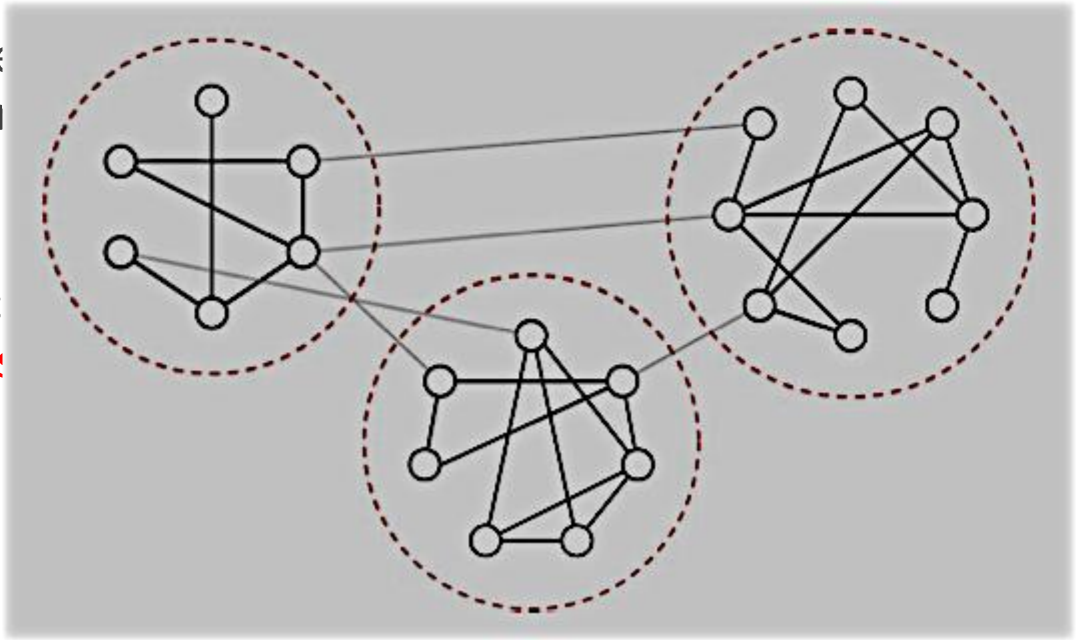
調整萃取特徵方法

分群演算法

- scikit-learning已提供多種現成的分群演算法可以使用。
  - K-means
  - Affinity Propagation
  - Hierarchical clustering
  - DBSCAN
- 雖然已經有多種分群演算法可以直接使用，但無可避免的是這些方法都還是需要人來挑選演算法參數，容易淪為先射箭再畫靶。
- 所以我們嘗試導入網路分析(network analysis)中的Modularity方法來進行微結構分類，最大的不同在於Modularity是非監督式學習，不需要人工決定分群的參數，比起其他方法要來的客觀。

Modularity of networks

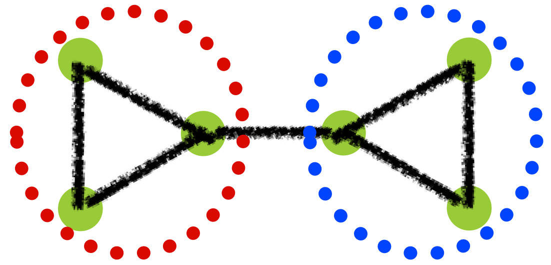
- Definition of a module: loosely linked island of densely connected nodes.
- Partitioning a ne are similar to each other and are as differ
- In order to desc ne a similarity measure and we also need a



Modularity of networks

- Definition of modularity:

$$Q = \sum_{s=1}^{N_M} \left[\frac{l_s}{L} - \left(\frac{d_s}{2L} \right)^2 \right]$$



where

- N_M: number of modules in the network
- l_s: number of intra-modular links in module s
- d_s: sum of the degrees of the nodes in module s
- L: total number of links in the network

N_M=2

l_1=3, l_2=3

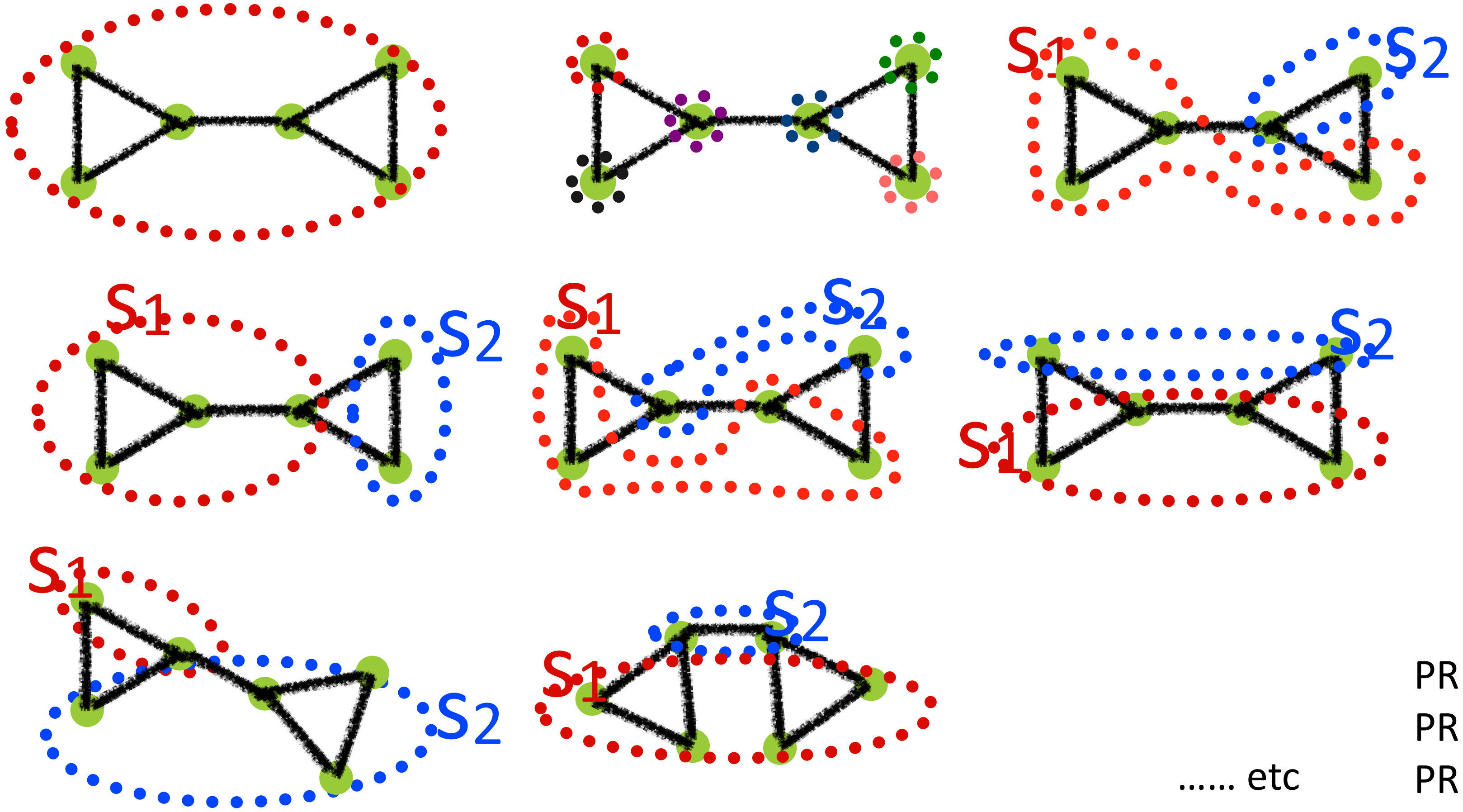
d_1=7, d_2=7

L=7

- Q = 0.357

Roger Guimerà, et al.: Nature **433**, 895 (2005)

