SISSO

Sure Independence Screening and Sparsifying Operator

(SISSO.3.2)

User Guide

用户指南

Runhai Ouyang

Shanghai

September 1, 2022

Contents

1. Introduction

SISSO is a data-driven approach that combines symbolic regression and compressed sensing to identify models and descriptors in explicit mathematical expressions for materials science, which was originally proposed in 2018 [Phys. Rev. Mater. 2, 083802 (2018)]. Several implementations of the SISSO method can be available, and this user guide is for the SISSO FORTRAN code created and maintained by Runhai Ouyang at https://github.com/rouyang2017/SISSO. The capabilities of the SISSO code is not limited to those published in 2018, but more features within the SISSO framework have being developed and included. As of the version 3.1, the following features are available in the SISSO code:

- Regression & Classification
  [R. Ouyang, S. Curtarolo, E. Ahmetcik, M. Scheffler, L. M. Ghiringhelli, Phys. Rev. Mater. 2, 083802 (2018)]
- Multi-task learning
  [R. Ouyang, E. Ahmetcik, C. Carbogno, M. Scheffler, L. M. Ghiringhelli, J. Phys.: Mater. 2, 024002 (2019)]
- Variable selection assisted symbolic regression
  [Z. Guo, S. Hu, Z.-K. Han, R. Ouyang, J. Chem. Theory Comput. 18, 4945 (2022)]
- Sign-constrained multi-task learning
  [J. Wang, H. Xie, Y. Wang, R. Ouyang, submitted]

2. Installation

A FORTRAN MPI compiler is required to compile the SISSO parallel program. Below are two options for compiling the program using an IntelMPI compiler (other compilers may work as well but the Intel compilers are recommended). Download and unzip the package in Linux operating systems, and then go to the folder 'src' and do:

1) mpiifort -fp-model precise var_global.f90 libsisso.f90 DI.f90 FC.f90 SISSO.f90 -o ~/bin/SISSO

or

2) mpiifort -O2 var_global.f90 libsisso.f90 DI.f90 FC.f90 SISSO.f90 -o ~/bin/SISSO

Note: option (1) enables better accuracy and run-to-run reproducibility of floating-point calculations; (2) is ~ 2X faster than (1) but tiny run-to-run variations may happen between processors of different types, e.g. Intel and AMD.

If 'mpi' related errors present during the compilation, try opening the file 'var_global.f90' and replace the line "use mpi" with "include 'mpif.h'". However, " use mpi " is strongly encouraged (see https://www.mpi-forum.org/docs/mpi-3.1/mpi31-report/node411.htm).

Modules of the program:
var_global.f90 ! declaring the global variables.
libsisso.f90 ! subroutines and functions for mathematical operations.
DI.f90 ! model sparsification (descriptor identification).
FC.f90 ! feature construction.
SISSO.f90 ! the main program.

3. Running SISSO

To run SISSO for regression, classification, multi-task learning, sign-constrained multi-task learning, the two files, SISSO.in and train.dat, have to be present in the working directory. Their templates can be found in the folder “input_templates”. The simple command-line usage is:

```
SISSO > log
```

You may need to remove resource limit by running the command 'ulimit -s unlimited'. However, you are recommended to run the code on clusters, as the SISSO jobs often require many CPU cores and much memory. For example, use this command in your submission script:

mpirun -np number_of_cores SISSO > log

To run the VS-SISSO, the additional VarSelect.py program has to be present (can be copied from the folder “utilities”) as well in the working directory. Set the parameters in VarSelect.py for your jobs and run the VS-SISSO by:

Python3 VarSelect.py

4. Input and output files

4.1 Input files

SISSO.in

The file SISSO.in contains most of the control parameters for users to define the SISSO job on the information of the type of machine learning (regression, classification, multi-task learning, sign-constrained multi-task learning), model complexity, number of data and input features, sparsifying operators and so on. The code reads the file SISSO.in line by line to detect the keywords and corresponding values after the “=”. Blank line, space, and comments starting with the exclamation mark (!) will be ignored. Users can find the template for the SISSO.in in the folder “input_templates”. Instructions on how to set the control parameters are described in the Section 5.

train.dat

There are several templates available for the train.dat file according to the specific machine learning job, including the train.dat_regression, train.dat_classification, train.dat_regression_multitask, and train.dat_classification_multitask in the folder “input_templates”. That is, users can take the template corresponding the SISSO job defined in the SISSO.in to input your training data, and then rename the file as train.dat before running SISSO.

In train.dat_regression, the first line are the strings to explain/name each column. Apart from the first line, each of the following lines represent one data, with the first entry in this line being a string to explain this data, e.g. compound name, the second entry of this line being the value of the target property of this compound, and the entries since the third are the values for each feature. Note that any string should **avoid using the special characters** of space, comma, Tab, and mathematical operators.

