Example 2: Fingerprint atomic crystal structures and learning their energy

Huan Tran

The main objective of this example is to get a small dataset of atomic crystal structures and their energy, fingerprint them, and develop some ML models using different learning algorithms.

1. Download data

The dataset contains 329 equilibrium structures of 13 different stoichiometries of Mg and Si, whose energy was computed using DFT. This dataset was reported in [T. D. Huan, *Pressure-stabilized binary compounds of magnesium and silicon*, Phys. Rev. Materials **2**, 023803 (2018)]. It will be obtained from www.matsml.org. More information on the avaiable datasets can be found at www.matsml.org as well.

In [1]:
```python
from matsml.data import Datasets
import os
import pandas as pd

# Load data
ds_name='crystals_MgSi'
data=Datasets(ds_name=ds_name)
data.load_dataset()

# have a look at the content
print (pd.read_csv(os.path.join(os.getcwd(),str(ds_name),'summary.csv')))
```

```
  matsML, v1.3.0
  *****
  Load requested dataset(s)
  Data saved in crystals_MgSi
                file_name     target
0    mg2si_struct_01.vasp  -8.924797
1    mg2si_struct_02.vasp -34.985707
2    mg2si_struct_03.vasp -17.246812
3    mg2si_struct_04.vasp -34.062642
4    mg2si_struct_05.vasp -34.035175
..                    ...        ...
324   mgsi_struct_30.vasp -40.698471
325   mgsi_struct_31.vasp -40.598719
326   mgsi_struct_32.vasp -40.499177
327   mgsi_struct_33.vasp  -6.706034
328   mgsi_struct_34.vasp -40.362384

[329 rows x 2 columns]
```

2. Fingerprint the obtained data

Two kinds of crystal fingerprints will be used in this example

1. Ewald sum matrix [Fe. Faber, A. Lindmaa, O. Anatole von Lilienfeld, and R. Armiento. *Crystal structure representations for machine learning models of formation energies* Int. J. Quantum Chem., 115, 1094 (2015)] is an analogy to the Coulomb matrix for molecules, and its size also depends on the number of atoms of the structure. We used a similar projection on a set of Gaussian. Keyword for this fingerprint is **pesm_crystals**.
2. Smooth Overlap of Atomic Positions (SOAP) [S. De, A. P. Bartók, G. Csányi, and M. Ceriotti, *Comparing molecules and solids across structural and alchemical space*, Phys. Chem. Chem. Phys. **18**, 13754 (2016)] is a more sophisticated fingerprint. Different from the Ewald sum matrix which is defined for the whole system, SOAP is defined for each atom. Herein, for simplicity, the atomic fingerprints are added up to make the fingerprint for the whole system. In some ML potential, the SOAP fingerprints are used in a different way, involving the "atomic energy". The keyword for SOAP in matsML is **soap_crystals**.

In [2]:
```python
import os
import pandas as pd
from matsml.fingerprint import Fingerprint

summary = os.path.join(os.getcwd(), 'crystals_MgSi/summary.csv')
data_loc = os.path.join(os.getcwd(), 'crystals_MgSi')
fp_dim = 20                               # intended fingerprint dimensionality
verbosity = 0                             # verbosity, 0 or 1

# Ewald sum matrix
data_params_pesm = {
    'summary': summary,
    'data_loc': data_loc,
    'fp_file': 'fp_crystals_MgSi_pesm.csv',
    'fp_type': 'pesm_crystals',
    'fp_dim': fp_dim,
    'verbosity': verbosity,
}

fp_pesm = Fingerprint(data_params_pesm)
fp_pesm.get_fingerprint()

# SOAP
data_params_soap = {
    'summary': summary,
    'data_loc': data_loc,
    'fp_file': 'fp_crystals_MgSi_soap.csv',
    'fp_type': 'soap_crystals',
    'fp_dim': fp_dim,
    'verbosity': verbosity,
}

fp_soap = Fingerprint(data_params_soap)
fp_soap.get_fingerprint()
```

```
  Atomic structure fingerprinting
    summary                       /home/huantran/work_local/matsml/examples/ex3_crystals/crystals_MgSi/summary.c
sv
    data_loc                      /home/huantran/work_local/matsml/examples/ex3_crystals/crystals_MgSi
    fp_type                       pesm_crystals
    fp_file                       fp_crystals_MgSi_pesm.csv
    fp_dim                        20
    verbosity                     0
  Read input
    num_structs                   329
  Computing Ewald sum Matrix
    [==================================================] 100%
  Projecting Ewald sum matrix to create fingerprints
    [==================================================] 100%
  Done fingerprinting, results saved in fp_crystals_MgSi_pesm.csv
  Atomic structure fingerprinting
    summary                       /home/huantran/work_local/matsml/examples/ex3_crystals/crystals_MgSi/summary.c
sv
    data_loc                      /home/huantran/work_local/matsml/examples/ex3_crystals/crystals_MgSi
    fp_type                       soap_crystals
    fp_file                       fp_crystals_MgSi_soap.csv
    fp_dim                        20
    verbosity                     0
  Read input
    num_structs                   329
  Computing SOAP fingerprint with DScribe
    [==================================================] 100%
  Done fingerprinting, results saved in fp_crystals_MgSi_soap.csv
```