Modularity of networks



...... etc

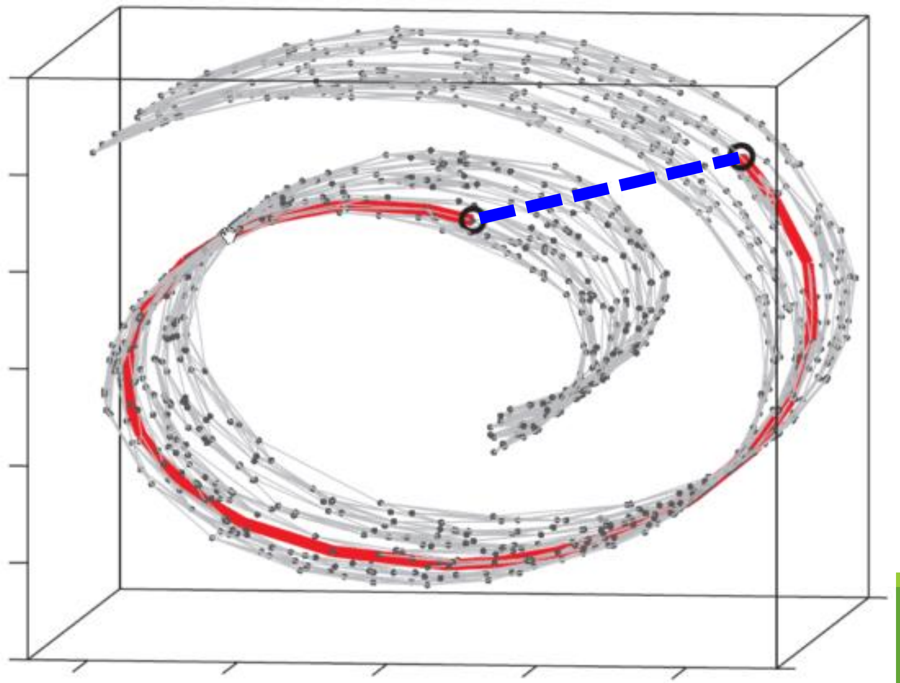
PRE **68**, 026121 (2003)
PRE **69**, 026113 (2004)
PRE **70**, 025101 (2004)

建立網路

- Modularity是基於網路分析的方法，所以比起其他分群演算法，需要多一個建立資料點的網路關係。
- 網路建立於資料點的資料空間。
- 距離的定義有許多種，歐式距離、曼哈頓距離、 Dijkstra distances …
- 網路連通的定義也要選擇。

Isomap

- Isomap is an extension of multi dimensional scaling (MDS), where pairwise euclidean distances between data points are replaced by geodesic distance on a high-dimensional manifold which is constructed by these data points.



For two arbitrary points (circled) on a nonlinear manifold, their Euclidean distance in the high dimensional input space (length of blue dashed line) may not accurately reflect their intrinsic similarity.

The red solid line is the geodesic distance (*i.e.* Dijkstra's distance) and the blue dashed line is the euclidean distance between two points, respectively.

Joshua B. Tenenbaum, et al.: Science **290**, 2319 (2000).

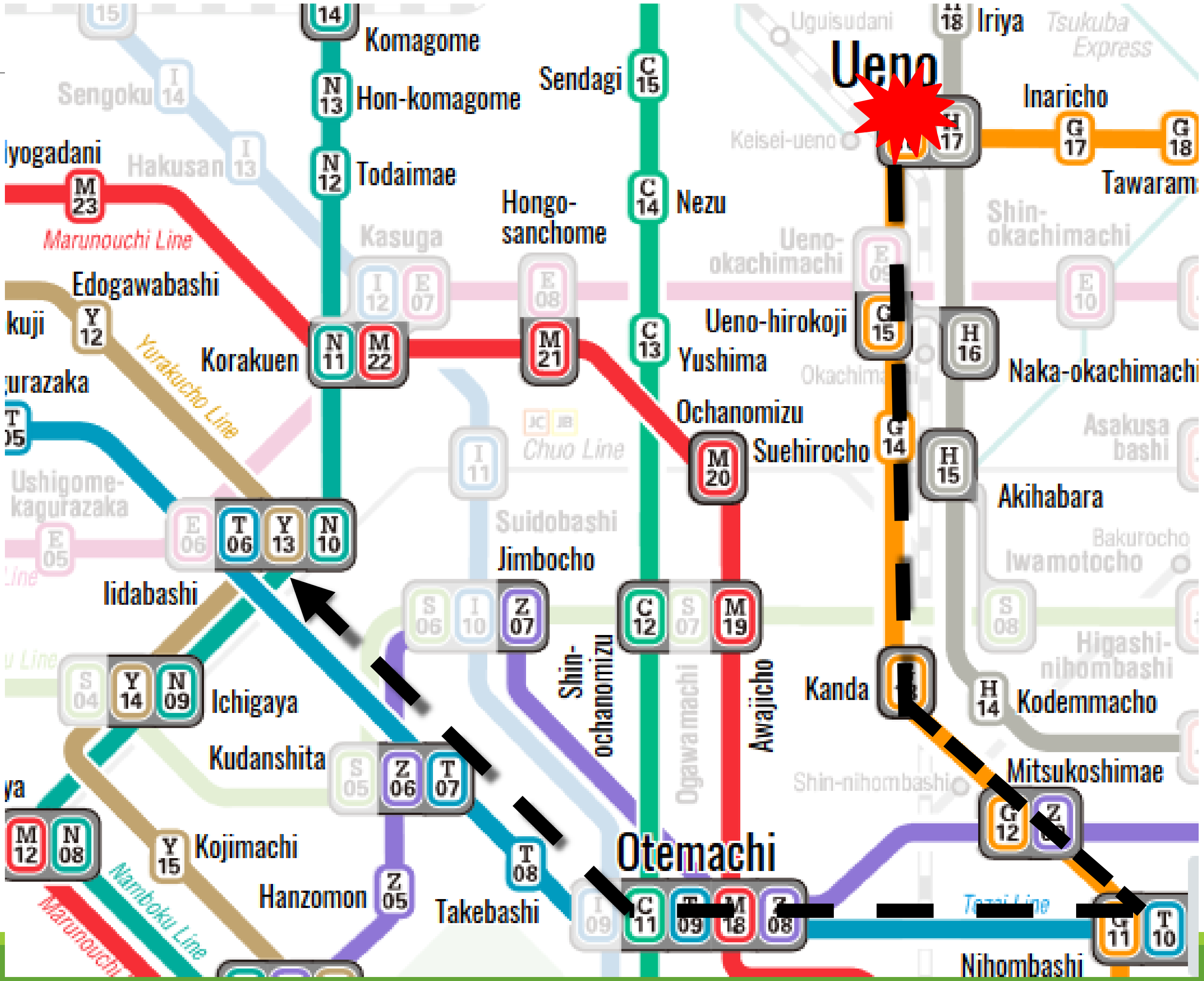
Dijkstra's algorithm

- Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. It was conceived by computer scientist Edsger W. Dijkstra in 1956.

Ex. Ueno and Iidabashi

- Euclidean distances : 3.2km
- Dijkstra distances : 3.4+3.7=7.1km

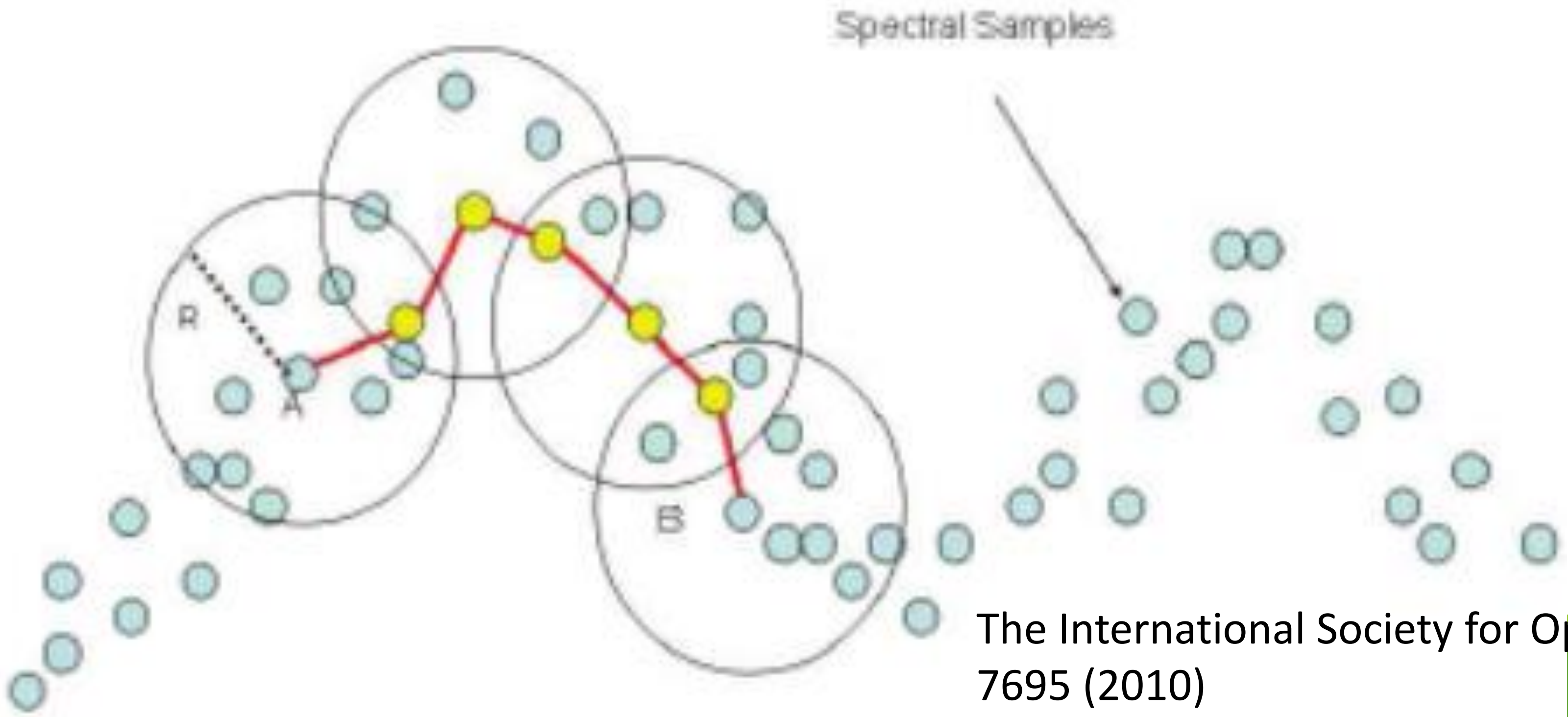
(Ueno → Nihonbashi→Iidabashi)



Construct Networks

Steps:

- Build graph with k-neighbors or ε-ball.
- Weight graph with euclidean distance.
- Compute pairwise geodesic distances by Dijkstra's algorithm.