In train.dat_classification, the format is the same as in the train.dat_regression, except that

the second column for the target property is removed. No property values are needed for classification. SISSO only require the category information of the dataset. For example, supposing there are three categories for the target property, with 50, 60, 70 data for category A, B, and C, respectively, then one will set “nsample = (50, 60, 70)” in the SISSO.in. Correspondingly, you need to sequentially arrange the three groups of data in the train.data, i.e. the 50 data from line 2 to 51, the 60 data from line 52 to line 111, and the 70 data from line 112 to line 181.

In train.dat_regression_multitask, and train.dat_classification_multitask, the formats are the same with that of train.dat_regression and train.dat_classification, respectively, except that the data are sequentially arranged task by task. For example, supposing there are three tasks in regression, with 100, 200, 300 data for task 1, 2, and 3, respectively, then one will set “nsample = 100, 200, 300” in the SISSO.in. Correspondingly, you need to arrange the 100 data from line 2 to line 101, the 200 data from line 102 to line 301, and the 300 data from line 302 to line 601. In multi-task classification, supposing there are two tasks, with the data (50, 60, 70) for the first task and (80, 90, 70) for the second task, then one will set “nsample = (50, 60, 70), (80, 90, 70)” in the SISSO.in, and arrange the (50, 60, 70) data from line 2 to line 181, and the (80, 90, 70) data from line 182 to line 421.

4.2 Output files

SISSO.out

The information of control parameters, feature spaces, and the identified best model are output to the file SISSO.out. For a SISSO-nD calculation (the model dimension is set to n in SISSO.in), besides of the nD model, the mD models ($1 \leq \mathrm{m} < \mathrm{n}$) are also created in the SISSO.out because they are needed to generate the nD model. However, these mD models from the SISSO-nD calculation are often less accurate than those from calculations directly for the mD. The reason is that the computationally affordable size of the SIS-subspace for a SISSO-mD job can be far greater than that in the SISSO-nD job. Unless the convergence of the model accuracy against the size is reached, increasing the size is necessary to approach the solutions of global minimum. Therefore, it is recommended to perform a SISSO-xD calculation if you would like to obtain a xD model, with a large SIS-subspace size for the xD. More details about this information can be found in the Section 6.

For regression, the functional form of a nD model is $\mathrm{y} = \sum_1^n \beta_i d_i + \beta_0$, where the $\{d_i\}$ is termed the nD descriptor, $\{\beta_i\}$ are the corresponding coefficients, and β_0 is the intercept. The number before each feature ($[d_i]$) in the SISSO.out is the line number of this feature in the file Uspace.expressions in the folder SIS_subspaces (see introduction below). In multi-task learning regression, all the tasks share the same nD descriptor, but each task have a set of different coefficients and the intercept.

For classification, there are only descriptors but no coefficients in the SISSO.out. A nD

descriptor means a n-dimensional map. Currently, up to 3D are implemented for classification, yet the convex-hull construction for 3D is not stable and bugs remain to be fixed. As described in Section 6.2, the classification error is defined as the number of data in all overlap regions, and the overlap volume as the second metric. If there is no any overlap between all domains, the overlap volume is negative, whose magnitude indicating the smallest distance between the domains. With the identified nD descriptor, one could straightforwardly apply support vector classification to find the separating hyperplanes, or decision tree to find the separating rules.

models

This folder contains the files for the information of the top n models/descriptors, where the n can be set in SISSO.in. The features in each descriptor are indicated by their "Feature_ID" which is the line number in the file Uspace.expressions in the folder SIS_subspaces. For regression, additional files for the coefficients are provided, for example, (c_i,i=0,n)_j,j=1,ntask, where c_0 is the intercept, and c_1 to c_n are the coefficients corresponding to d_1 to d_n in the linear model $\mathrm{y} = \sum_1^n c_i d_i + c_0$.

desc_dat

The folder "desc_dat" contains the data files of the identified descriptors shown in the SISSO.out. The values of y_true (from training data), y_pred (from the model fitting), and the descriptor for each compound are provided. The files are named "desc_nd_pm.dat", denoting the descriptor of nD for the mth property (task). m = 1 for usual learning with one property, and m > 1 for multi-task learning.

SIS_subspaces

There are n SIS-subspaces in a SISSO-nD calculation. These subspaces, including the feature expressions and feature data, are kept in the folder "SIS_subspaces". The file "Uspace.expressions" and "Uspace_p001.dat" contain the feature expressions and feature data of the union of all these subspaces, respectively. For example, the features in Uspace.expressions from line 1 to N have their corresponding data in Uspace_p001.dat from column 2 to N+1 for regression (the target property occupies the first column), and from 1 to N for classification. The "p001" in the Uspace_p001.dat denote the property 1 (task 1). The features in the Uspace.expressions are ranked according to the SIS score, i.e. the correlation between each feature and the target property.

convex$n$d_hull

In classification, each category domain is approximated by the convex volume of the data

group. The file convex$n$d_hull contains the information to construct the nD volume for each domain in the space of the identified best descriptor.

log

The log file (SISSO > log) has the progress information of a SISSO job. One can estimate how much time are required to finish the job based on the time spent to reach certain progress, e.g. 20%.

VS_results

If one is performing the VS-SISSO, then the results are output to the file VS_results which contains the information of the model error and selected variables and more details from each iteration.