The fingerprinting step maybe a bit slow for a tutorial because we need to set n_atoms_max=28, which results in quite large Ewald sum matricies. A version of fingerprinted data can also be obtained in case you want to skip this step.

In [3]:
```python
from matsml.data import Datasets
import os
import pandas as pd

# Load data
data = Datasets(ds_soap='fp_crystals_MgSi_soap', ds_pesm='fp_crystals_MgSi_pesm')
data.load_dataset()

print(os.path.isfile('fp_crystals_MgSi_soap.csv.gz'))
print(os.path.isfile('fp_crystals_MgSi_pesm.csv.gz'))
```

```
  Load requested dataset(s)
  Data saved in fp_crystals_MgSi_soap.csv.gz
  Data saved in fp_crystals_MgSi_pesm.csv.gz
True
True
```

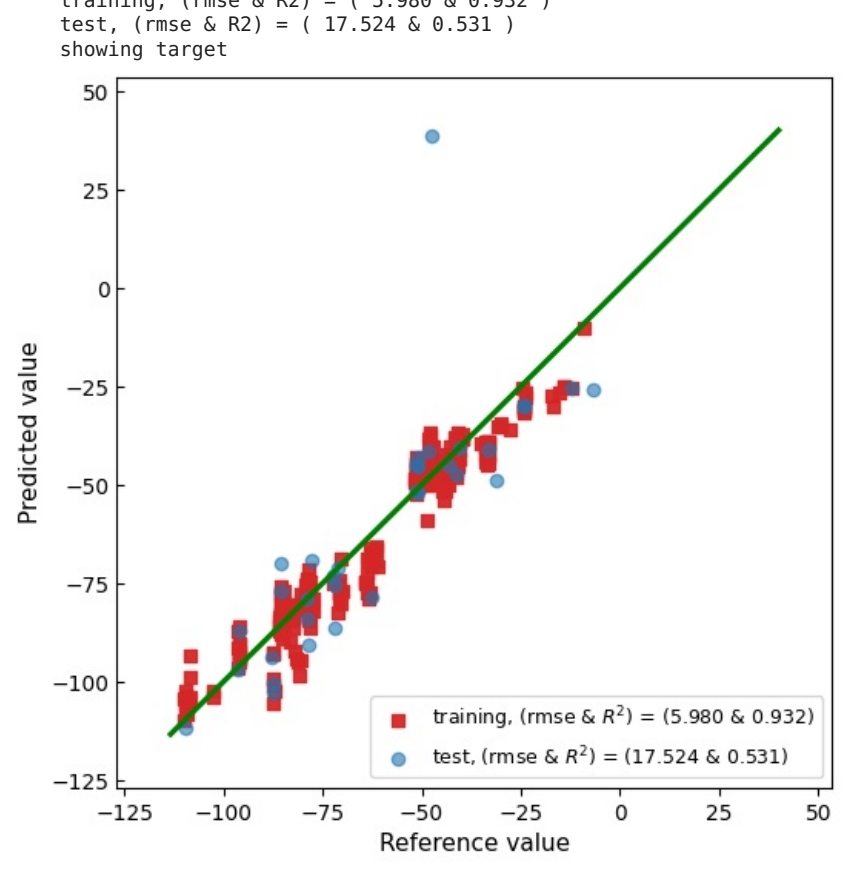
3. Train several ML models with two fingerprint files just created