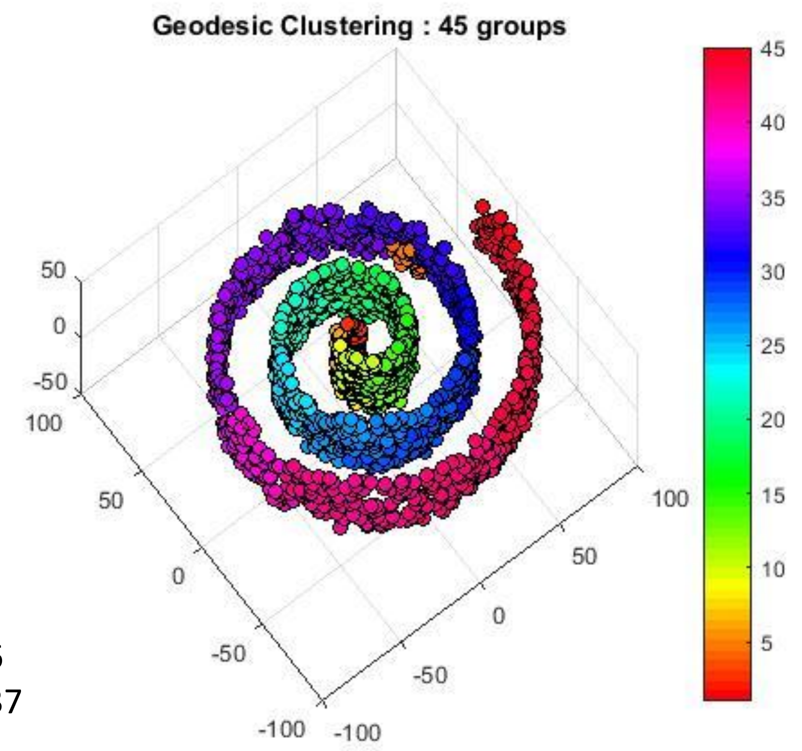
The International Society for Optical Engineering 7695 (2010)

Construct Networks ε-ball method

- Steps:

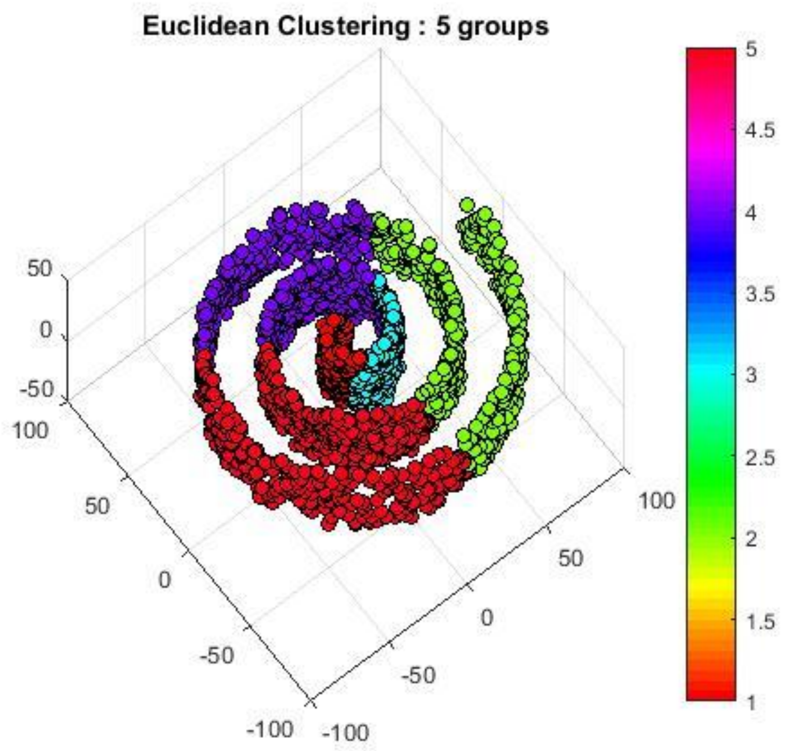
1. 算出資料點兩兩距離。
2. 將所有距離排序，並依序設為ε-ball的半徑。
3. 定義每個點與ε-ball內的點都有連接，以此建立graph。
4. 檢查graph是否為連通圖(Connected graph)，若為否，則挑選更大的距離作為ε-ball的半徑，重複3,4步驟。
5. 得到連通圖後再計算modularity。

Testing case - swiss roll manifold



Nodes = 2000
Edges = 6036
Ave. Degree = 6.036
Modularity Q = 0.937

(5-neighbors)

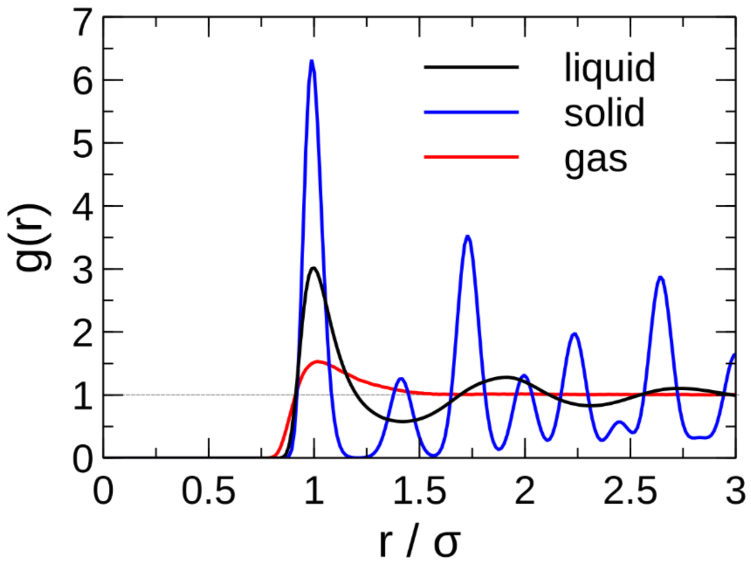


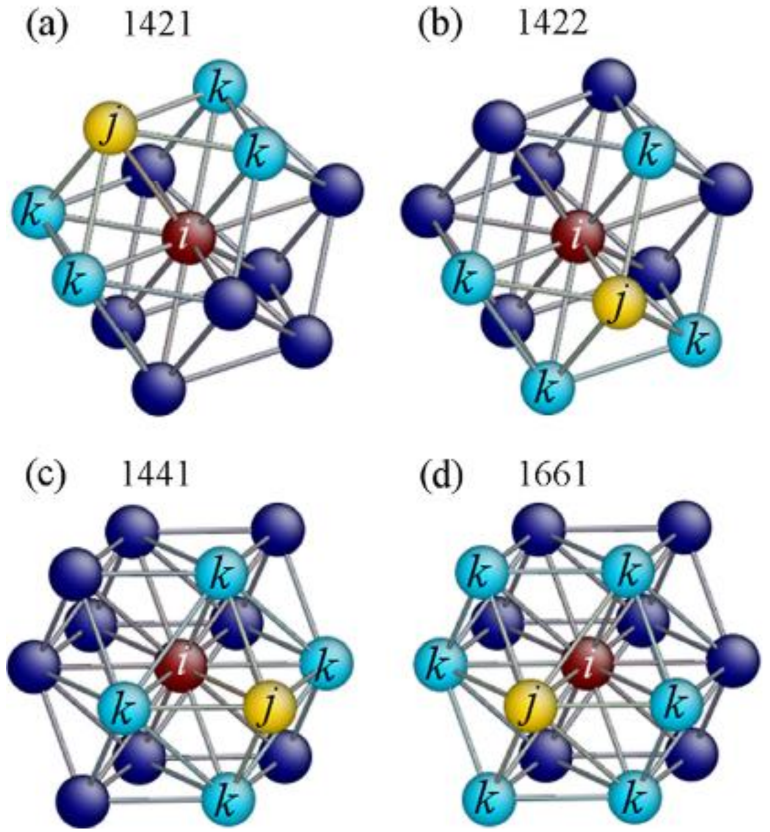
Nodes = 2000
Edges = 1999000
Ave. Degree = 1999
Modularity Q = 0.187