5. Control parameters

5.1 SISSO.in

ptype

ptype = 1 or 2. The type of the target property. 1: regression with continuous properties, 2: classification with categorical properties.

ntask

(R&C). ntask = integer. Number of tasks. ntask = 1: the usual machine learning with single task, ntask >1: multi-task learning (MTL). The R&C indicates that this keyword is used by both regression and classification.

task_weighting

(R). task_weighting = 1 or 2. The weighting for MTL regression. 1: no weighting (all tasks treated equally regardless of their respective dataset size), 2: each task is weighted by the number of data of this task divided by the total number of data of all tasks.

scmt

(R). scmt = .true. or .false. Sign-Constrained MTL for regression is invoked if .true.

desc_dim

(R&C). desc_dim = integer. Dimension of the descriptor or model, usually from one to five. The higher the dimension, the smaller the computationally affordable size of SIS-subspace. If the size is set to 1 in each iteration, SISSO becomes the method OMP.

nsample

(R) nsample = integer, integer, …;
(C) nsample = (integer, integer, …), (integer, integer, …), …;
Number of samples in the train.dat. For regression, if the input is one integer, it means single-task learning; if there are multiple input integers separated by comma, it means an MTL regression job, with each integer being the number of data for the corresponding task. For classification, if there is only one round bracket, it means single-task classification, with each integer being the number of data for the corresponding category; if there are multiple round brackets separated by comma, it means an MTL classification job, with each bracket describing the data information for the corresponding task.

restart

(R&C). restart = 0 or 1. 0: starts the job from scratch, 1: continues the job (the progress information of last stop was stored in the file CONTINUE)

nsf

(R&C). nsf = integer. Number of scalar features provided in the file train.dat. The “scalar feature” means each feature is one column in the dataset, or one number for one material.

ops

(R&C). ops = ‘(operator)(operator)(operator)…’. Users can customize the operators from the list: {+, -, *, /, exp, exp-, ^-1, ^2, ^3, sqrt, cbrt, log, |-|, scd, ^6, sin, cos}, where the “exp-”, “cbrt”, “|-|”, “scd” means exp(-x), cubic root, |x1-x2|, and standard Cauchy distribution, respectively.

fcomplexity

(R&C). fcomplexity = integer. Maximal feature complexity, usually from 0 to 7. For example, with fcomplexity=3, all the features with complexity from 0 up to 3 are generated. All the input variables and the expressions generated in SISSO by mathematical combination between the input variables are all termed features. The feature complexity is defined as the number of mathematical operators in a feature. Thus, 0 means the input variable that are without any mathematical combinations in SISSO. Greater than 7 will lead to extremely huge feature space (tier > 3, see section 6.1) that often makes it computationally infeasible.

funit

(R&C). funit=(integer:integer)(integer:integer)… Features with the same units are grouped together in the train.dat. For example, features from the first to the 5th have energy dimension with unit eV, features from the 6th to the 9th have the length dimension with unit angstrom, the 10th feature is dimensionless, and the 11th feature has another dimension. Correspondingly, in SISSO.in one can set funit = (1:5)(6:9)(11:11). Features not included in the funit are treated as dimensionless. The purpose is to enable SISSO perform dimension analysis and avoid unreasonable combinations such as “energy + length” and “energy - length”.

The unit of each feature is represented by a jD vector, where j is the number of unit type (number of brackets from the input for **funit**). If there are 6 features in train.dat, with their units in eV, eV, eV, Å, Å, Å, and in SISSO.in the funit = (1:3)(4:6), then the feature units are represented by
feature1: 1 0
feature2: 1 0
feature3: 1 0
feature4: 0 1
feature5: 0 1
feature6: 0 1
The plus “+” and minus “-” only apply to two features with a same unit vector. The unit vector for the new feature generated by mathematical operations on existing features is defined as:
feature1 + feature2, feature1 – feature2 → new_features: v = v1 (or v2)
feature1*feature2 → new_feature: v = v1 + v2
feature1/feature2 → new_feature: v = v1 -v2
log(feature1), exp(feature1), sin(feature1), cos(feature1) → new_features: v = 0
(feature1)^n → v = v1*n

SISSO also accept the input of units from a file named “feature_unit”, if more flexible setting is needed for the units of the input features. For example, if the units of the 6 input features are in eV, $\text{eV}^{1/2}$, eV, Å^2, Å, Å, then user can prepare the file in the working directory with the following content:

1 0
0.5 0
1 0
0 2
0 1
0 1

fmax_min

(R&C). fmax_min = float. The threshold that if the maximal absolute value of the data in a feature is smaller than fmax_min, it means the magnitude of all values in the feature are so small that the feature will be treated as zero-feature in SISSO and will be discarded.

fmax_max

(R&C). fmax_max = float. The threshold that if the maximal absolute value of the data in a feature is smaller than fmax_min, it means the magnitude of all values in the feature are so small that the feature will be treated as infinity-feature in SISSO and will be discarded.