In [4]:
```python
# data parameters for learning

id_col = ['id']                     # ID column in the fingerprint data
y_cols = ['target']                 # target (y) column(s)
comment_cols = []                  # columns not used as id, x, or y
n_trains = 0.9                      # 90% for training, 10% for validating
sampling = 'random'                 # method for train/test splitting
x_scaling = 'minmax'
y_scaling = 'minmax'

data_params_pesm = {
    'data_file': 'fp_crystals_MgSi_pesm.csv',
    'id_col': id_col,
    'y_cols': y_cols,
    'comment_cols': comment_cols,
    'y_scaling': y_scaling,
    'x_scaling': x_scaling,
    'sampling': sampling,
    'n_trains': n_trains,
}

data_params_soap = {
    'data_file': 'fp_crystals_MgSi_soap.csv',
    'id_col': id_col,
    'y_cols': y_cols,
    'comment_cols': comment_cols,
    'y_scaling': y_scaling,
    'x_scaling': x_scaling,
    'sampling': sampling,
    'n_trains': n_trains,
}
```

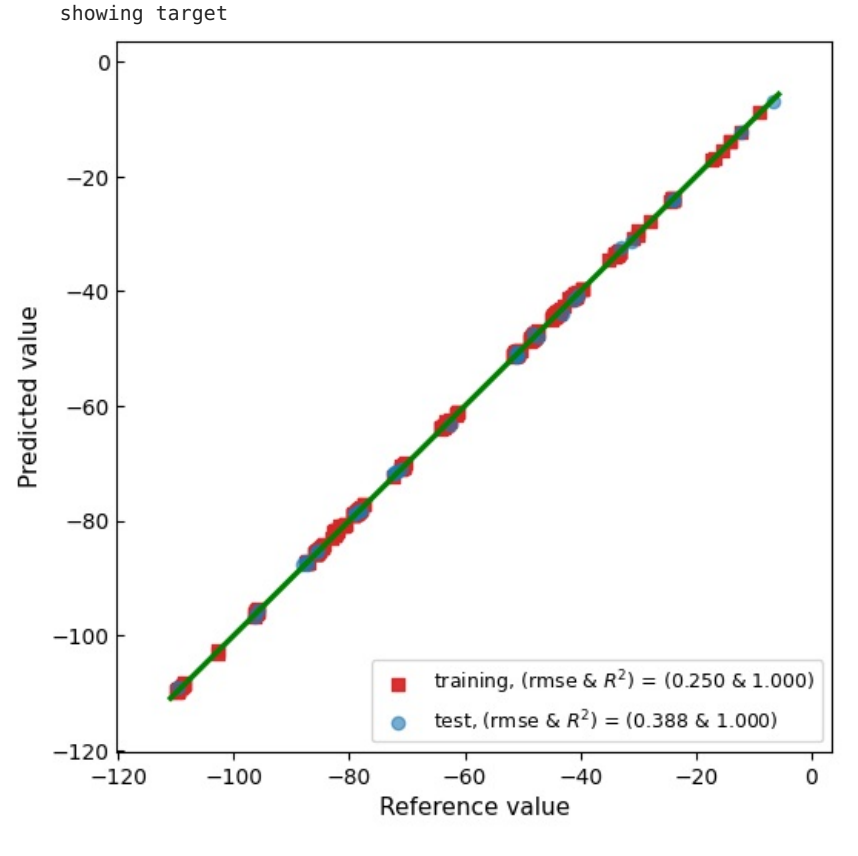
3a. Fully-connected NeuralNet

In [5]:
```python
from matsml.models import FCNN

# Model parameters
layers = [4, 4]              # list of nodes in hidden layers
epochs = 2000                # Epochs
nfold_cv = 5                 # Number of folds for cross validation
use_bias = True              # Use bias term or not
model_file = 'model_nn.pkl'  # Name of the model file to be created
verbosity = 0                # Verbosity, 0 or 1
batch_size = 32              # Default batch size
loss = 'mse'
activ_funct = 'selu'         # Options: "tanh", "relu", and more
optimizer = 'nadam'          # Options: "Nadam", "Adam", and more

model_params = {
    'layers': layers,
    'activ_funct': activ_funct,
    'epochs': epochs,
    'nfold_cv': nfold_cv,
    'optimizer': optimizer,
    'use_bias': use_bias,
    'model_file': model_file,
    'loss': loss,
    'batch_size': batch_size,
    'verbosity': verbosity,
    'rmse_cv': False,
}

# PESM
model = FCNN(data_params=data_params_pesm, model_params=model_params)
model.train()
model.plot(pdf_output=False)

# SOAP
model = FCNN(data_params=data_params_soap, model_params=model_params)
model.train()
model.plot(pdf_output=False)
```

```
  Checking parameters
    model_file not ends with ".weights.h5", renamed

  Learning fingerprinted/featured data
    algorithm                     fully connected NeuralNet w/ TensorFlow
    layers                        [4, 4]
    activ_funct                   selu
    epochs                        2000
    optimizer                     nadam