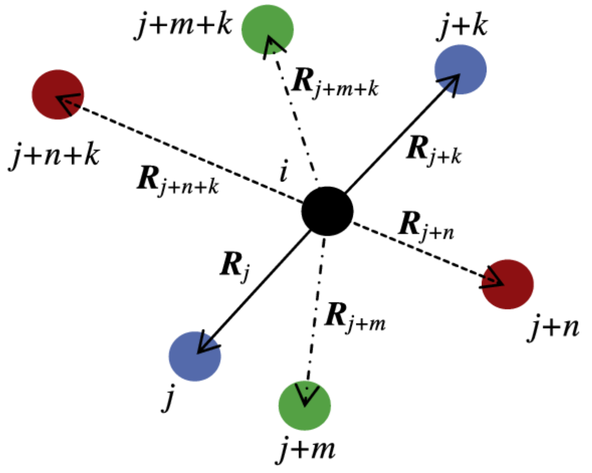
(1999-neighbors)

Classification of Crystallographic Groups by Machine Learning

A new powerful approach helping us to Identify the microstructure.





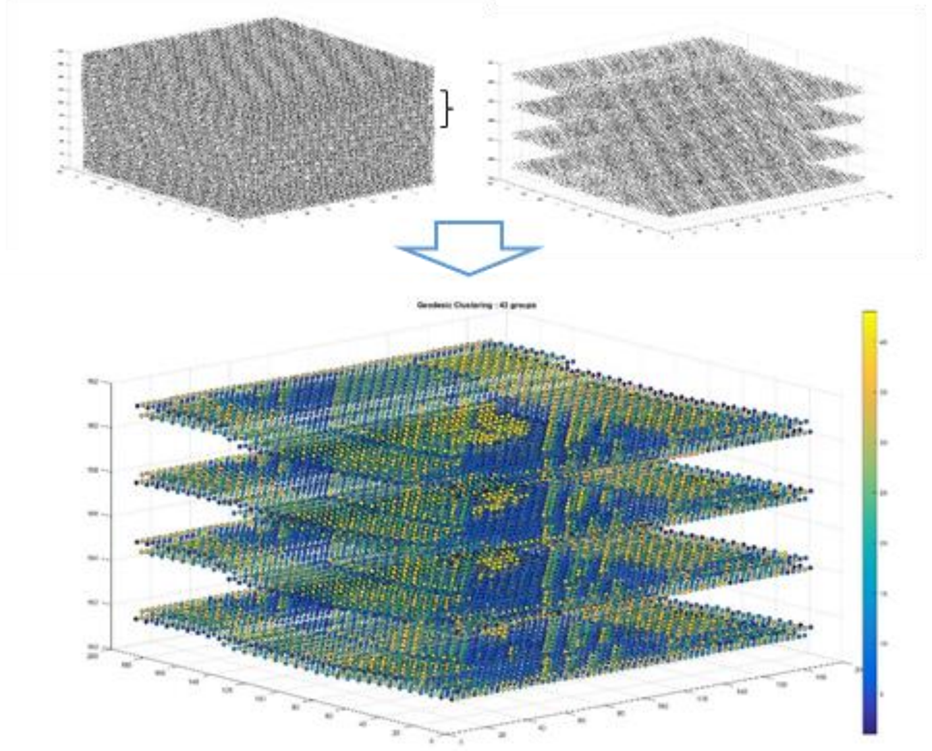


合作計畫	無
合作團隊	交大材料鄒年棣實驗室

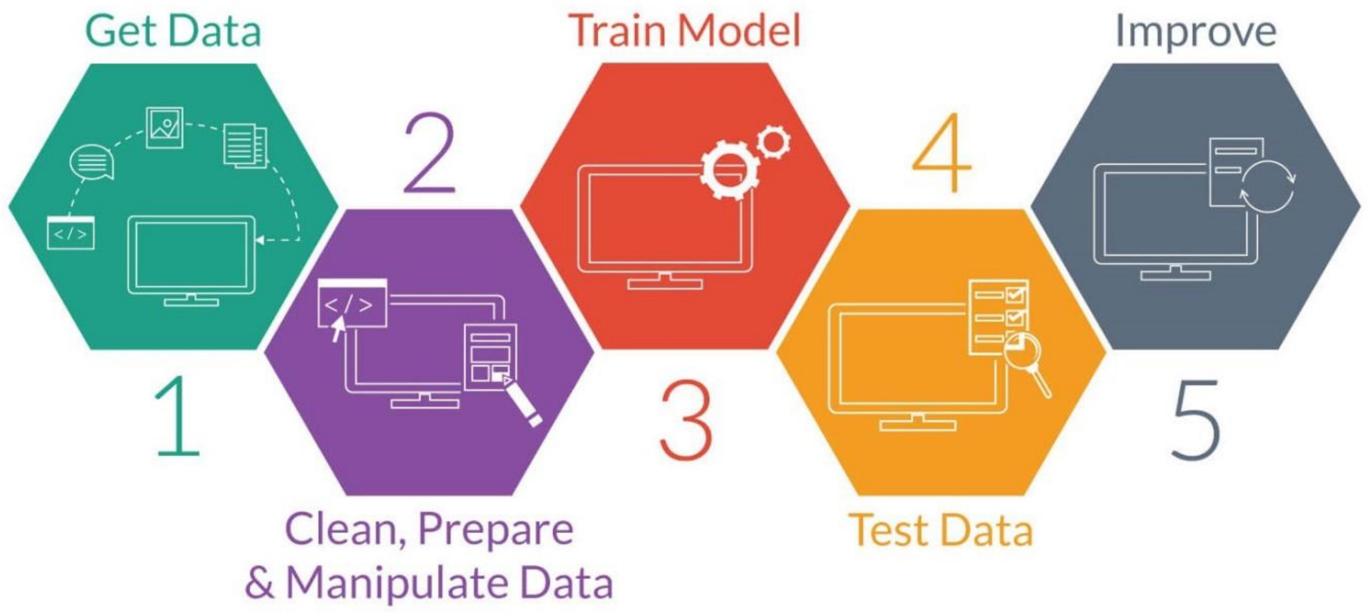
晶粒邊界辨識

晶粒邊界分群機器

Grain Boundary Detection



案例	透過非監督式學習，完成晶粒邊界辨識、結構分群
客戶目標	晶粒邊界辨識，達成結構分群，以進行區塊結構相似度分析，並與人為經驗公式相互驗證
問題困難描述	人工觀察大量模擬結果資料耗工費時，傳統分群方法需要先前知識才能協助分群。
訓練資料來源	分子動力學模擬結果
機器學習引擎	Modularity
結果	針對晶粒邊界辨識與其結構達成自動分群，與人工經驗法則結果一致。
Status	會議論文已發表 論文撰寫中