nf_sis

(R&C). nf_sis = integer (or integer_1, integer_2, …, interger_desc_dim). The size of the SIS-subspaces. There will be n subspaces for a SISSO-nD calculation. If there is only one input integer, then all the n subspaces have the same size. Otherwise, users can input n integers, separated by comma, for the size of each subspace.

method_so

('L0' for R&C, 'L1L0' for R only). method_so = 'L0' or 'L1L0'. 'L0' is the ℓ_0-norm minimization sparsifying method. 'L1L0' is the method proposed in the paper [L. M. Ghiringhelli et al., PRL 114, 105503 (2015)], i.e. screening certain number of features, says 30, first by using LASSO, followed with ℓ_0 to identify the best 1D, 2D, 3D, … models/descriptors. 'L0' is recommended for its accuracy over the 'L1L0' for low-dimensional models.

nl1l0

(R). nl1l0 = integer. The number of features selected by LASSO. Other control parameters for the LASSO are not explicitly included in the SISSO.in template. Important parameters are in the following default values (they can be changed by including them in the SISSO.in):

$L1\_max\_iter = 1e6$

The maximal number of iterations.

L1_dens=120
Number of points to sample the penalty parameter λ. The λ_i take values by using the expression: log10(λ_i) = log10(λ_{max}) – (i-1)[log10(λ_{max}) - log10(λ_{min})]/L1_dens. Here, the λ_{max} is the smallest λ for which all the coefficients are zero [J. Friedman, T. Hastie, R. Tibshirani, J. Stat. Softw. 33, (2010)], and the λ_{min} is defined as 0.001∗ λ_{max}.*

L1_tole =1e-6
The tolerance for the optimization with a given λ.

L1_minrmse=1e-3
The condition for LASSO to stop further trial of new λ if the model RMSE < L1_minrmse

L1_warm_start = .true.
Using the solution of last λ to start the next optimization with a new λ.

fit_intercept

(R). fit_intercept = .true. or .false. Fit to a nonzero / zero intercept for the linear model.

metric

(R). metric = 'RMSE' or 'MaxAE'. The metric for model selection in regression: RMSE (root-mean-squared error) or MaxAE (max absolute error).

nmodels

(R&C). nmodels = integer. The top n models to output (see the folder 'models').

isconvex

(C). isconvex = (k,k,k,…). Each domain can be restricted to be convex of nonconvex. k takes the value 1 or 0, 1: YES; 0: NO.

bwidth

(C). bwidth = float. The boundary tolerance for each domain to include the data that are outside but very close to the domain.

5.2 VarSelect.py

If one is using the VarSelect.py to do the VS-SISSO, then additional control parameters can be found in the VarSelect.py. For details of the VS-SISSO algorithm, please refer to [J. Chem. Theory Comput. 18, 4945 (2022)] or section 6.3.

n_init = integer # Initial size for the S.

n_RS = integer # The size of the S_a.

n_max = integer # Maximal size of the S $(S = S_a + S_b)$.

nstep_max = integer # maximal iterations

nstep_converge = integer # stop if the best model unchanged for certain steps.

restart = 0 or 1 # 0: start from scratch, 1: continue the job

runSISSO = '*command-to-run-SISSO***'** # E.g. 'mpirun -np 64 SISSO > slog'.

6. Algorithms

6.1 Regression

As described in [Phys. Rev. Mater. 2, 083802 (2018)], the process of SISSO mainly comprises of two steps. First, feature construction. With the input variables and mathematical operators, SISSO generates all the possible meaningful expressions within prescribed complexity. Here, both the variables and expressions are all termed features. The feature complexity is defined as the number of operators in a feature. Thus, all the input variables have complexity 0, and a feature such as $x + y/z$ has complexity 2. All these features form a space, termed feature space.

Given the set of operators $\hat{\boldsymbol{H}}$ (e.g. $\hat{\boldsymbol{H}} = \{+, -, \times, \div, \sqrt{\ }, \exp(\ ), \log(\ ), ^2, ...\}$) and initial feature space $\boldsymbol{\Phi}_0$ which contains all the input variables, the feature space of tier T is defined as

$$\boldsymbol{\Phi}_T \equiv \boldsymbol{\Phi}_{T-1} \bigcup \hat{\boldsymbol{H}}[\varphi_1, \varphi_2], \qquad \forall \varphi_1, \varphi_2 \in \boldsymbol{\Phi}_{T-1}, \tag{1}$$

where φ_1, φ_2 are features in $\boldsymbol{\Phi}_{T-1}$ (for unary operators only φ_1 is considered). The average "+" and difference "−" operators apply to two features with the same unit defined by

users. The size of $\boldsymbol{\Phi}_T$ roughly increases with $(\#\boldsymbol{\Phi}_0)^{2^T} \times (\#\widehat{\boldsymbol{H}}_2)^{2^T-1}$, where $\#\boldsymbol{\Phi}_0$ and $\#\widehat{\boldsymbol{H}}_2$ are the numbers of elements and binary operators in $\boldsymbol{\Phi}_0$ and $\widehat{\boldsymbol{H}}_2$, respectively. Thus, the size of feature space $\boldsymbol{\Phi}$ expands rapidly with the T and number of input features in $\boldsymbol{\Phi}_0$.