    nfold_cv                      5
    verbosity                     0

  Read data
    data file                     fp_crystals_MgSi_pesm.csv
    data size                     329
    training size                 90.0  %
    test size                     10.0  %
    x dimensionality              19
    y dimensionality              1
    y label(s)                    ['target']

  Scaling x                       minmax
    xscaler saved in              xscaler.pkl

  Scaling y                       minmax

  Prepare train/test sets         random

  Building model                  FCNN
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1765089381.378611 1035390 gpu_device.cc:2020] Created device /job:localhost/replica:0/task:0/device
:GPU:0 with 4598 MB memory:  -> device: 0, name: NVIDIA GeForce RTX 2060, pci bus id: 0000:01:00.0, compute cap
ability: 7.5
  Training model w/ cross validation
I0000 00:00:1765089382.022953 1035497 device_compiler.h:196] Compiled cluster using XLA!  This line is logged a
t most once for the lifetime of the process.
8/8 [==============================] - 0s 506us/step
2/2 [==============================] - 0s 751us/step
    cv,rmse_train,rmse_test,rmse_opt: 0 0.060035 0.088732 0.088732
8/8 [==============================] - 0s 502us/step
2/2 [==============================] - 0s 799us/step
    cv,rmse_train,rmse_test,rmse_opt: 1 0.056679 0.084836 0.084836
8/8 [==============================] - 0s 520us/step
2/2 [==============================] - 0s 799us/step
    cv,rmse_train,rmse_test,rmse_opt: 2 0.055376 0.071329 0.071329
8/8 [==============================] - 0s 511us/step
2/2 [==============================] - 0s 784us/step
    cv,rmse_train,rmse_test,rmse_opt: 3 0.054338 0.072815 0.071329
8/8 [==============================] - 0s 460us/step
2/2 [==============================] - 0s 717us/step
    cv,rmse_train,rmse_test,rmse_opt: 4 0.055081 0.068306 0.068306
    Optimal ncv:  4 ; optimal NET saved
  FCNN trained, now make predictions & invert scaling
10/10 [==============================] - 0s 445us/step
    unscaling y: minmax
       rmse training     target           5.979827
2/2 [==============================] - 0s 2ms/step
    unscaling y: minmax
       rmse test         target           17.523834
  Predictions made & saved in "training.csv" & "test.csv"

  Plot results in "training.csv" & "test.csv"
    training, (rmse & R2) = ( 5.980 & 0.932 )
    test, (rmse & R2) = ( 17.524 & 0.531 )
    showing target
```