使用軟體產生資料

檔案處理

特徵萃取

建立網路

分群演算法

用測試資料檢驗演算法

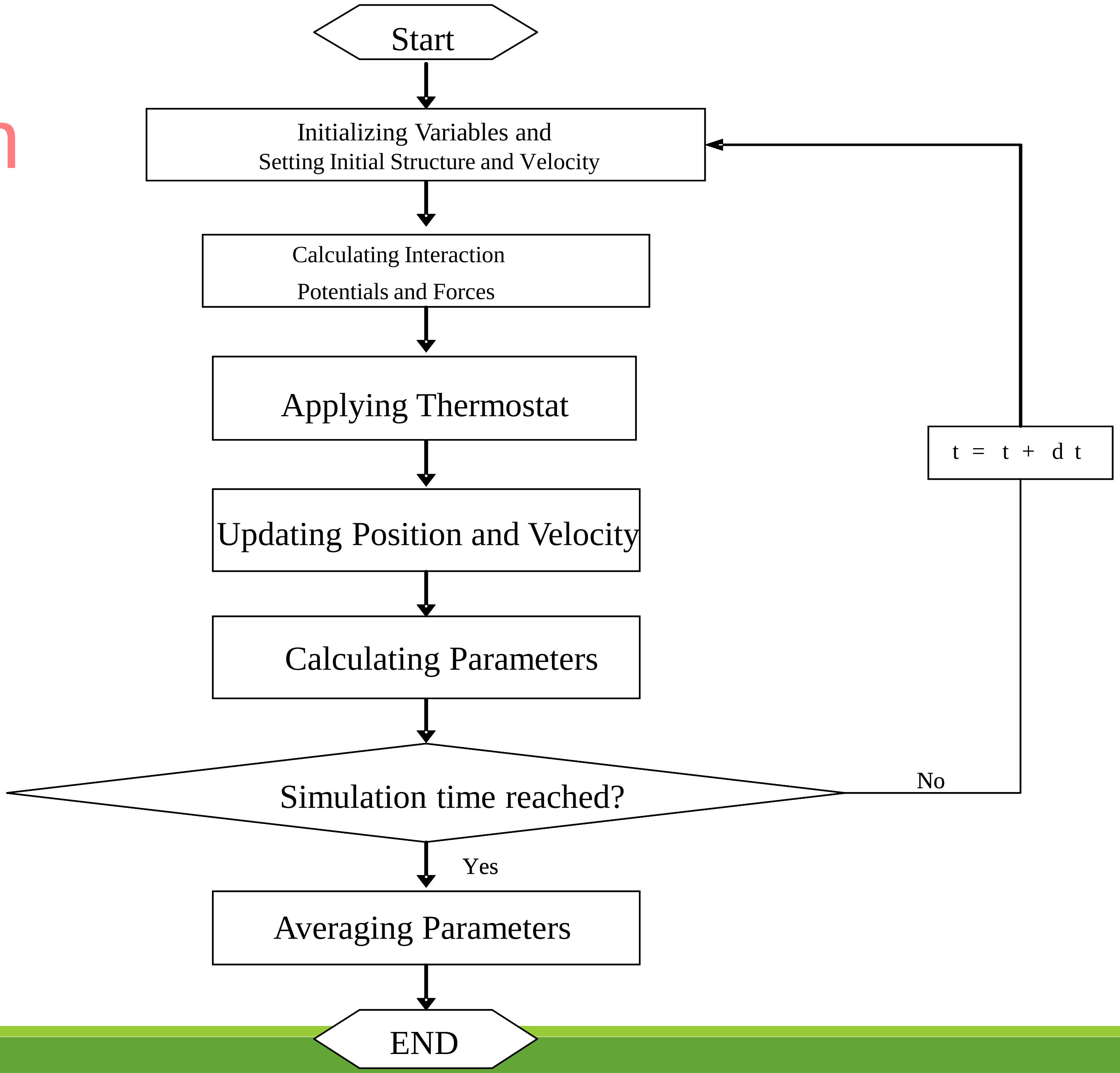
調整萃取特徵方法

Molecular Dynamics (MD)

- **A computer simulation technique that allows one to predict the time evolution of a system of interacting particles (atoms, molecules,.. etc.)**
- **N-body simulation**
- 利用一個描述原子間作用關係來描述一個系統。
- 系統內的所有原子遵循著牛頓運動定律。

Quantum Molecular Dynamics; Classical Molecular Dynamics

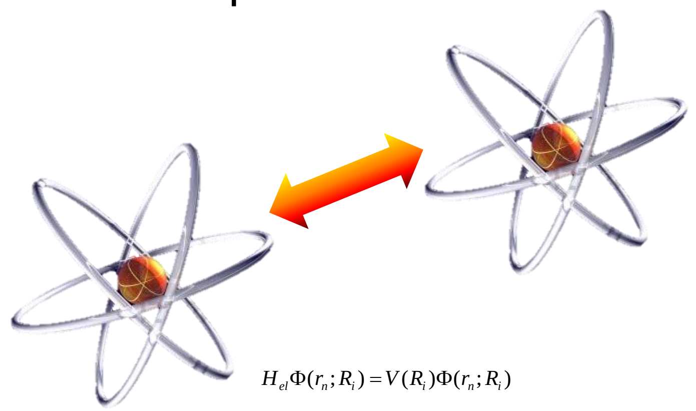
Flow Chart of MD simulation


Start
Initializing Variables and Setting Initial Structure and Velocity
Calculating Interaction Potentials and Forces
Applying Thermostat
Updating Position and Velocity
Calculating Parameters
Simulation time reached?
No
t = t + d t
Yes
Averaging Parameters
END

Equation of motion

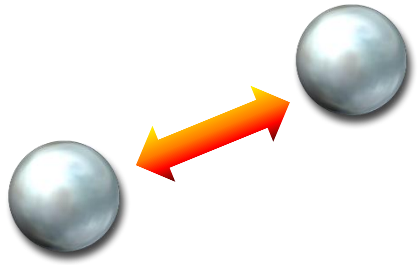
How Does Molecular Dynamics Work?

In the molecular dynamics method, each atom is treated as a point mass in space

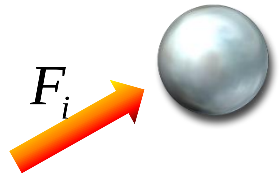


Born and Oppenheimer $\sqrt{\frac{M}{m}} \approx 100$

Simplify



Once the force on each atom is computed, atomic motion is determined through application of Newton's Laws



$$F_i(t) = m_i \frac{d^2 r_i(t)}{dt^2} = -\frac{\partial \Phi(r_i)}{\partial r_i}$$

Second-order ordinary differential equation which can be numerically integrated to find new atomic positions!

Packages for molecular modeling

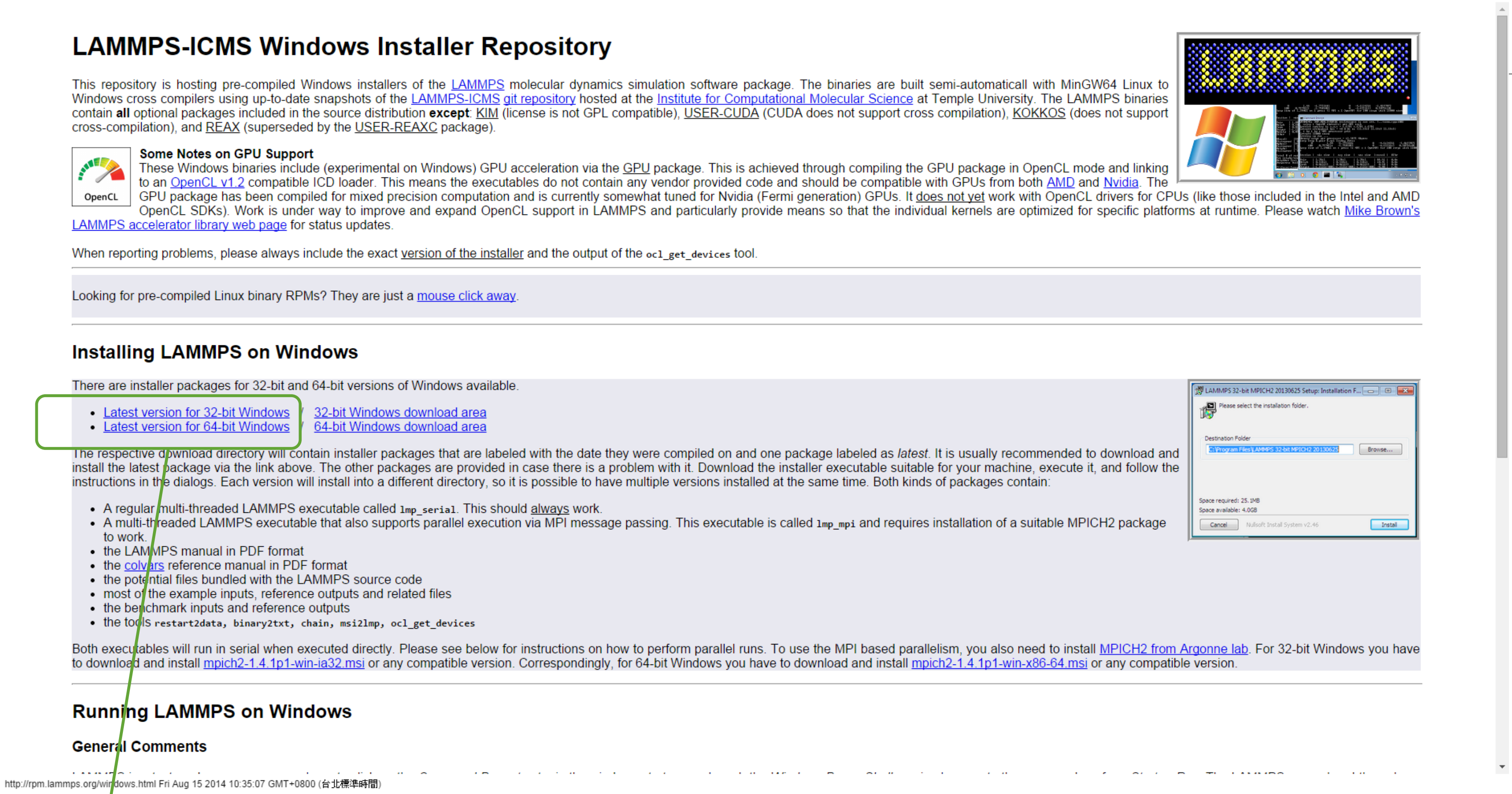
Package Name	Developer	Features
LAMMPS	Sandia National Labs	atomic, polymeric, biological, metallic, granular, and coarse-grained systems
NAMD	Parallel Programming Labs	structural biology
GROMACS	Science for Life Labs	general purpose
DL_POLY	Daresbury Labs	general purpose
Desmond	D. E. Shaw Research	biological and chemical systems
AMBER	Peter Kollman et. Al.	biomolecules
SPaSM	Los Alamos National Labs	Large scale simulation

Installation of Windows pre-built binary (ICMS version)

LAMMPS Windows Installer Repository

https://packages.lammps.org/windows.html

Webpage Screenshot



LAMMPS-ICMS Windows Installer Repository

This repository is hosting pre-compiled Windows installers of the LAMMPS molecular dynamics simulation software package. The binaries are built semi-automaticall with MinGW64 Linux to Windows cross compilers using up-to-date snapshots of the LAMMPS-ICMS git repository hosted at the Institute for Computational Molecular Science at Temple University. The LAMMPS binaries contain **all** optional packages included in the source distribution **except**: KIM (license is not GPL compatible), USER-CUDA (CUDA does not support cross compilation), KOKKOS (does not support cross-compilation), and REAX (superseded by the USER-REAXC package).

Some Notes on GPU Support
These Windows binaries include (experimental on Windows) GPU acceleration via the GPU package. This is achieved through compiling the GPU package in OpenCL mode and linking to an OpenCL v1.2 compatible ICD loader. This means the executables do not contain any vendor provided code and should be compatible with GPUs from both AMD and Nvidia. The GPU package has been compiled for mixed precision computation and is currently somewhat tuned for Nvidia (Fermi generation) GPUs. It does not yet work with OpenCL drivers for CPUs (like those included in the Intel and AMD OpenCL SDKs). Work is under way to improve and expand OpenCL support in LAMMPS and particularly provide means so that the individual kernels are optimized for specific platforms at runtime. Please watch Mike Brown's LAMMPS accelerator library web page for status updates.

When reporting problems, please always include the exact version of the installer and the output of the `ocl_get_devices` tool.

Looking for pre-compiled Linux binary RPMs? They are just a mouse click away.

Installing LAMMPS on Windows

There are installer packages for 32-bit and 64-bit versions of Windows available.

- Latest version for 32-bit Windows / 32-bit Windows download area
- Latest version for 64-bit Windows / 64-bit Windows download area

The respective download directory will contain installer packages that are labeled with the date they were compiled on and one package labeled as *latest*. It is usually recommended to download and install the latest package via the link above. The other packages are provided in case there is a problem with it. Download the installer executable suitable for your machine, execute it, and follow the instructions in the dialogs. Each version will install into a different directory, so it is possible to have multiple versions installed at the same time. Both kinds of packages contain:

- A regular multi-threaded LAMMPS executable called `lmp_serial`. This should always work.
- A multi-threaded LAMMPS executable that also supports parallel execution via MPI message passing. This executable is called `lmp_mpi` and requires installation of a suitable MPICH2 package to work.
- the LAMMPS manual in PDF format
- the colvars reference manual in PDF format
- the potential files bundled with the LAMMPS source code
- most of the example inputs, reference outputs and related files
- the benchmark inputs and reference outputs
- the tools `restart2data, binary2txt, chain, msi2lmp, ocl_get_devices`

Both executables will run in serial when executed directly. Please see below for instructions on how to perform parallel runs. To use the MPI based parallelism, you also need to install MPICH2 from Argonne lab. For 32-bit Windows you have to download and install mpich2-1.4.1p1-win-ia32.msi or any compatible version. Correspondingly, for 64-bit Windows you have to download and install mpich2-1.4.1p1-win-x86-64.msi or any compatible version.

Running LAMMPS on Windows

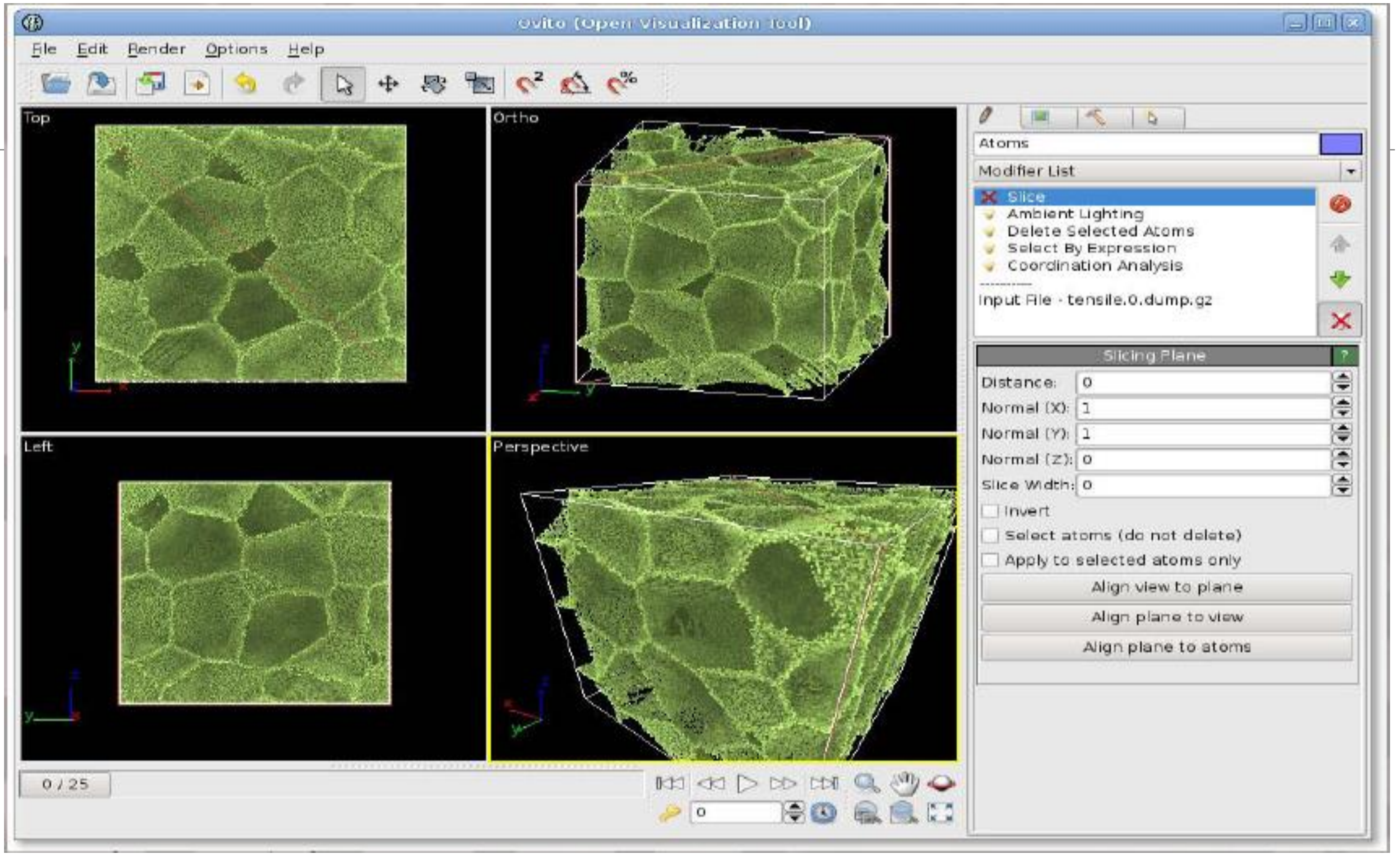
General Comments

http://rpm.lammps.org/windows.html Fri Aug 15 2014 10:35:07 GMT+0800 (台北標準時間)

Choose executable file based on the architecture of your PC, then double click on it to install.
If you want to run LAMMPS in parallel mode, mpich2-1.4.1p1-win-x86-64.msi is also needed.