Second, descriptor identification. the feature space $\boldsymbol{\Phi} = \{\varphi_i\}$ is assumed to be a basis set to expand the target property $P = \sum_i^I \beta_i \varphi_i$, and the best few features $\{\varphi_k\}$ that $P \approx \sum_k^K \beta_k \varphi_k$, $K \ll I$, are identified via sparse regression. The $\{\varphi_k\}$ is called a K-dimension descriptor. The dimension and feature complexity are the two aspects defining the overall complexity of the model. Whether accurate and low-dimensional models/descriptors can be identified is critically dependent on the quality of the feature space, and hence the input features, in addition to the reliability of the dataset.

More detailly, the descriptor identification algorithm is described in Figure 1. The model/descriptor dimension is a hyperparameter that users will test progressively from low to high until the leftover residual error is within quality expectation. For a given dimension n, SISSO will perform n iterations of the process of "feature space construction → subspace selection via sure independence screening (SIS) → descriptor identification via a sparsifying operator (SO)" process. In SIS, all the features of the whole feature space are firstly standardized and ranked by $|< P, \varphi_i >|$, and then the top $\#S_{1D}$ features are selected into the subspace S_{1D}. Note the standardization is only performed in SIS for the purpose of feature ranking, but not for feature construction and model building. Construction of the feature space is repeated in each iteration because the space is too large to be stored in the computer memory, and the selection of the SIS-subspace is performed on the fly. n iterations are required because n SIS-subspaces are needed to find the nD descriptor. Each SIS-subspace is selected based on the absolute of the inner product $|< \Delta, \varphi_i >|$, where the Δ is the residual error from previous iteration (The Δ is the P for the first iteration).

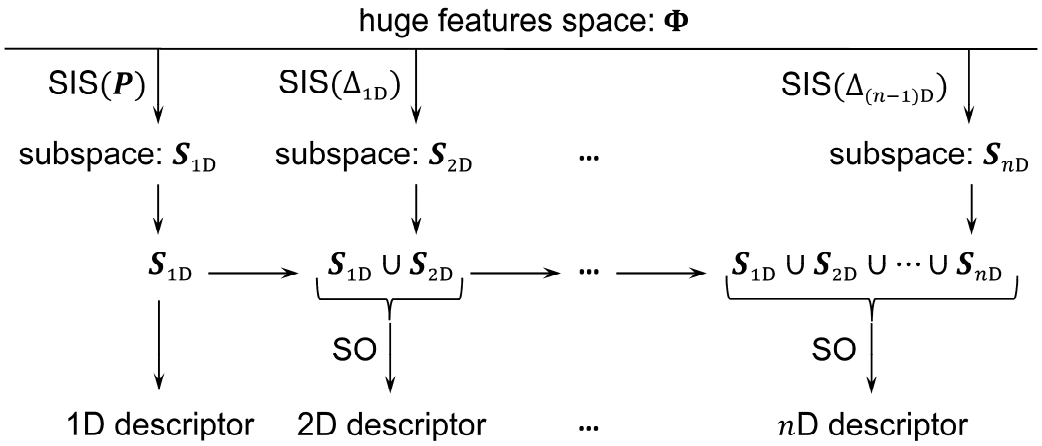

Figure 1. The SISSO algorithm.

For the sparse regression, ℓ_0-norm minimization is often adopted as the SO in SISSO:

$$\widehat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\text{arg}min}(||\boldsymbol{P} - \boldsymbol{\Phi\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_0), \quad (2)$$

where the λ is a tuning hyperparameter (penalty parameter) to control the model sparsity (dimension). Though the ℓ_0 -norm minimization is a NP-hard problem, it remains computationally feasible if one focus on low-dimensional solutions, e.g. < 5D, with reasonable SIS-subspace size. For example, if $n = 2$, the size of each SIS-subspace can be as large as 250,000 (union of all SIS-subspaces: 250,000*2=500,000), with few hours of runtime using ~ 100 CPU cores. The total number of evaluated models by ℓ_0 is thus C(500000,2) = $1.2*10^{11}$. Supposing $1.2*10^{11}$ is the computationally allowed largest number, if $n = 3, 4, 5$, then the corresponding size of each SIS-subspace is about 3000, 330, 90, respectively. **For this reason users should set corresponding SIS-subspace size for different $\boldsymbol{n}$.** In principle, the larger the SIS-subspace, the higher probability of identifying the best models before the convergence is reached. In case $n > 10$ when each SIS-subspace can accept only 1 feature because of the limitation of usual computing power, SISSO becomes the orthogonal matching pursuit (OMP) method. Therefore, SISSO is a method in between the SO (SIS-subspace equal the total feature space) and the OMP (each SIS-subspace contains only 1 feature). In addition to ℓ_0, LASSO is also available as the SO for its computational efficiency, unfortunately it often yields poor results when combining with SIS.

6.2 Classification

In classification, the SISSO framework remains the same, but the definition for the loss function and the SIS are modified. In the space of descriptors, each category's domain is approximated as the volume (area, in two dimensions) within the convex hull of the corresponding training data, as shown in Figure 2. Given a property with M categories, the error $||\boldsymbol{P} - \boldsymbol{\Phi\beta}||_2^2$ in regression is replaced by the total overlap between all the M convex volumes. The ℓ_0-norm minimization for classification is defined as:

$$\hat{\boldsymbol{c}} = \underset{\boldsymbol{c}}{\text{arg}min}\left(\sum_{i=1}^{M-1}\sum_{j=i+1}^{M} O_{ij} + \lambda||\boldsymbol{c}||_0\right), \quad (3)$$

where O_{ij} is the number of data in the overlap region between the i and j domains, $\boldsymbol{c}$ is a sparse vector (0/1 elements) so that a feature k is selected (deselected) when c_k = 1(0), and λ is a tuning hyperparameter controlling the number of nonzero elements in $\boldsymbol{c}$.

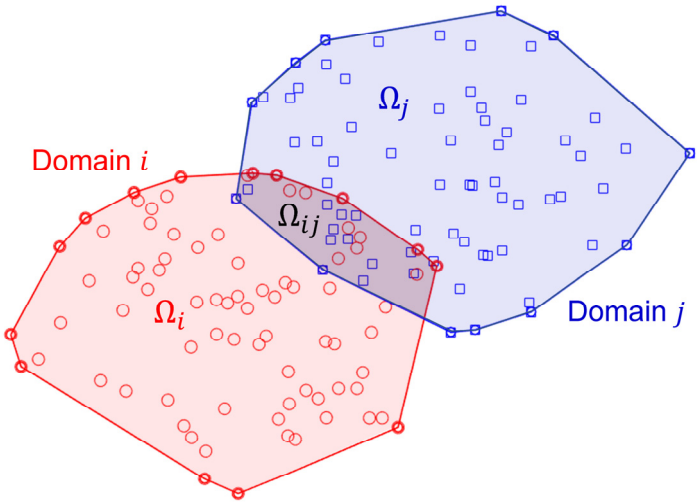

Figure 2. The domains for each dataset are described by convex volume. The overlap area is indicated by the Ω_{ij}.

The solutions to Equation (3) may not be unique. Of all the possible solutions of Equation (3) having the same dimension and overlap, the one with minimal n-dimensional overlap volume is selected:

$$\Omega \equiv \frac{2}{M(M-1)} \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} \frac{\Omega_{ij}}{\min(\Omega_i, \Omega_j)}, \tag{4}$$

where Ω_i, Ω_j, and Ω_{ij} are the n-dimensional volumes of the i, j, and overlap ij domains. Ω has values from 0 to 1. For example, with $M = 2$, if one domain is totally included in another domain, the $\Omega = 1$; if two domains are separated, $\Omega = 0$.

In SIS, the correlation between the property and a feature $|< P, \varphi_i >|$ for regression is now replaced by the $\left(\sum_{i=1}^{M-1} \sum_{j=i+1}^{M} O_{ij} + 1\right)^{-1}$ for classification so that the higher the correlation, the lower the overlap between datasets in the one-dimensional descriptor space. With the above new definitions for SIS and SO, SISSO can be applied to find the best descriptor that all the domains have the largest separations.

Note that above definition of error based on minimization of domain overlap is different from usual classification error based on separating hyperplanes or rules. Thus, the best classification model from SISSO may differ from the best from support vector classification or decision tree because of their different metrics for model selection. However, with a given nD descriptor from SISSO, one can straightforwardly find the corresponding separating hyperplanes and rules by using support vector classification and decision tree, respectively.

6.3 Variable selection assisted symbolic regression

As discussed in section **1.1**, the size of feature space $\boldsymbol{\Phi}_T$ roughly increase with $(\#\boldsymbol{\Phi}_0)^{2^T}$. Thus, increasing the number of input features will lead to rapid growth of the $\boldsymbol{\Phi}_T$. In SISSO, $\boldsymbol{\Phi}_{T-1}$ need to be stored in memory in order to generate the $\boldsymbol{\Phi}_T$. To ensure fast computation and better scaling with the number of processors, the parallelization in SISSO adopts the distributed memory system. Thus, the required memory to store the matrix of a feature space with M features and N data points in double precision, running on P CPU cores, is approximately $M \times N \times P \times 8/10^9$ GB. In addition, each processor requires memory to store a SIS-subspace. Therefore, the P ($\boldsymbol{\Phi}_{T-1}$+SIS-subspace) are responsible for the primary memory demand in a SISSO job. To generate $\boldsymbol{\Phi}_3$ (feature complexity up to 7), with available memory 256 GB, hundreds of training data, the largest size of $\boldsymbol{\Phi}_0$ is often limited to be less than 30.

To remove the limit on the size of $\boldsymbol{\Phi}_0$ and avoid huge demand of memory, a new algorithm named VS-SISSO that integrates SISSO with iterative variable selection (VS) was developed [J. Chem. Theory Comput. 18, 4945 (2022)]. The idea of VS-SISSO is shown in Figure 3, and the five steps are described in text as below.