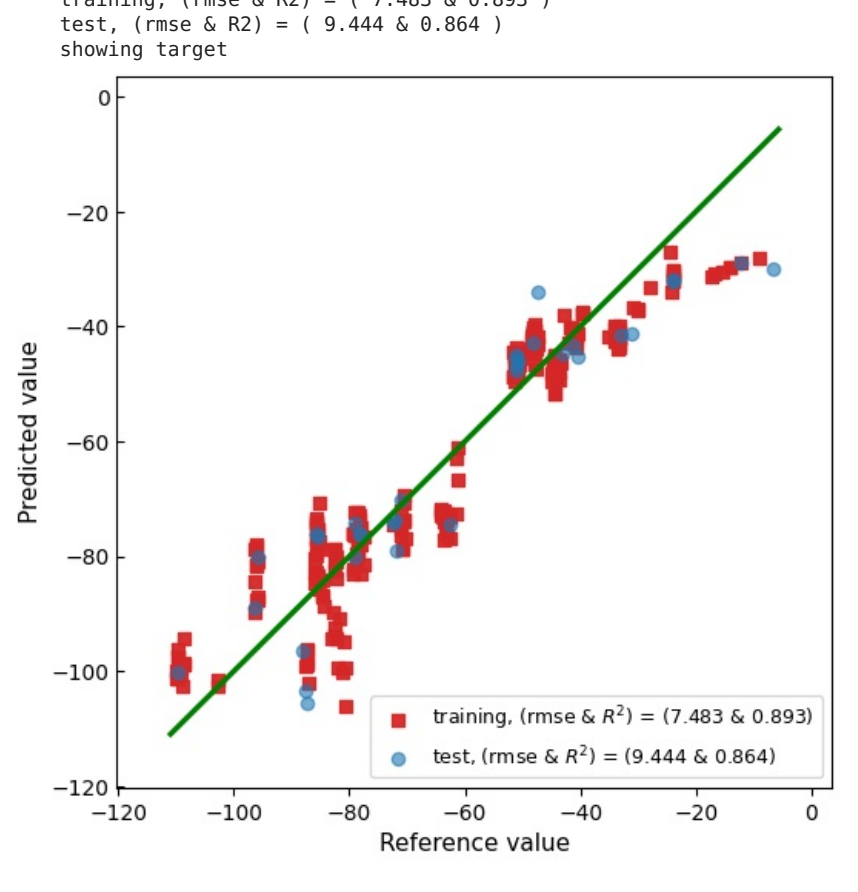

```
  Checking parameters
    model_file not ends with ".weights.h5", renamed

  Learning fingerprinted/featured data
    algorithm                     fully connected NeuralNet w/ TensorFlow
    layers                        [4, 4]
    activ_funct                   selu
    epochs                        2000
    optimizer                     nadam
    nfold_cv                      5
    verbosity                     0

  Read data
    data file                     fp_crystals_MgSi_soap.csv
    data size                     329
    training size                 90.0  %
    test size                     10.0  %
    x dimensionality              735
    y dimensionality              1
    y label(s)                    ['target']

  Scaling x                       minmax
    xscaler saved in              xscaler.pkl

  Scaling y                       minmax

  Prepare train/test sets         random

  Building model                  FCNN
  Training model w/ cross validation
8/8 [==============================] - 0s 558us/step
2/2 [==============================] - 0s 885us/step
    cv,rmse_train,rmse_test,rmse_opt: 0 0.004027 0.006279 0.006279
8/8 [==============================] - 0s 545us/step
2/2 [==============================] - 0s 836us/step
    cv,rmse_train,rmse_test,rmse_opt: 1 0.002263 0.004627 0.004627
8/8 [==============================] - 0s 564us/step
2/2 [==============================] - 0s 942us/step
    cv,rmse_train,rmse_test,rmse_opt: 2 0.003888 0.004913 0.004627
8/8 [==============================] - 0s 544us/step
2/2 [==============================] - 0s 859us/step
    cv,rmse_train,rmse_test,rmse_opt: 3 0.003208 0.004591 0.004591
8/8 [==============================] - 0s 500us/step
2/2 [==============================] - 0s 902us/step
    cv,rmse_train,rmse_test,rmse_opt: 4 0.002134 0.003330 0.003330
    Optimal ncv:  4 ; optimal NET saved
  FCNN trained, now make predictions & invert scaling
10/10 [==============================] - 0s 506us/step
    unscaling y: minmax
       rmse training     target           0.249655
2/2 [==============================] - 0s 1ms/step
    unscaling y: minmax
       rmse test         target           0.388191
  Predictions made & saved in "training.csv" & "test.csv"

  Plot results in "training.csv" & "test.csv"
    training, (rmse & R2) = ( 0.250 & 1.000 )
    test, (rmse & R2) = ( 0.388 & 1.000 )
    showing target
```