Ovito

http://www.ovito.org/


Ovito (Open Visualization Tool)
File Edit Render Options Help
Top
Ortho
Left
Perspective
0 / 25

Atoms

Modifier List

- Slice
- Ambient Lighting
- Delete Selected Atoms
- Select By Expression
- Coordination Analysis

Input File - tensile.0.dump.gz

Slicing Plane

Distance: 0
Normal (X): 1
Normal (Y): 1
Normal (Z): 0
Slice Width: 0

☐ Invert
☐ Select atoms (do not delete)
☐ Apply to selected atoms only

Align view to plane
Align plane to view
Align plane to atoms

- Feature : A fast analyzing configuration viewer for crystal structures ➔ FCC BCC HCP ….
- Format : xyz, cfg

使用軟體產生資料
0_data preparsion

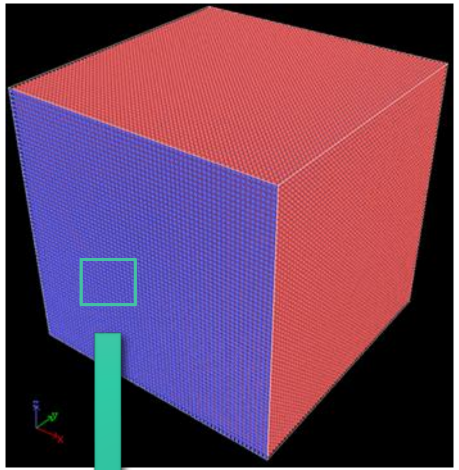
- LAMMPS執行檔C:\Program Files\LAMMPS 64-bit 16Aug2018\bin\lmp_serial.exe
- LAMMPS模擬設定檔案
  - in.Cu
  - in.Si
  - in.melt
- LAMMPS勢能函數檔案
  - Cu_u3.eam
  - Si.tersoff
  - LJ(沒有檔案)
- 執行指令run_serial.bat
  - "C:\Program Files\LAMMPS 64-bit 16Aug2018\bin\lmp_serial.exe" -in in.Cu
- 輸出檔案
  - 1_data/dump.*.cfg
  - 1_data/dump.*.data
- 視覺化軟體ovito

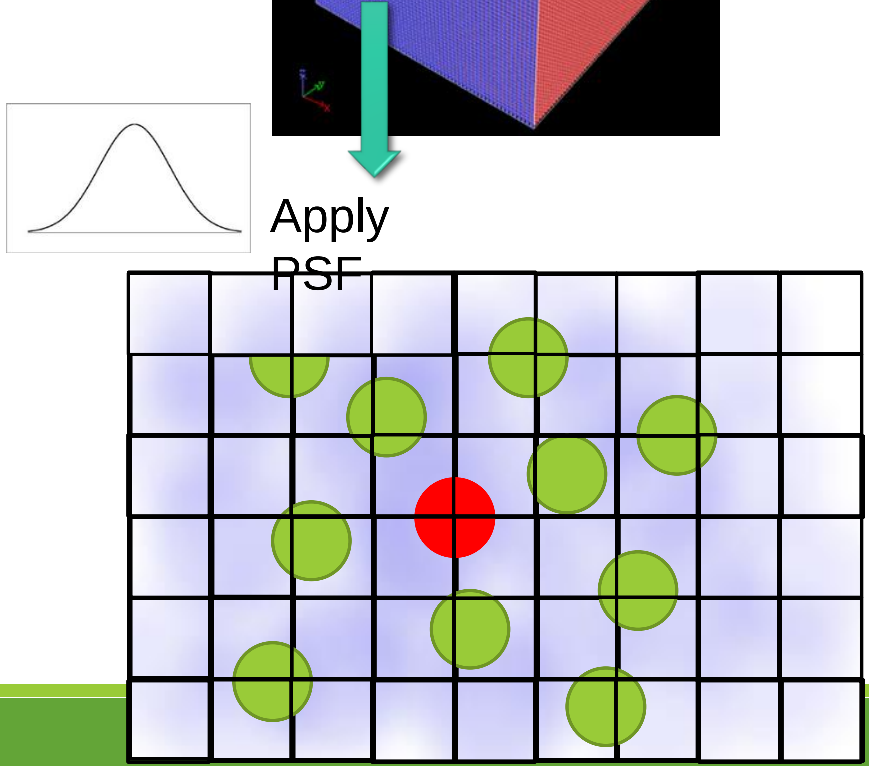
檔案處理&特徵萃取

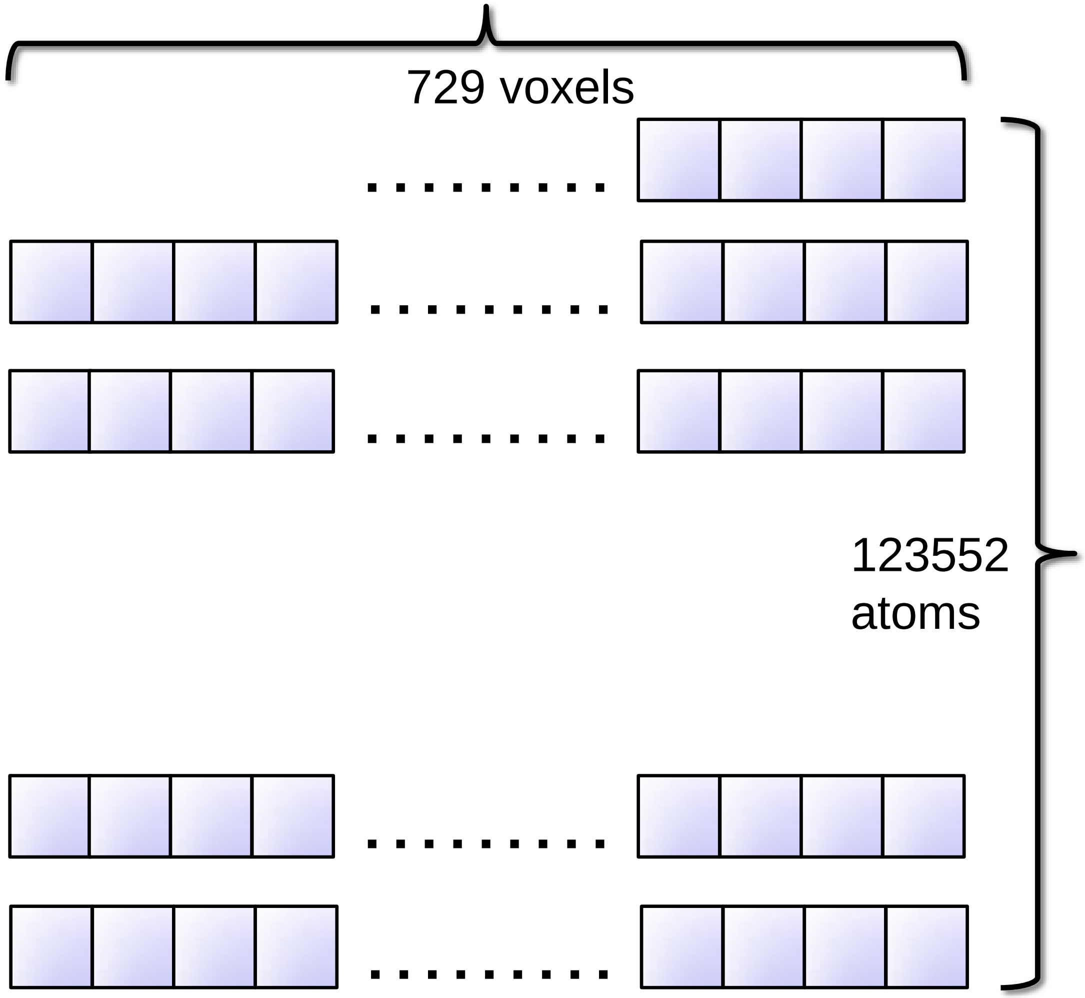
- LAMMPS可以輸出dumpfile(cfg)，xyz，trajectory file(dcd)，自己打造Parser的話不用特別考慮。
  - dumpfile(cfg):在模擬過程中標準輸出檔案，ascii檔，人可以直接讀和編輯。 (MDANALSIS不能讀，需要用ovito轉成data檔)
  - xyz:輸出另一種檔案形式，ascii檔，人可以直接讀和編輯。(MDANALSIS可以讀，但是缺了mass info。)
  - trajectory file:輸出另一種檔案形式，binary檔，人不能直接解讀。(MDANALSIS可以讀，ovito讀不了。)
- 特徵萃取:結構分類問題是局部的，考慮一個原子的特徵，需要從與周遭原子的關係下手。
  - Voxelize local region (This class)
  - Local environment
  - Nearest neighbor

Feature of local environment

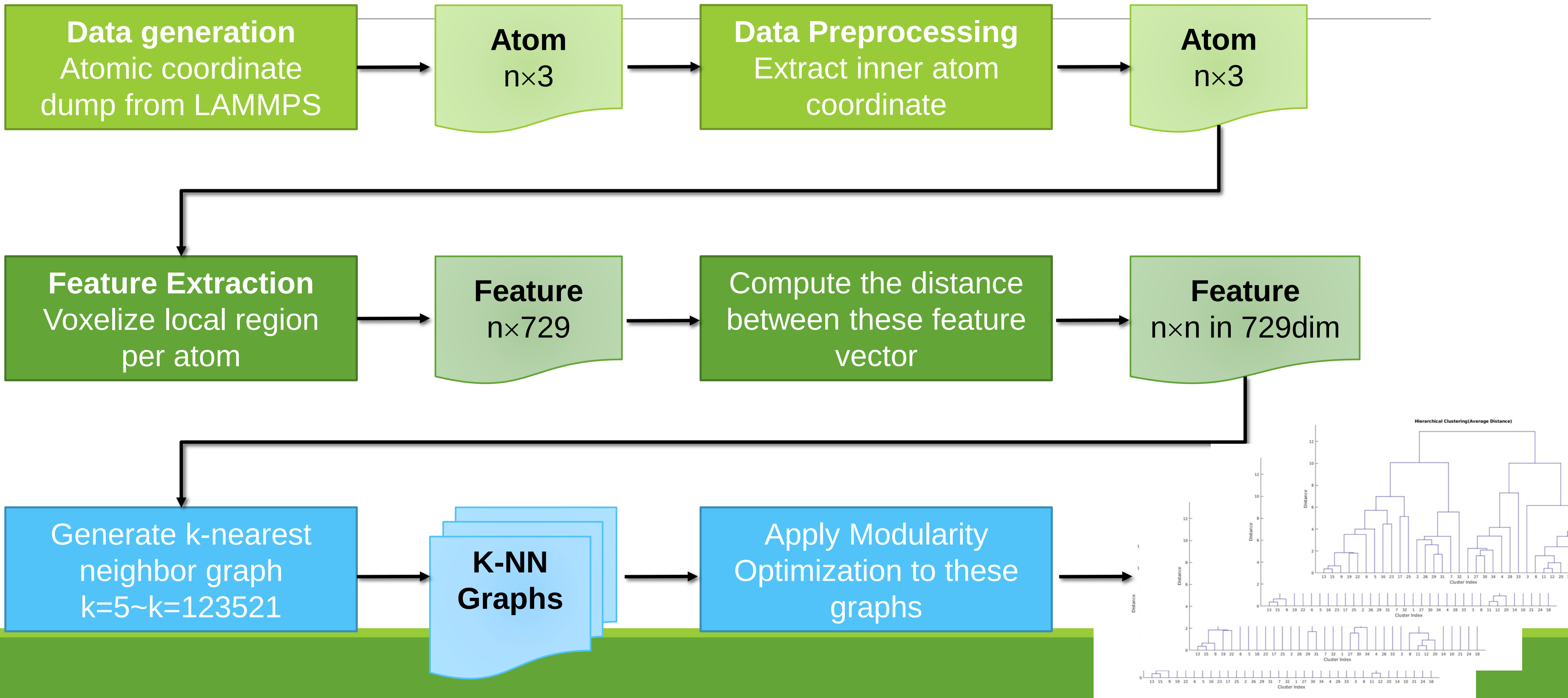
```
>python feature_engineering.py
```







Flowchart


Data generation
Atomic coordinate dump from LAMMPS
Atom
n×3
Data Preprocessing
Extract inner atom coordinate
Atom
n×3
Feature Extraction
Voxelize local region per atom
Feature
n×729
Compute the distance between these feature vector
Feature
n×n in 729dim
Generate k-nearest neighbor graph
k=5~k=123521
K-NN Graphs
Apply Modularity Optimization to these graphs
Hierarchical Clustering(Average Distance)
Distance
Cluster Index

More on descriptors

- Atom-centered Symmetry Functions (ACSF)
- Smooth Overlap of Atomic Positions (SOAP)
- Gaussian descriptor
- Behler type Symmetry function(目前最多人採用)
- …

建立網路

- Modularity是基於網路分析的方法，所以比起其他分群演算法，需要多一個建立資料點的網路關係。
- 網路式建立於資料點的空間，不是原本問題的卡式座標。
- 距離的定義有許多種，歐式距離、曼哈頓距離、 Dijkstra distances …
- 網路連通的定義也要選擇。

Construct Networks ε-ball method

- Steps:

1. 算出資料點兩兩距離。
2. 將所有距離排序，並依序設為ε-ball的半徑。
3. 定義每個點與ε-ball內的點都有連接，以此建立graph。
4. 檢查graph是否為連通圖(Connected graph)，若為否，則挑選更大的距離作為ε-ball的半徑，重複3,4步驟。
5. 得到連通圖後再計算modularity。

Construct Networks construct_network.ipynb

```
#載入資料點
file = 'features.csv'
features_space = pd.read_csv(file)
features_space.shape
```

(61, 730)

61顆原子

1+9*9*9=730 voxels

Construct Networks construct_network.ipynb

```
natom = features_space.shape[0]
ndim = features_space.shape[1]
#建立連線關係表作為networkx產生圖的輸入
outfile = 'network.csv'
with open(outfile, 'w') as output:
    writer = csv.writer(output, delimiter=' ')
```

Construct Networks construct_network.ipynb

```
for i in list(range(natom)):
        features_i = features_space.iloc[i, :].values
        for j in list(range(i+1,natom)):
            features_j = features_space.iloc[j, :].values
            dist=0.0
            for k in list(range(1,ndim)):
                dist += (features_i[k] - features_j[k])**2
            print( int(features_i[0]), int(features_j[0]), dist)
            writer.writerow( [int(features_i[0]), int(features_j[0]), dist] )
```

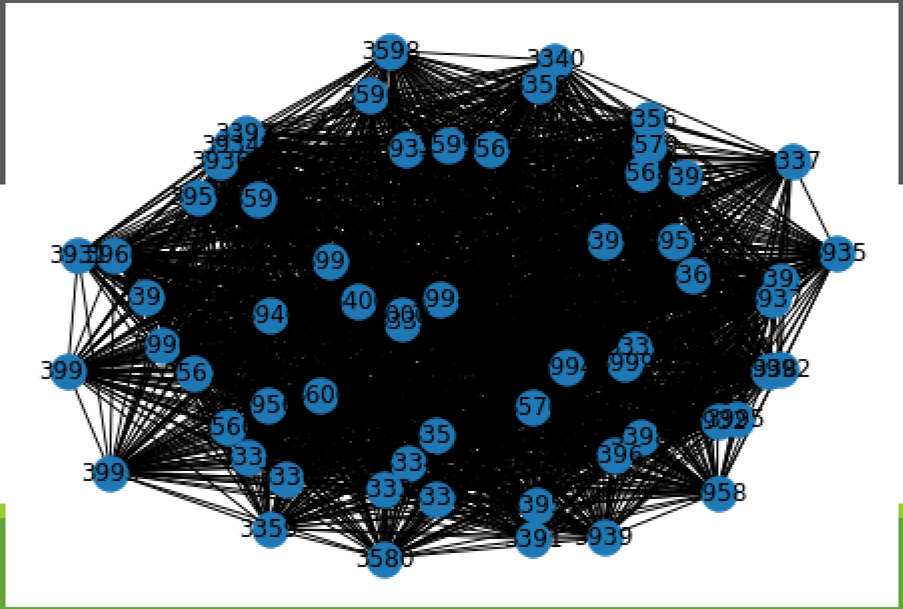
start point   end point   distance

Construct Networks construct_network.ipynb

```
import community
import networkx as nx
import matplotlib.pyplot as plt

#讀入先前建立的連線關係表
Edges = pd.read_csv('Edges.csv', sep='  ', header=None )

E = np.array(Edges.iloc[:, 0:3].values)
```

Construct Networks construct_network.ipynb

```
#Use pre-defined linkage (Edges.csv) to constructure whole network
G = nx.Graph()
for i in range(0, len(E)):
    e = ( str(int(E[i,0])), str(int(E[i,1])), E[i,2] )
    G.add_weighted_edges_from([(e)])

nx.draw(G, with_labels=True)
```



Calculate Modularity construct_network.ipynb

```
# https://python-louvain.readthedocs.io/en/latest/
partition = community.best_partition(G)
# community.best_partition回傳的物件是dict，利用.values取出各群的內容，。
# 利用set建立無序的資料集合，再用len取得這個集合的長度，這就是分群後的結果。
size = float(len(set(partition.values())))
print("community:", size)
# community.modularity就直接計算modularity，graph太大的話，就要考慮自己實做平行版本的modularity
mod = community.modularity(partition,G)
print("modularity:", mod)
```

Calculate Modularity construct_network.ipynb

```
# draw the graph based on the result of modularity
pos = nx.spring_layout(G)
# color the nodes according to their partition
cmap = matplotlib.cm.get_cmap('viridis', max(partition.values()) + 1)
nx.draw_networkx_nodes(G, pos, partition.keys(), node_size=40,
                       cmap=cmap, node_color=list(partition.values()))
nx.draw_networkx_edges(G, pos, alpha=0.5)
plt.show()
```