1) Obtaining a subset Sa from the large pool of input features by random search, which has priority on unvisited features before all are visited once.
2) Let S being the union of Sa and Sb, where Sb (initially empty) is the set of variables appearing in the best model so far from symbolic regression.
3) Perform symbolic regression (SISSO) with variables in S to update the model, which has fixed complexity in all iterations.
4) Accept and update the Sb if the training accuracy of the model is improved.
5) Stop if the accuracy converges or the iteration hits maximum; otherwise, go to step 1.

Performance of the VS-SISSO can be found in [J. Chem. Theory Comput. 18, 4945 (2022)].

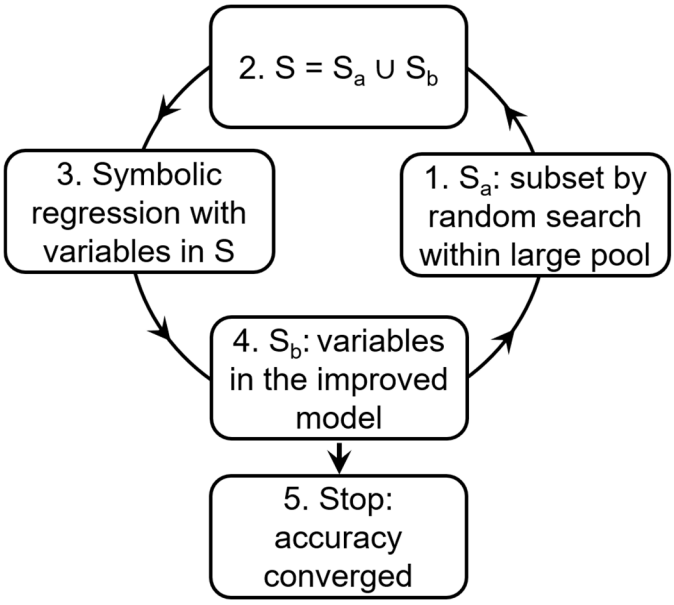

Figure 3. The VS-SISSO algorithm.

6.4 Multi-task learning

Multi-task learning within the framework of SISSO (MT-SISSO) was designed to identify a common n-dimensional descriptor for several related properties [J. Phys.: Mater. 2, 024002 (2019)]. Given T properties, each of which can be expanded in the same feature space $\{\varphi_i\}$, $P^t = \sum_i^N \beta_i^t \varphi_i$, the outcome of MT-SISSO is a common sparse model, says 2D, $P^t = \beta_1^t \varphi_1 + \beta_2^t \varphi_2$. The coefficients $\{\beta_1^t, \beta_2^t\}$ are different for different t (task), but the 2D descriptor $\{\varphi_1, \varphi_2\}$ is the same for all the tasks. With the feature space $\boldsymbol{\Phi} = \{\varphi_i\}$ and sparsifying operator ℓ_0, the problem of multi-task learning for regression can be described as:

$$\widehat{\boldsymbol{B}} = \underset{\boldsymbol{B}}{\operatorname{arg}min} \sum_{t=1}^{T} \frac{1}{N^t} (||\boldsymbol{P}^t - \boldsymbol{\Phi}\boldsymbol{\beta}^t||_2^2 + \lambda ||\boldsymbol{B}||_0), \tag{5}$$

where $\boldsymbol{\Phi}$ is the same for all the tasks, $\boldsymbol{B}$ is the coefficients matrix in which each column is a coefficients vector $\boldsymbol{\beta^t}$, and N^t is the number of data in the tth property $\boldsymbol{P}^t$. In Equation (5), all the tasks are treated equally, regardless of the dataset size for each task. The corresponding model error is defined as $\mathrm{RMSE} = \sqrt{\sum RMSE_t \,/T}$, where $RMSE_t$ is the root-mean-squared error of task t. Equation (5) can be slightly modified so that the tasks are weighted according to their dataset size:

$$\widehat{\boldsymbol{B}} = \underset{\boldsymbol{B}}{\operatorname{arg}min} \sum_{t=1}^{T} \frac{N^t}{\sum N^t} (||\boldsymbol{P}^t - \boldsymbol{\Phi}\boldsymbol{\beta}^t||_2^2 + \lambda ||\boldsymbol{B}||_0), \tag{6}$$

Corresponding model error is then revised to $\mathrm{RMSE} = \sqrt{\sum RMSE_t{}^2 \cdot N^t / \sum N^t}$.

In the SIS step of MT-SISSO, the correlation of each feature with the target property takes the form of quadratic mean of the $|< P^t, \varphi_i >|$ over all the tasks:

$$\theta_i = \sqrt{\sum_{t=1}^{T} < P^t, \varphi_i >^2 / T}. \tag{7}$$

SIS ranks the features according to θ_i, and selects the features with high θ_i to form subspaces.

Multi-task learning can also be applied to classification problems [J. Phys.: Mater. 2, 024002 (2019)]. Given T properties (tasks), the goal of multi-task learning is to find a common descriptor for T maps, on each of which M^t domains are separated. The best model can be obtained by minimizing the overlap between domains:

$$\widehat{\boldsymbol{C}} = \underset{\boldsymbol{C}}{\mathrm{argmin}} \left(\sum_{t=1}^{T} \sum_{i=1}^{M^t - 1} \sum_{j=i+1}^{M^t} O_{ij} + \lambda ||\boldsymbol{C}||_0 \right). \tag{8}$$

Accordingly, the metric for SIS to evaluate the importance of a feature is redefined as:

$$\theta_i = \left(\sum_{t=1}^{T} \sum_{i=1}^{M^t - 1} \sum_{j=i+1}^{M^t} O_{ij} + 1 \right)^{-1}. \tag{9}$$

6.5 Sign-constrained multi-task learning

In multi-task learning, the target properties of different tasks can be the same physical quantity, and they are distinguished as different tasks because of some reasons, says the datasets are inconsistent because they come from different experiments. In this case, the goal of multi-task learning is to identify a common descriptor for the same quantity but different datasets. However, when the sign of coefficients between different tasks are different, the model becomes confusing. For example, considering the two-task 2D model $P^t = \beta_1^t \varphi_1 + \beta_2^t \varphi_2$, the first coefficient can be $\beta_1^1 < 0$ and $\beta_1^2 > 0$ for task 1 and 2, respectively. In this situation, if the P^1 and P^2 represent the same physical quantity, then it means that the dependency of the quantity P on the same feature φ_1 is opposite between the two tasks. This could happen when the model is overfitting or one of the two datasets is unreliable. To avoid this difficulty, a sign constraint can be added to the Equation (5) as:

$$\widehat{\boldsymbol{B}} = \underset{\boldsymbol{B}}{\operatorname{argmin}} \sum_{t=1}^{T} \frac{1}{N^t} \|\boldsymbol{P}^t - \boldsymbol{\Phi}\boldsymbol{\beta}^t\|_2^2 + \lambda\|\boldsymbol{B}\|_0 \quad (10)$$

$$\text{subject to: } \operatorname{sgn}\left(\beta_j^t \cdot \beta_j^{t'}\right) \geq 0 \quad \forall\, t, t',$$

where the sgn is the sign function. As shown in the paper [J. Wang et al., submitted], inclusion of the sign-constraint only causes slightly increase of the training errors for high-complexity models, but the models are made more resonable with the coefficients between all the tasks having the same sign. This method is termed SCMT-SISSO.

To find the solution of Equation (10), all possible signs in the linear Equation will be tested, and the one with lowest model error will be selected. For example, in the 2D model $P^t = \beta_1^t\varphi_1 + \beta_2^t\varphi_2$, the models with ($\beta_1^t < 0,\ \beta_2^t < 0, \forall$t), ($\beta_1^t < 0,\ \beta_2^t > 0, \forall$t), ($\beta_1^t > 0,\ \beta_2^t < 0, \forall$t), and ($\beta_1^t > 0,\ \beta_2^t > 0, \forall$t) will all be calculated, with sign-constrained coefficients optimization problem solved by using the coordinate descent algorithm. As a result, the time cost for the sparse regression in SCMT-SISSO will be 4 times more than that in MT-SISSO for the 2D model.

7. Utilities

Useful tools for data processing and results analysis are provided in the folder “utilities”.

k-fold-cv.f90

With the given train.dat and SISSO.in, run the k-fold-cv to create k folders, in each of which a new train.dat (including k-1 out of the k parts of the total training data), predict.dat (the remaining 1 of the k parts), and corresponding adjusted SISSO.in are generated. Usage: (1) ifort k-fold-cv.f90 -o k-fold-cv (2) ./k-fold-cv (3) check the data and submit the k jobs.

SISSO_predict.f90

With the given SISSO.out and predict.dat, run SISSO_predict to read the models from the SISSO.out and predict the unknown data in the file predict.dat. The predict.dat has the same format with that of train.dat. The column for the data of target property is needed so that the prediction RMSE can be calculated. If the data of the target property for these test materials are unknown, user can simply set them to zero, and ignore the calculated RMSE by SISSO_predict. The SISSO_predict requires parameters from either interactive input or a file named SISSO_predict_para that contains those parameters. Usage: (1) ifort SISSO_predict.f90 -o SISSO.predict (2) ./SISSO_predict.

VarSelect.py

The python program for running the VS-SISSO as mentioned earlier.

SVC.py

The python program of support vector classification for creating the separating hyperplanes based on the SISSO-identified descriptor, i.e. the file train.dat containing the n-features of the nD descriptor. Usage: python SVC.py >out.

af2traindat.f90

Generating the train.dat for compounds based on a table of atomic features. Running this program requires the presence of the file atom_features and samplelist (templates available in the “utilities”). Usage: (1) ifort af2traindat.f90 -o af2traindat (2) ./af2traindat.

Ionic_Radii

The extended dataset of Shannon radii from the work [Chem. Mater. 32, 595 (2020)].