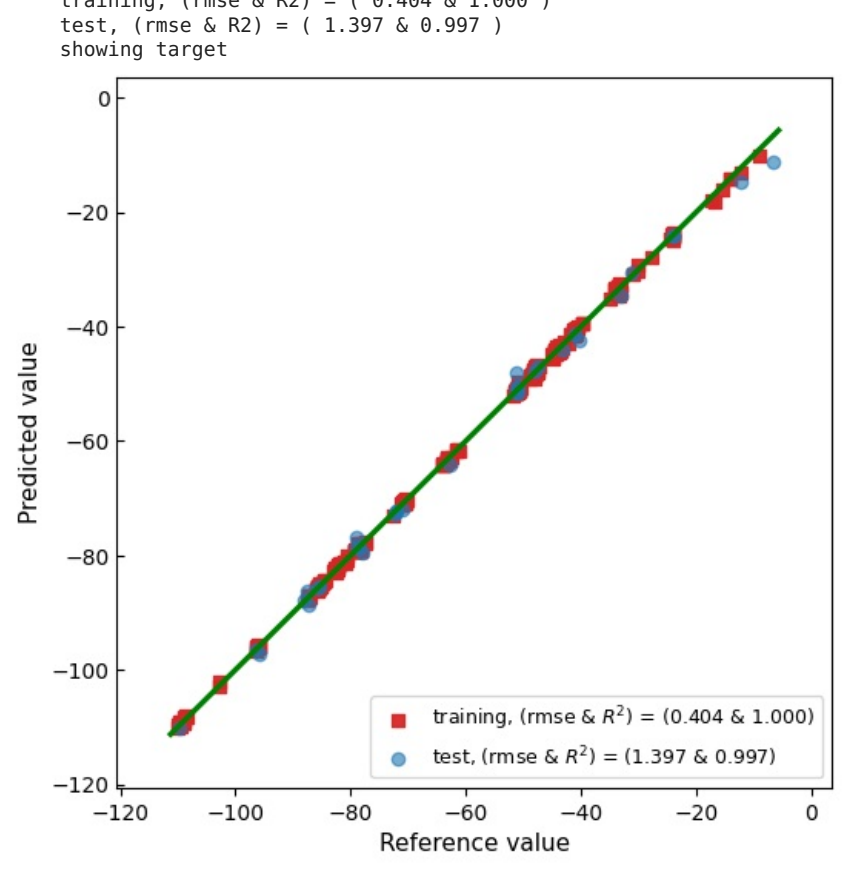
3b. KRR

In [6]:
```python
from matsml.models import KRR

# Model parameters
nfold_cv = 5                     # Number of folds for cross validation
model_file = 'model_krr.pkl'     # Name of the model file to be created
alpha = [-2, 5]
gamma = [-2, 5]
n_grids = 10
kernel = 'rbf'

model_params = {
    'kernel': kernel,
    'nfold_cv': nfold_cv,
    'model_file': model_file,
    'alpha': alpha,
    'gamma': gamma,
    'n_grids': n_grids,
}

# PCM
model = KRR(data_params=data_params_pesm, model_params=model_params)
model.train()
model.plot(pdf_output=False)

# SOAP
model = KRR(data_params=data_params_soap, model_params=model_params)
model.train()
model.plot(pdf_output=False)
```

```
  Checking parameters
    all passed                    True

  Learning fingerprinted/featured data
    algorithm                     kernel ridge regression w/ scikit-learn
    kernel                        rbf
    nfold_cv                      5
    alpha                         [-2, 5]
    gamma                         [-2, 5]
    number of alpha/gamma grids   10

  Read data
    data file                     fp_crystals_MgSi_pesm.csv
    data size                     329
    training size                 90.0  %
    test size                     10.0  %
    x dimensionality              19
    y dimensionality              1
    y label(s)                    ['target']

  Scaling x                       minmax
    xscaler saved in              xscaler.pkl

  Scaling y                       minmax

  Prepare train/test sets         random

  Building model                  KRR

  Training model w/ cross validation

  KRR model trained, now make predictions & invert scaling
    unscaling y: minmax
       rmse training     target           7.482781
    unscaling y: minmax
       rmse test         target           9.443576

  Predictions made & saved in "training.csv" & "test.csv"

  Plot results in "training.csv" & "test.csv"
    training, (rmse & R2) = ( 7.483 & 0.893 )
    test, (rmse & R2) = ( 9.444 & 0.864 )
    showing target
```



