Machine Learning for Structural Analysis in FCC Crystal

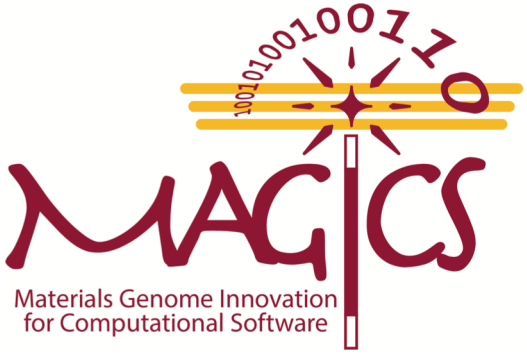
Pankaj Rajak

Argonne National Laboratory

Ken-ichi Nomura, Nitish Baradwaj

Collaboratory for Advanced Computing & Simulations

University of Southern California



U.S. DEPARTMENT OF ENERGY | Office of Science

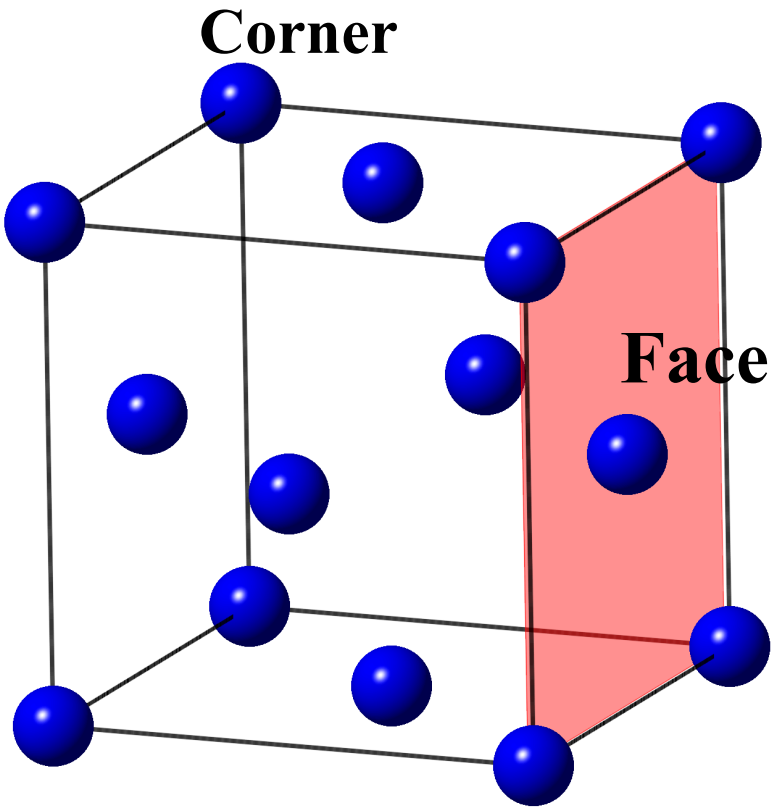
Basic Energy Sciences

Outline

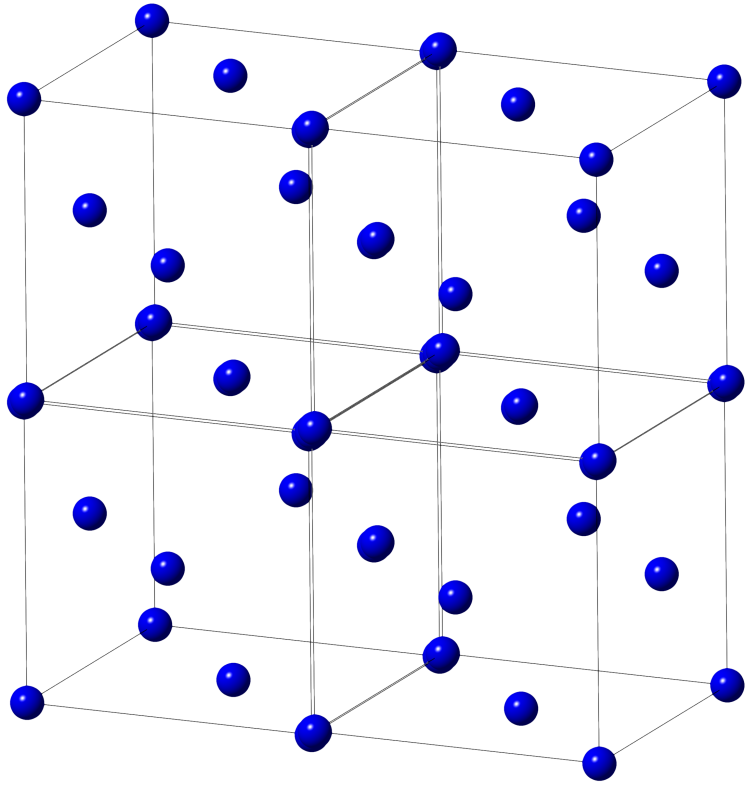
- FCC crystal and Nanoindentation MD simulation
- Machine Learning (ML) model for structural analysis
- Hands-on session:
  - Structural analysis using ML
  - Visualization of predicted label in OVITO

Face Centered Cubic (FCC) Crystal

In FCC crystal, each unit cell contain atoms on all the 8 corners and the 6 faces



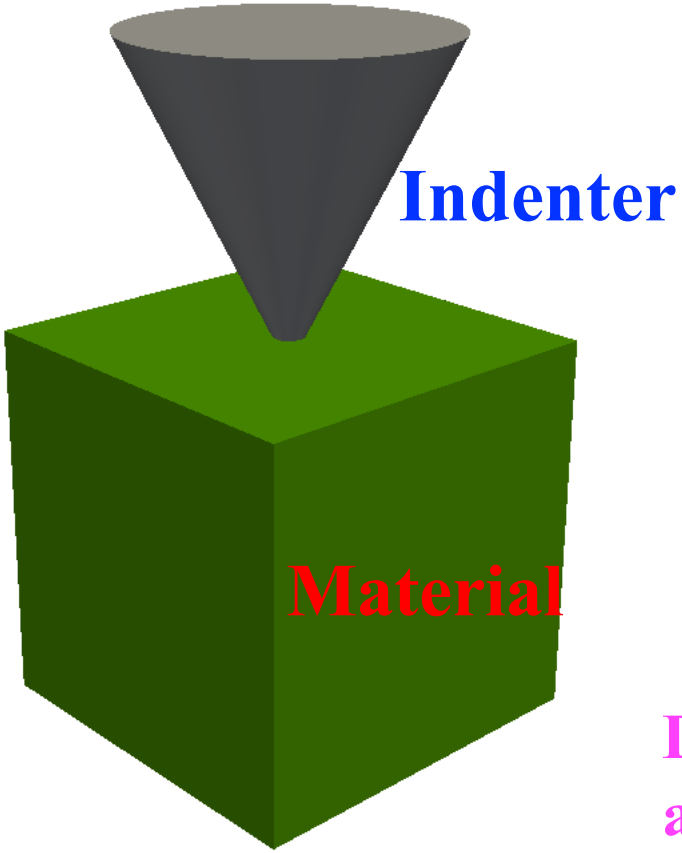
FCC unit cell

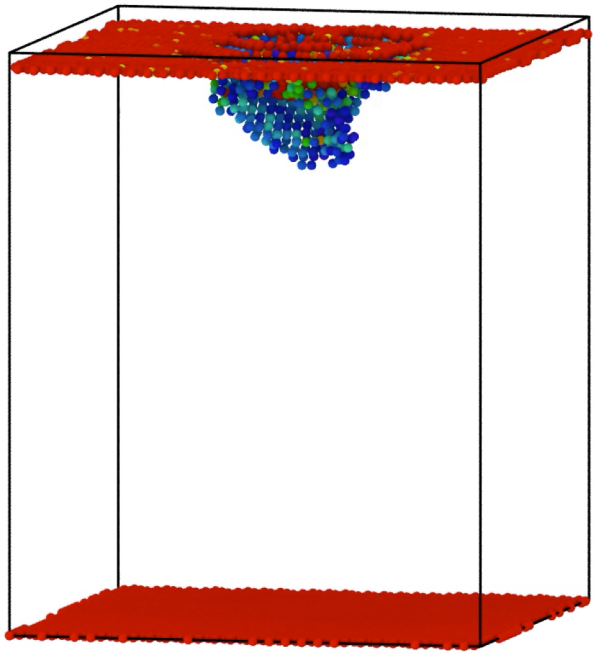


2×2×1 FCC cell

Nanoindentation Simulation of FCC Crystal using Molecular Dynamics (MD)

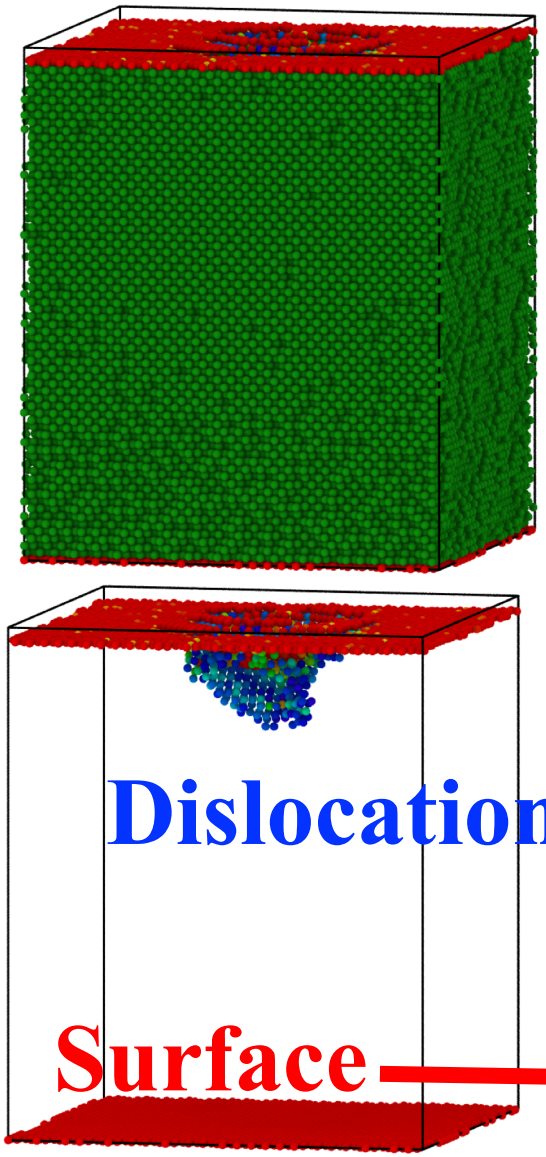
Nano-indentation Simulation: **Used to study mechanical properties of material (hardness, elastic constant)**





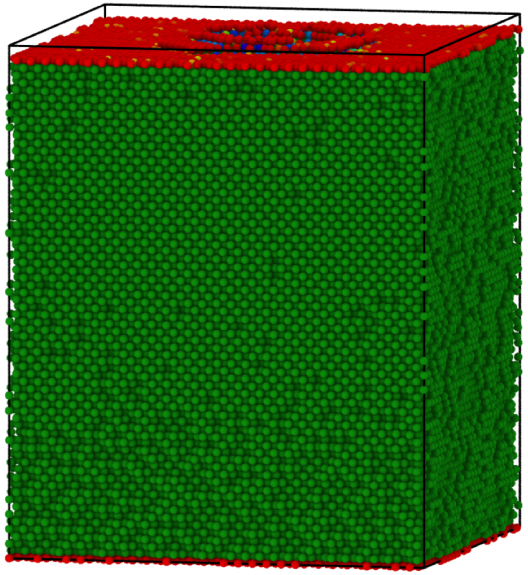
During MD simulation, dislocations are generated inside the material

Various Atomic Configurations in FCC crystal

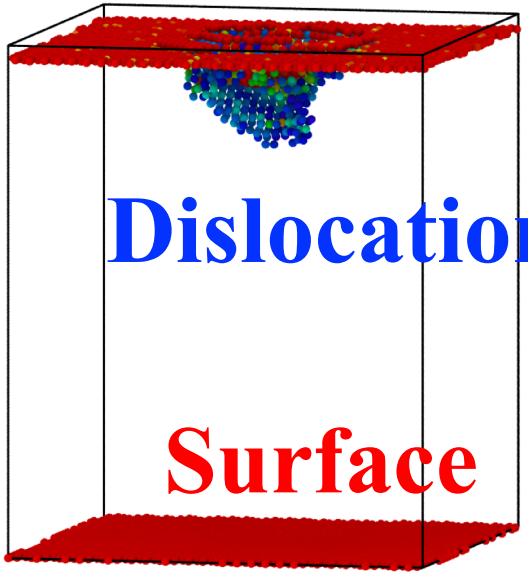


- Dislocation & surface atoms have missing neighbors

Various Atomic Configurations in FCC Crystal



Bulk Atom



Dislocation

Surface

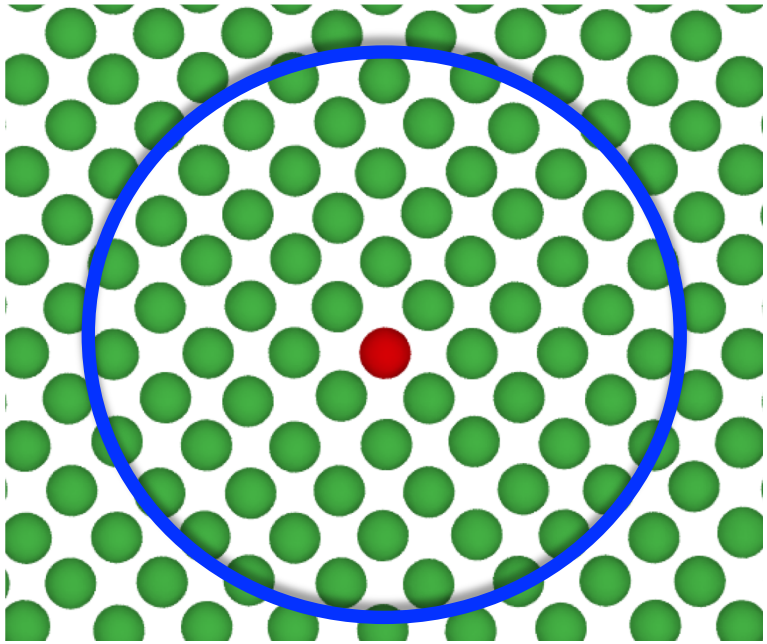
- **Goal: Build a Machine Learning (ML) model that can identify all these structures**

Labels for atomic configurations generated during nanoindentation simulation

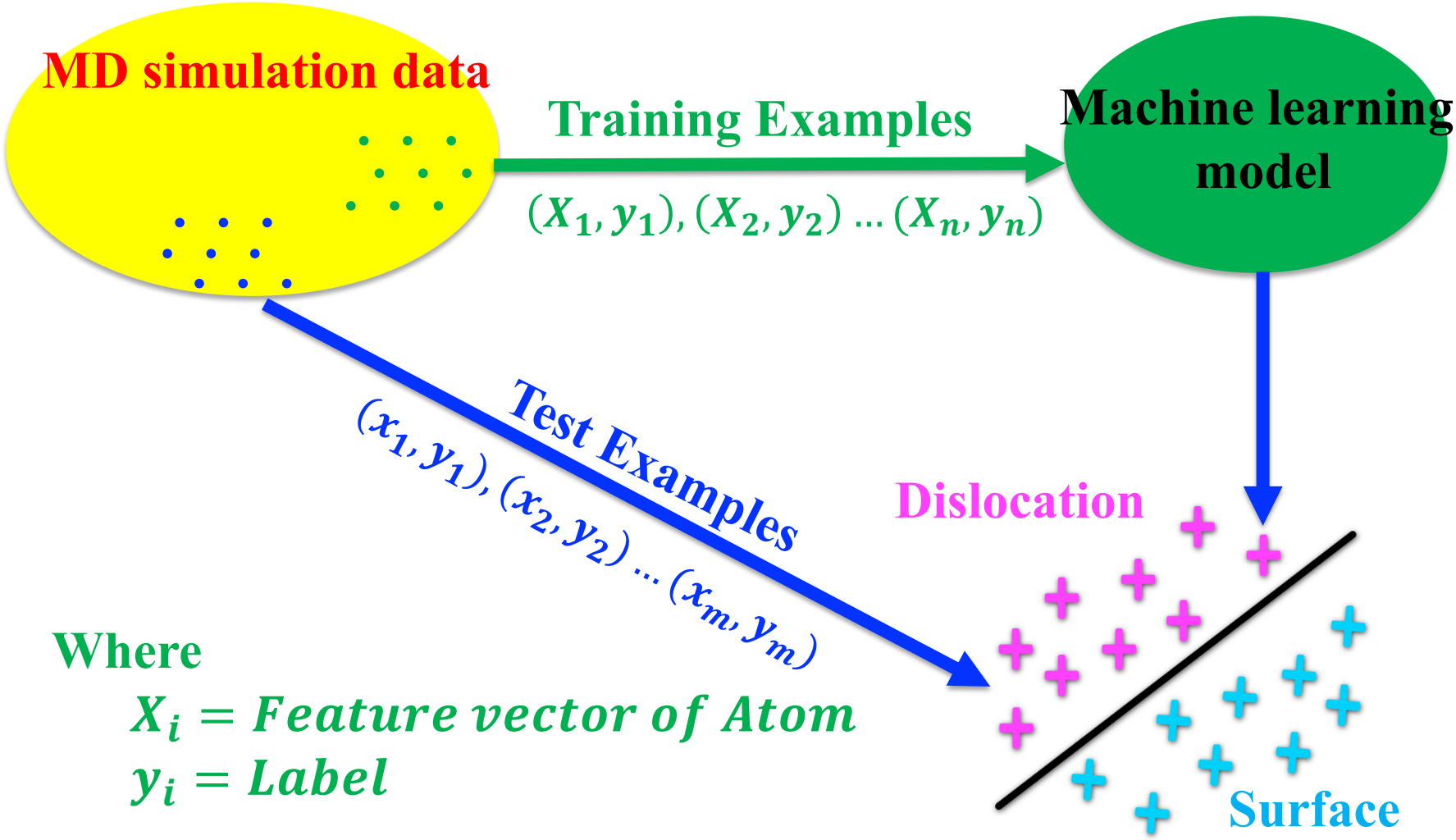
	Label (Y)
Bulk Atom	0
Surface	1
Dislocation	2

Feature Vector

A mathematic representation for each atom which uniquely describes the local environment of an atom

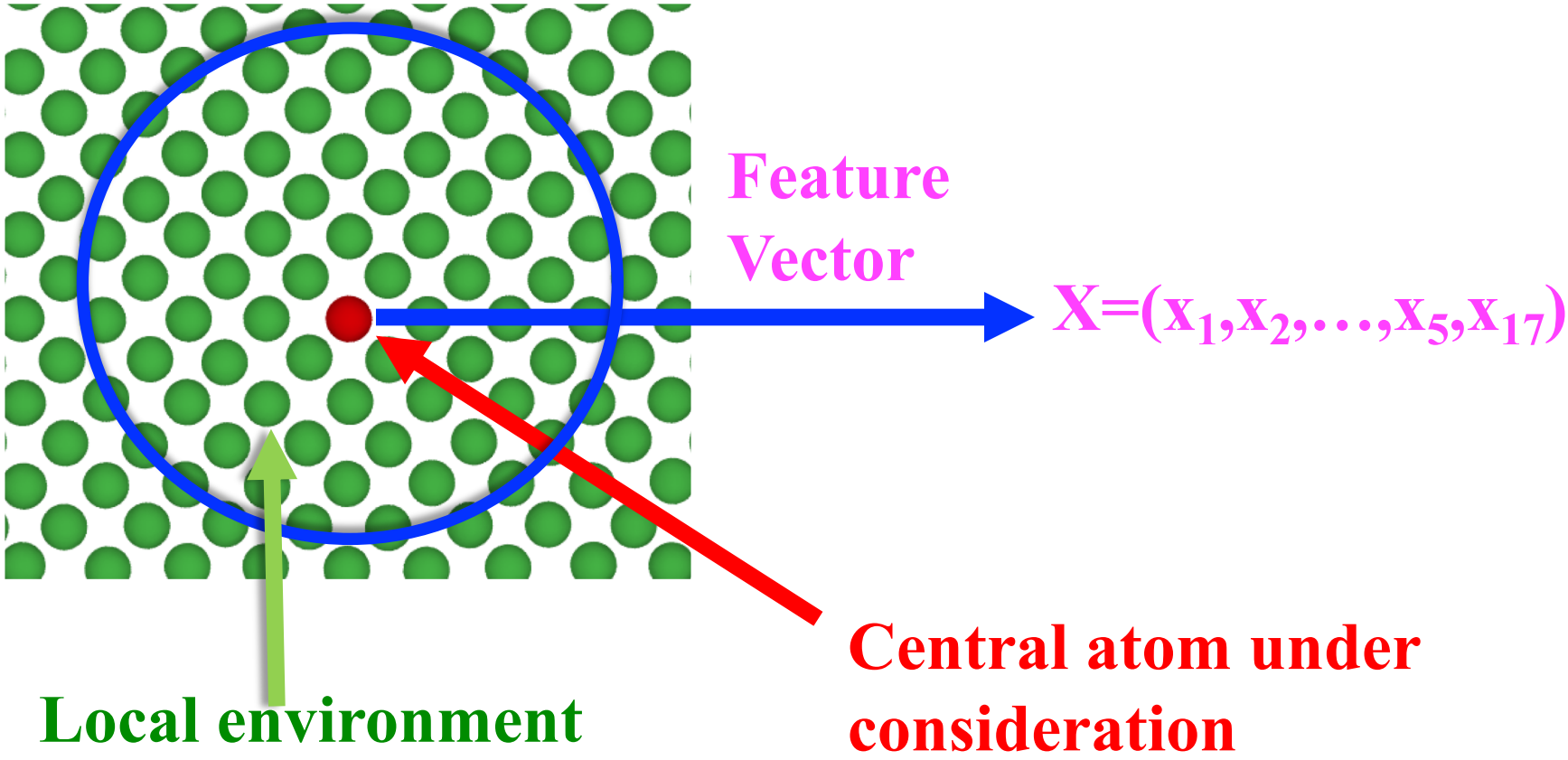


Objective: Build a Machine Learning (ML) Model for Structural Analysis



Step 1: Create Feature Vector for Each Atom

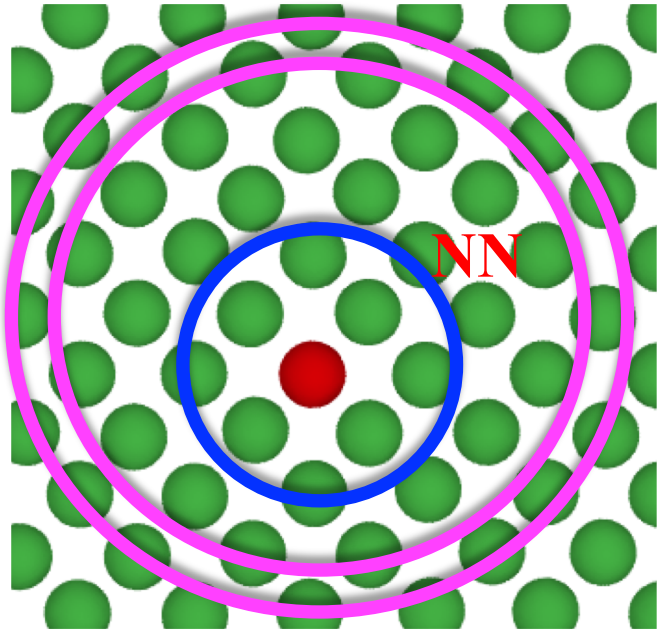
Convert atomic coordinates into a feature vector that captures the local geometry around each atom



Step 1: Create Feature Vector for Each Atom

Feature Vector: Each atoms local environment will be represented using 17 different properties

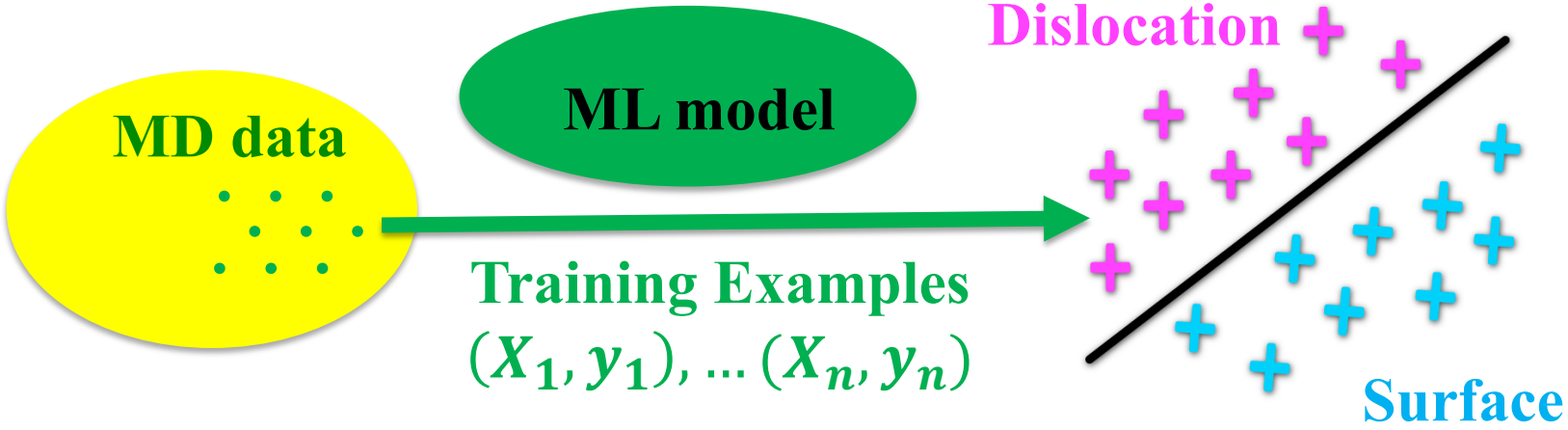
1. No. of nearest neighbor (NN)
2. Average distance of NN
3. Minimum distance of NN
4. Maximum distance of NN
5. Average distance between NN
6. Minimum distance between NN
7. Maximum distance between NN
8. NN's average numbers of neighbor
9. NN's neighbor's average distance
10. NN's neighbor's minimum distance
11. NN's neighbor's maximum distance



12-14 Number of neighbor's between 3-4, 4-5, 5-6A

15-17 Average distance of neighbor's between 3-4, 4-5, 5-6A

Step 2: Build a Linear Classifier using Machine Learning



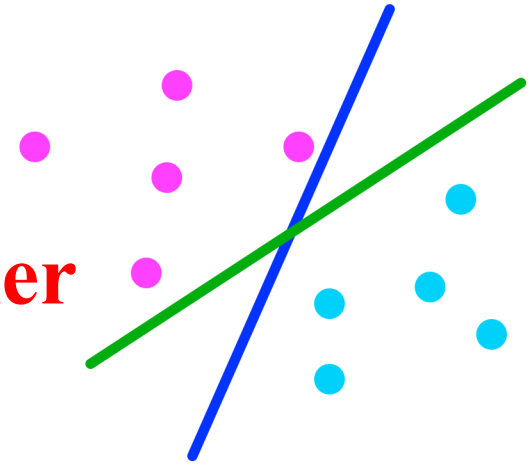
ML model (a linear classifier)

$$y_{predicted} = w_1 x_1 + \cdots + w_{17} x_{17} - b \begin{cases} > 0 \ (Dislocation) \\ \leq 0 \ (Surface) \end{cases}$$

w_i : tunable parameters that we will learn using training examples

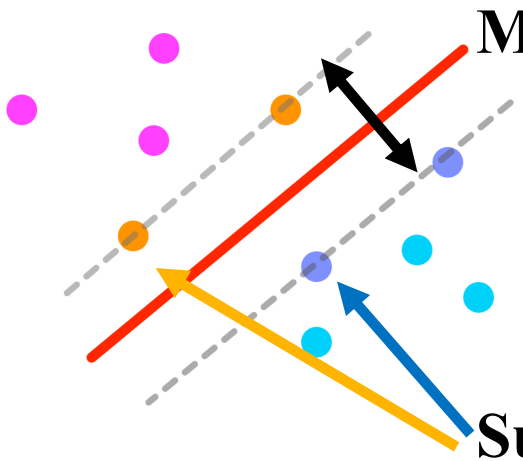
Linear Support Vector Machine (SVM)

Linear classifier



Many possible choices for decision boundary

Linear SVM



Maximizes the margin

Step 3: Evaluate Accuracy of Model

- **Compare the accuracy of the model on training and test data.**

$$Error = \frac{1}{N}\left(y_{predicted} - y_{true}\right)^2$$

$$Accuracy = 1 - Error$$

- **Visualize the predicted labels by the ML model on test data using OVITO**

Machine Learning Module for Structural Analysis

Files under Ml_module

```
$    ls Ml_module
```

```
Makefile            Ni_train.xyz        createfeature.h
Ni_ML.ipynb         SVM_model.py        feature/
Ni_ML.py            atom_property.c     readinput.py
Ni_test1.xyz        atom_property.h
Ni_test2.xyz        createfeature.c
```

1) atom_property.c and createfeature.c: C code to read atomic configuration and create feature vector for atoms

2) Ni_ML.py: python code to create and train ML classifier

3) SVM_model.py: python script to build the ML model

4) Ni_train.xyz, Ni_test1.xyz and Ni_test2.xyz : Atomic coordinate for training and test data

Files under Ml_module

```
$    ls Ml_module
```

```
Makefile            Ni_train.xyz       createfeature.h
Ni_ML.ipynb         SVM_model.py       feature/
Ni_ML.py            atom_property.c    readinput.py
Ni_test1.xyz        atom_property.h
Ni_test2.xyz        createfeature.c
```

1) atom_property.c and createfeature.c: C code to read atomic configuration and create feature vector for atoms

2) Ni_ML.py: python code to create and train ML classifier

3) SVM_model.py: python script to build the ML model

4) Ni_train.xyz, Ni_test1.xyz and Ni_test2.xyz : Atomic coordinate for training and test data

Files under Ml_module

```
$    ls Ml_module
```

```
Makefile            Ni_train.xyz       createfeature.h
Ni_ML.ipynb         SVM_model.py       feature/
Ni_ML.py            atom_property.c    readinput.py
Ni_test1.xyz        atom_property.h
Ni_test2.xyz        createfeature.c
```

1) atom_property.c and createfeature.c: C code to read atomic configuration and create feature vector for atoms

2) Ni_ML.py: python code to create and train ML classifier

3) SVM_model.py: python script to build the ML model

4) Ni_train.xyz, Ni_test1.xyz and Ni_test2.xyz : Atomic coordinate for training and test data

Files under Ml_module

```
$    ls Ml_module

Makefile            Ni_train.xyz      createfeature.h
Ni_ML.ipynb         SVM_model.py      feature/
Ni_ML.py            atom_property.c   readinput.py
Ni_test1.xyz        atom_property.h
Ni_test2.xyz        createfeature.c
```

1) atom_property.c and createfeature.c: C code to read atomic configuration and create feature vector for atoms

2) Ni_ML.py: python code to create and train ML classifier

3) SVM_model.py: python script to build the ML model

4) Ni_train.xyz, Ni_test1.xyz and Ni_test2.xyz : Atomic coordinate for training and test data

Compute Feature Vector from Atomic Coordinates for Training and Validation

- Using training data (Ni_train.xyz), compute feature vector for each atom
- First, build executable (c_feature) from the C codes (atom_property.c and createfeature.c).

```
make
```

- Using c_feature, compute feature vector from training data

```
./c_feature Ni_train.xyz feature/train.txt
```

Ni_train.xyz : *Input atomic coordinate in XYZ format*
feature/train.txt : *Output that contains feature vector for each atoms*

File Format that Contains Atomic Coordinates and Features

- Each line of the created feature vector contains atom type, x y z coordinates, label and 17 dimension feature vector for each atom

```
$   ls feature/train.txt
```

```
114376
17
  101.481003   130.103226   101.481003
Ni          1.756660     4.199579     1.764920     2     8.000000     2.471503     2.444679     2.500690     3.391102
Ni          1.746970     5.941798     0.018508     0    11.000000     2.480045     2.458448     2.507447     3.551417
Ni          5.244970     4.214318     1.754720     2     8.000000     2.467677     2.449782     2.491687     3.391526
Ni          3.489960     5.923518     1.752180     0    11.000000     2.474269     2.449782     2.492165     3.541959
Ni          3.495490     4.193375     0.006788     2     7.000000     2.476350     2.457604     2.497069     3.304576
Ni          8.770810     4.207564     1.757480     2     8.000000     2.473605     2.448607     2.490758     3.395039
Ni          6.990840     5.939387     1.742840     0    12.000000     2.478066     2.454397     2.486933     3.549592
Ni          8.760920     5.940617     0.005285     0    12.000000     2.477614     2.464313     2.500152     3.548933
Ni          6.998650     4.182088     0.009597     2     8.000000     2.480736     2.468257     2.497350     3.387989
Ni         12.249300     4.191433     1.748420     2     8.000000     2.471725     2.464085     2.477986     3.381069
Ni         10.503100     5.947286     1.755110     0    12.000000     2.474824     2.455090     2.500750     3.544996
Ni         12.237800     5.942111     0.005140     0    12.000000     2.474072     2.455739     2.493437     3.543886
Ni         15.734100     4.192092     1.764940     2     8.000000     2.480301     2.461069     2.498584     3.390528
Ni         13.984100     5.945300     1.749540     0    12.000000     2.472308     2.454947     2.483916     3.541379
Ni         13.987700     4.219180     0.002308     2     8.000000     2.467989     2.455739     2.481780     3.391241
Ni         15.745700     5.948635     0.000045     0    12.000000     2.474543     2.451119     2.490067     3.544575
Ni         17.493999     5.937821     1.738920     0    12.000000     2.474458     2.447819     2.490233     3.544428
```

type x,y,z coordinate label Feature vector

File Format that Contains Atomic Coordinates and Features

- Each line of the created feature vector contains atom type, x y z coordinates, label and 17 dimension feature vector for each atom

```
$   ls feature/train.txt
```

```
114376
17
  101.481003   130.103226   101.481003
Ni          1.756660     4.199579     1.764920     2     8.000000     2.471503     2.444679     2.500690     3.391102
Ni          1.746970     5.941798     0.018508     0    11.000000     2.480045     2.458448     2.507447     3.551417
Ni          5.244970     4.214318     1.754720     2     8.000000     2.467677     2.449782     2.491687     3.391526
Ni          3.489960     5.923518     1.752180     0    11.000000     2.474269     2.449782     2.492165     3.541959
Ni          3.495490     4.193375     0.006788     2     7.000000     2.476350     2.457604     2.497069     3.304576
Ni          8.770810     4.207564     1.757480     2     8.000000     2.473605     2.448607     2.490758     3.395039
Ni          6.990840     5.939387     1.742840     0    12.000000     2.478066     2.454397     2.486933     3.549592
Ni          8.760920     5.940617     0.005285     0    12.000000     2.477614     2.464313     2.500152     3.548933
Ni          6.998650     4.182088     0.009597     2     8.000000     2.480736     2.468257     2.497350     3.387989
Ni         12.249300     4.191433     1.748420     2     8.000000     2.471725     2.464085     2.477986     3.381069
Ni         10.503100     5.947286     1.755110     0    12.000000     2.474824     2.455090     2.500750     3.544996
Ni         12.237800     5.942111     0.005140     0    12.000000     2.474072     2.455739     2.493437     3.543886
Ni         15.734100     4.192092     1.764940     2     8.000000     2.480301     2.461069     2.498584     3.390528
Ni         13.984100     5.945300     1.749540     0    12.000000     2.472308     2.454947     2.483916     3.541379
Ni         13.987700     4.219180     0.002308     2     8.000000     2.467989     2.455739     2.481780     3.391241
Ni         15.745700     5.948635     0.000045     0    12.000000     2.474543     2.451119     2.490067     3.544575
Ni         17.493999     5.937821     1.738920     0    12.000000     2.474458     2.447819     2.490233     3.544428
```

type x,y,z coordinate label Feature vector

File Format that Contains Atomic Coordinates and Features

- Each line of the created feature vector contains atom type, x y z coordinates, label and 17 dimension feature vector for each atom

```
$   ls feature/train.txt
```

```
114376
17
  101.481003   130.103226   101.481003
Ni        1.756660    4.199579    1.764920    2    8.000000    2.471503    2.444679    2.500690    3.391102
Ni        1.746970    5.941798    0.018508    0   11.000000    2.480045    2.458448    2.507447    3.551417
Ni        5.244970    4.214318    1.754720    2    8.000000    2.467677    2.449782    2.491687    3.391526
Ni        3.489960    5.923518    1.752180    0   11.000000    2.474269    2.449782    2.492165    3.541959
Ni        3.495490    4.193375    0.006788    2    7.000000    2.476350    2.457604    2.497069    3.304576
Ni        8.770810    4.207564    1.757480    2    8.000000    2.473605    2.448607    2.490758    3.395039
Ni        6.990840    5.939387    1.742840    0   12.000000    2.478066    2.454397    2.486933    3.549592
Ni        8.760920    5.940617    0.005285    0   12.000000    2.477614    2.464313    2.500152    3.548933
Ni        6.998650    4.182088    0.009597    2    8.000000    2.480736    2.468257    2.497350    3.387989
Ni       12.249300    4.191433    1.748420    2    8.000000    2.471725    2.464085    2.477986    3.381069
Ni       10.503100    5.947286    1.755110    0   12.000000    2.474824    2.455090    2.500750    3.544996
Ni       12.237800    5.942111    0.005140    0   12.000000    2.474072    2.455739    2.493437    3.543886
Ni       15.734100    4.192092    1.764940    2    8.000000    2.480301    2.461069    2.498584    3.390528
Ni       13.984100    5.945300    1.749540    0   12.000000    2.472308    2.454947    2.483916    3.541379
Ni       13.987700    4.219180    0.002308    2    8.000000    2.467989    2.455739    2.481780    3.391241
Ni       15.745700    5.948635    0.000045    0   12.000000    2.474543    2.451119    2.490067    3.544575
Ni       17.493999    5.937821    1.738920    0   12.000000    2.474458    2.447819    2.490233    3.544428
```

type x,y,z coordinate label Feature vector

File Format that Contains Atomic Coordinates and Features

- Each line of the created feature vector contains atom type, x y z coordinates, label and 17 dimension feature vector for each atom

```
$   ls feature/train.txt
```

```
114376
17
  101.481003   130.103226   101.481003
Ni          1.756660     4.199579     1.764920     2     8.000000     2.471503     2.444679     2.500690     3.391102
Ni          1.746970     5.941798     0.018508     0    11.000000     2.480045     2.458448     2.507447     3.551417
Ni          5.244970     4.214318     1.754720     2     8.000000     2.467677     2.449782     2.491687     3.391526
Ni          3.489960     5.923518     1.752180     0    11.000000     2.474269     2.449782     2.492165     3.541959
Ni          3.495490     4.193375     0.006788     2     7.000000     2.476350     2.457604     2.497069     3.304576
Ni          8.770810     4.207564     1.757480     2     8.000000     2.473605     2.448607     2.490758     3.395039
Ni          6.990840     5.939387     1.742840     0    12.000000     2.478066     2.454397     2.486933     3.549592
Ni          8.760920     5.940617     0.005285     0    12.000000     2.477614     2.464313     2.500152     3.548933
Ni          6.998650     4.182088     0.009597     2     8.000000     2.480736     2.468257     2.497350     3.387989
Ni         12.249300     4.191433     1.748420     2     8.000000     2.471725     2.464085     2.477986     3.381069
Ni         10.503100     5.947286     1.755110     0    12.000000     2.474824     2.455090     2.500750     3.544996
Ni         12.237800     5.942111     0.005140     0    12.000000     2.474072     2.455739     2.493437     3.543886
Ni         15.734100     4.192092     1.764940     2     8.000000     2.480301     2.461069     2.498584     3.390528
Ni         13.984100     5.945300     1.749540     0    12.000000     2.472308     2.454947     2.483916     3.541379
Ni         13.987700     4.219180     0.002308     2     8.000000     2.467989     2.455739     2.481780     3.391241
Ni         15.745700     5.948635     0.000045     0    12.000000     2.474543     2.451119     2.490067     3.544575
Ni         17.493999     5.937821     1.738920     0    12.000000     2.474458     2.447819     2.490233     3.544428
```

type x,y,z coordinate label Feature vector

File Format that Contains Atomic Coordinates and Features

- Each line of the created feature vector contains atom type, x y z coordinates, label and 17 dimension feature vector for each atom

```
$    ls feature/train.txt
```

```
114376
17
  101.481003   130.103226   101.481003
Ni          1.756660     4.199579     1.764920     2     8.000000  2.471503  2.444679  2.500690  3.391102
Ni          1.746970     5.941798     0.018508     0    11.000000  2.480045  2.458448  2.507447  3.551417
Ni          5.244970     4.214318     1.754720     2     8.000000  2.467677  2.449782  2.491687  3.391526
Ni          3.489960     5.923518     1.752180     0    11.000000  2.474269  2.449782  2.492165  3.541959
Ni          3.495490     4.193375     0.006788     2     7.000000  2.476350  2.457604  2.497069  3.304576
Ni          8.770810     4.207564     1.757480     2     8.000000  2.473605  2.448607  2.490758  3.395039
Ni          6.990840     5.939387     1.742840     0    12.000000  2.478066  2.454397  2.486933  3.549592
Ni          8.760920     5.940617     0.005285     0    12.000000  2.477614  2.464313  2.500152  3.548933
Ni          6.998650     4.182088     0.009597     2     8.000000  2.480736  2.468257  2.497350  3.387989
Ni         12.249300     4.191433     1.748420     2     8.000000  2.471725  2.464085  2.477986  3.381069
Ni         10.503100     5.947286     1.755110     0    12.000000  2.474824  2.455090  2.500750  3.544996
Ni         12.237800     5.942111     0.005140     0    12.000000  2.474072  2.455739  2.493437  3.543886
Ni         15.734100     4.192092     1.764940     2     8.000000  2.480301  2.461069  2.498584  3.390528
Ni         13.984100     5.945300     1.749540     0    12.000000  2.472308  2.454947  2.483916  3.541379
Ni         13.987700     4.219180     0.002308     2     8.000000  2.467989  2.455739  2.481780  3.391241
Ni         15.745700     5.948635     0.000045     0    12.000000  2.474543  2.451119  2.490067  3.544575
Ni         17.493999     5.937821     1.738920     0    12.000000  2.474458  2.447819  2.490233  3.544428
```

type x,y,z coordinate label Feature vector

Python Class to Build SVM Linear Classifier

build_classifier provides three member functions; train, prediction and accuracy.

```
class build_classifier:
    def __init__(self, trainX, trainY):
        ▪ load training data (trainX)
        ▪ Load true labels (trainY)
        ▪ Normalize training data (trainX)
    def  train(self):
        ▪ Train the ML model using normalize training data
    def  predict(self):
        ▪ predict labels of test data using trained model
    def accuracy(self):
        ▪ computes prediction accuracy of the model
```

Python Class to Build SVM Linear Classifier

Training data (*trainX*) and true label (*trainY*) is necessary to instantiate the class.

```
class build_classifier:
    def __init__(self, trainX, trainY):
        ▪ load training data (trainX)
        ▪ Load true labels (trainY)
        ▪ Normalize training data (trainX)
    def train(self):
        ▪ Train the ML model using normalize training data
    def predict(self):
        ▪ predict labels of test data using trained model
    def accuracy(self):
        ▪ computes prediction accuracy of the model
```

Python Class to Build SVM Linear Classifier

Trains the ML model given training data and true labels.

```
class build_classifier:
    def __init__(self, trainX, trainY):
        ▪ load training data (trainX)
        ▪ Load true labels (trainY)
        ▪ Normalize training data (trainX)
    def train(self):
        ▪ Train the ML model using normalize training data
    def predict(self):
        ▪ predict labels of test data using trained model
    def accuracy(self):
        ▪ computes prediction accuracy of the model
```

Python Class to Build SVM Linear Classifier

Using the trained model, predict label for test data.

```
class build_classifier:
    def __init__(self, trainX, trainY):
        ▪ load training data (trainX)
        ▪ Load true labels (trainY)
        ▪ Normalize training data (trainX)
    def  train(self):
        ▪ Train the ML model using normalize training data
    def  predict(self):
        ▪ predict labels of test data using trained model
    def accuracy(self):
        ▪ computes prediction accuracy of the model
```

Python Class to Build SVM Linear Classifier

Calculate error and accuracy of the developed model.

class build_classifier:

def __init__(self, trainX, trainY):

- load training data (trainX)
- Load true labels (trainY)
- Normalize training data (trainX)

def train(self):

- Train the ML model using normalize training data

def predict(self):

- predict labels of test data using trained model

def accuracy(self):

- **computes prediction accuracy of the model**

1. Train the Linear SVM Classifier

- Instantiate build_classifier class

```
Ni_model = build_classifier (train_X, train_Y)
```

- Trains a linear classifier using svm.LinearSVC(), and store the model into a variable self.model.

```
Ni_model.train()
```

- train() also computes error and accuracy of the developed model.

Number of training examples: 18686
Training error = 3.16%
Training accuracy = 96.83%

2. Compute Accuracy of the Model using Test Data

- Convert atomic coordinates of test data (Ni_test1.xyz) into feature vector.

```
./c_feature Ni_test1.xyz feature/test_1.txt
```

- Predict labels of the test data using the trained model by predict() and accuracy() function.

```
label_pred = Ni_model.predict(testX)
accuracy = Ni_model.accuracy(textY,label_pred)
```

Output:

Test error = 1.03%
Test accuracy = 98.96%

3. Visualize the Predicted Label in OVITO

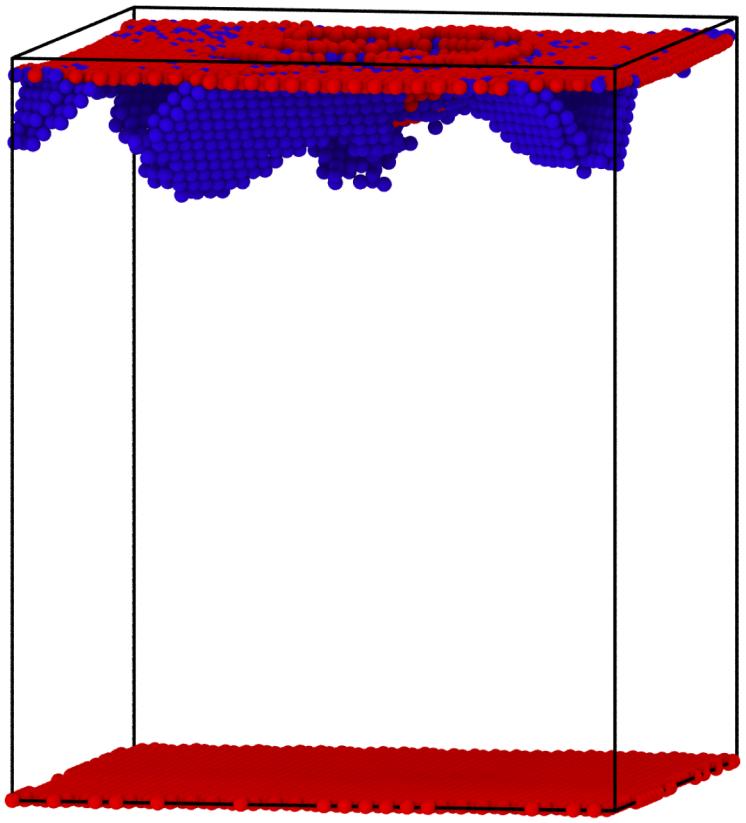
- writexyz() function creates an output file (output.xyz) that contains atomic coordinates and the true and predicted labels

```
writexyz(N_atoms,position,label_pred,label_true)
```

- Visualize the predicted label (output.xyz) in OVITO

3. Visualize the Predicted Label in OVITO

True Label



ML Predicted Label