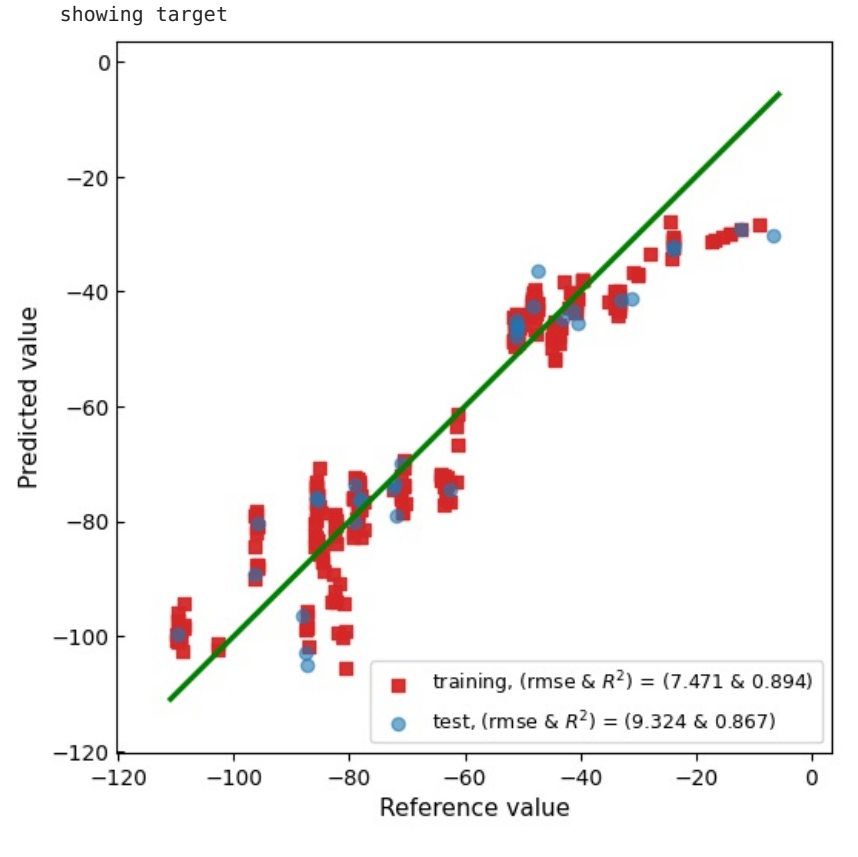

```
  Checking parameters
    all passed                    True

  Learning fingerprinted/featured data
    algorithm                     kernel ridge regression w/ scikit-learn
    kernel                        rbf
    nfold_cv                      5
    alpha                         [-2, 5]
    gamma                         [-2, 5]
    number of alpha/gamma grids   10

  Read data
    data file                     fp_crystals_MgSi_soap.csv
    data size                     329
    training size                 90.0  %
    test size                     10.0  %
    x dimensionality              735
    y dimensionality              1
    y label(s)                    ['target']

  Scaling x                       minmax
    xscaler saved in              xscaler.pkl

  Scaling y                       minmax

  Prepare train/test sets         random

  Building model                  KRR

  Training model w/ cross validation

  KRR model trained, now make predictions & invert scaling
    unscaling y: minmax
       rmse training     target           0.403504
    unscaling y: minmax
       rmse test         target           1.397075

  Predictions made & saved in "training.csv" & "test.csv"

  Plot results in "training.csv" & "test.csv"
    training, (rmse & R2) = ( 0.404 & 1.000 )
    test, (rmse & R2) = ( 1.397 & 0.997 )
    showing target
```



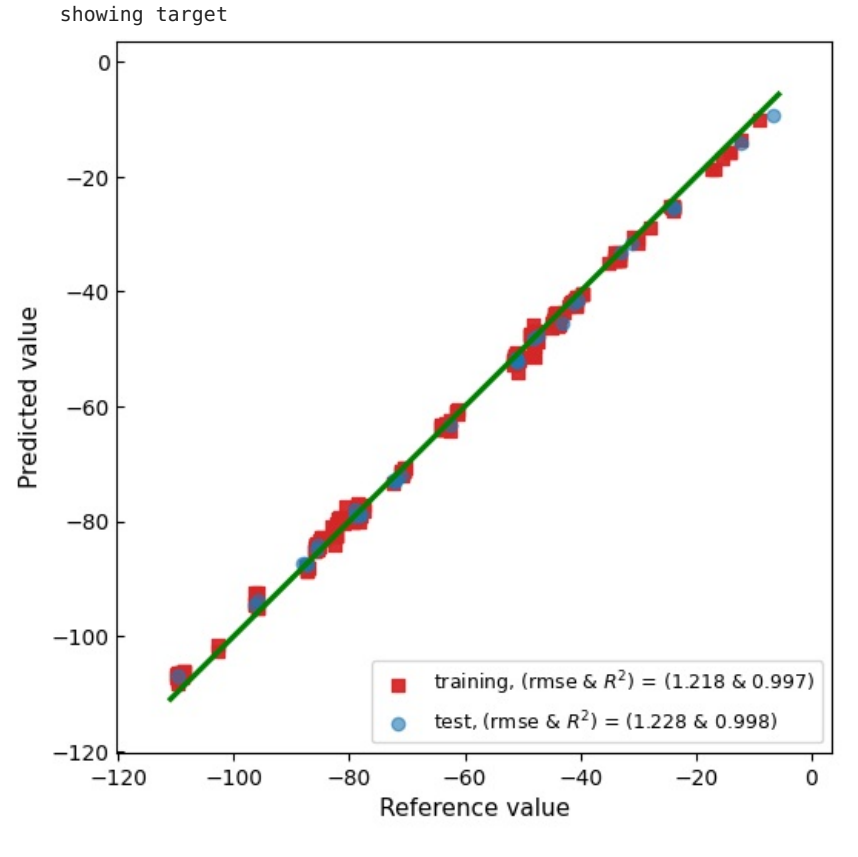
3c. GPR

In [7]:
```python
from matsml.models import GPR

# Model parameters
nfold_cv = 5                     # Number of folds for cross validation
model_file = 'model_gpr.pkl'     # Name of the model file to be created
verbosity = 0
rmse_cv = False
n_restarts_optimizer = 200

model_params = {
    'nfold_cv': nfold_cv,
    'n_restarts_optimizer': n_restarts_optimizer,
    'model_file': model_file,
    'verbosity': verbosity,
    'rmse_cv': rmse_cv,
}

# PCM
model = GPR(data_params=data_params_pesm, model_params=model_params)
model.train()
model.plot(pdf_output=False)

# SOAP
model = GPR(data_params=data_params_soap, model_params=model_params)
model.train()
model.plot(pdf_output=False)
```

```
  Checking parameters
    all passed                    True

  Learning fingerprinted/featured data
    algorithm                     gaussian process regression w/ scikit-learn
    kernel                        RBF
    nfold_cv                      5
    optimizer                     fmin_l_bfgs_b
    n_restarts_optimizer          200
    noise_lb                      0.1
    noise_ub                      10
    rmse_cv                       False

  Read data
    data file                     fp_crystals_MgSi_pesm.csv
    data size                     329
    training size                 90.0  %
    test size                     10.0  %
    x dimensionality              19
    y dimensionality              1
    y label(s)                    ['target']

  Scaling x                       minmax
    xscaler saved in              xscaler.pkl

  Scaling y                       minmax

  Prepare train/test sets         random

  Training model w/ cross validation
    cv,rmse_train,rmse_test,rmse_opt: 0 0.072537 0.074976 0.074976
    cv,rmse_train,rmse_test,rmse_opt: 1 0.071450 0.080932 0.074976
    cv,rmse_train,rmse_test,rmse_opt: 2 0.073953 0.070057 0.070057
    cv,rmse_train,rmse_test,rmse_opt: 3 0.069879 0.086898 0.070057
    cv,rmse_train,rmse_test,rmse_opt: 4 0.072047 0.078011 0.070057

  GPR model trained, now make predictions & invert scaling
    unscaling y: minmax
       rmse training     target           7.47112
    unscaling y: minmax
       rmse test         target           9.32385

  Predictions made & saved in "training.csv" & "test.csv"

  Plot results in "training.csv" & "test.csv"
    training, (rmse & R2) = ( 7.471 & 0.894 )
    test, (rmse & R2) = ( 9.324 & 0.867 )
    showing target
```




```
  Checking parameters
    all passed                    True

  Learning fingerprinted/featured data
    algorithm                     gaussian process regression w/ scikit-learn
    kernel                        RBF
    nfold_cv                      5
    optimizer                     fmin_l_bfgs_b
    n_restarts_optimizer          200
    noise_lb                      0.1
    noise_ub                      10
    rmse_cv                       False

  Read data
    data file                     fp_crystals_MgSi_soap.csv
    data size                     329
    training size                 90.0  %
    test size                     10.0  %
    x dimensionality              735
    y dimensionality              1
    y label(s)                    ['target']

  Scaling x                       minmax
    xscaler saved in              xscaler.pkl

  Scaling y                       minmax

  Prepare train/test sets         random

  Training model w/ cross validation
    cv,rmse_train,rmse_test,rmse_opt: 0 0.013691 0.013591 0.013591
    cv,rmse_train,rmse_test,rmse_opt: 1 0.013638 0.016626 0.013591
    cv,rmse_train,rmse_test,rmse_opt: 2 0.014097 0.017999 0.013591
    cv,rmse_train,rmse_test,rmse_opt: 3 0.013887 0.014546 0.013591
    cv,rmse_train,rmse_test,rmse_opt: 4 0.013202 0.015053 0.013591

  GPR model trained, now make predictions & invert scaling
    unscaling y: minmax
       rmse training     target           1.217937
    unscaling y: minmax
       rmse test         target           1.227573

  Predictions made & saved in "training.csv" & "test.csv"

  Plot results in "training.csv" & "test.csv"
    training, (rmse & R2) = ( 1.218 & 0.997 )
    test, (rmse & R2) = ( 1.228 & 0.998 )
    showing target
```



In [ ]: