Full Paper

www.molinf.com

molecular informatics

DOI: 10.1002/minf.201900107

iQSPR in XenonPy: A Bayesian Molecular Design Algorithm

Stephen Wu[+],*[a, b] Guillaume Lambard[+],[c] Chang Liu[+],[a] Hironao Yamada,[a, d] and Ryo Yoshida*[a, b, c]

Abstract: iQSPR is an inverse molecular design algorithm based on Bayesian inference that was developed in our previous study. Here, the algorithm is integrated in Python as a new module called iQSPR-X in the all-in-one materials informatics platform XenonPy. Our new software provides a flexible, easy-to-use, and extensible platform for users to build customized molecular design algorithms using pre-set modules and a pre-trained model library in XenonPy. In this paper, we describe key features of iQSPR-X and provide guidance on its use, illustrated by an application to a polymer design that targets a specific range of bandgap and dielectric constant.

Keywords: molecular design · machine learning · Bayesian inference · open source · polymer

1 Introduction

Inverse molecular design is the process of computationally creating new chemical structures that exhibit desired properties, and this approach has been one of the most important research subjects in materials science. For decades, scientists have searched for efficient methods of discovering novel materials for a wide variety of industrial and engineering applications. Conventional approaches have often relied on expert knowledge to investigate new structures by trial and error, starting from known materials and considering a relatively small sub-region in the whole search space. Although the chemical space of small organic molecules consists of approximately 10^{60} candidates,[1] the total number of currently known compounds is at most on the order of 10^{8}.[2] Hence, most of the chemical space remains unexplored, and the concept of computer-aided molecular design (CAMD) has emerged to accelerate this extremely slow discovery process.[3]

An early attempt by Joback and Stephanopoulos framed CAMD as an optimization problem with rule-based molecule enumeration.[4] In many subsequent works, materials properties were optimized within a search space that was pre-constrained to a small subspace built from expert-selected molecular fragments or chemical rules, using heuristic optimization algorithms such as genetic algorithms[5,6] and Monte Carlo based stochastic optimization.[7] For example, Miyao et al.[8,9] used a set of chemically favorable fragments and designed templates of specific molecular graphs that were combined with some mixture models for property predictions to generate a desired class of candidate molecules. Although these methods were a major step forward in the history of CAMD, they still suffered from a lack of capability to handle the large and highly diverse discrete spaces of candidate molecules.

In recent years, a new family of CAMD algorithms has emerged, inspired by the great success of modern machine learning (ML) methods. In particular, to broaden the search space, ML methods that use probabilistic language models based on deep neural networks (DNNs) have proliferated intensively since 2017.[10] In these methods, a language model is trained on a given set of existing molecules, the chemical structures of which are translated into a set of strings according to the simplified molecular-input line-entry system (SMILES) chemical language.[11] Models trained to recognize chemically realistic structures are then used to refine chemical strings in the molecular design calculation. Promising examples have included various types of varia-

[a] *S. Wu,[+] C. Liu,[+] H. Yamada, R. Yoshida*
The Institute of Statistical Mathematics, Research Organization of Information and Systems
10-3 Midori-cho, Tachikawa, Tokyo 190-8562, Japan
phone/fax +81 (0) 50–5533-8534
E-mail: stewu@ism.ac.jp
yoshidar@ism.ac.jp
[b] *S. Wu,[+] R. Yoshida*
The Graduate University for Advanced Studies, SOKENDAI,
10-3 Midori-cho, Tachikawa, Tokyo 190-8562, Japan
[c] *G. Lambard,[+] R. Yoshida*
Center for Materials Research by Information Integration (CMI^2), Research and Services Division of Materials Data and Integrated System (MaDIS), National Institute for Materials Science (NIMS)
1-2-1 Sengen, Tsukuba, Ibaraki 305-0047, Japan
[d] *H. Yamada*
School of Pharmacy, Tokyo University of Pharmacy and Life Sciences
1432-1 Horinouchi, Hachioji, Tokyo 192-0392, Japan
[+] *Equally contributed to this work*

Supporting information for this article is available on the WWW under https://doi.org/10.1002/minf.201900107

tional autoencoders,[12–15] generative adversarial networks,[16] recurrent neural networks,[17,18] and so on. These methods have been able to produce diverse chemical structures; however, they often require large training datasets to obtain a DNN-based generator that can produce chemically realistic molecules with grammatically valid SMILES. Datasets this large are unavailable in many applications. Furthermore, many of these methods generate chemically or grammatically invalid representations of molecules at relatively high rates, unless their hyperparameters are carefully tuned.[19–21]

Some previous works considered simpler generative models to avoid the need to train the model with large dataset. Yoshikawa et al.[22] exploited a grammatical evolution method with parallel computation to generate a diverse set of candidate molecules conditional on arbitrarily given design targets. Ikebata et al.[23] combined a simple probabilistic language model based on an *n*-gram representation of SMILES sequences with a Bayesian inference framework to sequentially modify a population of molecules into promising candidate molecules that would exhibit desired properties. For a more complete review of the above methods, Schwalbe-Koda and Gómez-Bombarelli[24] have provided a detailed overview of recent developments in inverse molecular design.

In this paper, we introduce iQSPR-X, a flexible software constructed to implement the Bayesian molecular design algorithm iQSPR, which was developed in our previous work.[23] The algorithm was implemented in XenonPy, a Python package with an integrated platform of materials informatics.[25] In contrast to the original iQSPR algorithm developed in R, the new version allows users to exploit various features of XenonPy as described below. The basic computational workflow consists of a two-step iteration: (1) current chemical structures are modified to new ones using a generator and (2) candidate molecules that show promise for desired properties are selected using an evaluator, which is a set of ML models for predicting material properties. The generator and the evaluator can be pre-trained separately with given training instances. Users can either train new models from scratch or reuse relevant pre-trained models from a model library in XenonPy, which covers a broad array of material properties for small molecules and polymers. In addition, when the available data on the structure-property relationship for a target task are limited, directly obtaining a reliable prediction model is difficult. However, an ML technique called transfer learning can be used to extract knowledge relevant to the target task from a large set of pre-trained models to help training new models more efficiently.[26] Successful application of the iQSPR method, in conjunction with transfer learning to overcome limited polymeric properties data, was demonstrated in our previous study, which achieved the discovery of new polymers with high thermal conductivity.[27] A set of tutorials distributed as Jupyter notebooks are available at the website of XenonPy,[25] and these include detailed explanations and sample codes for building customized generators and evaluators, performing the inverse design calculations, and using some of the convenient modules in XenonPy. In this paper, we highlight some key features of iQSPR-X and describe its application to the task of designing polymers using data from Polymer Genome (PG).[28,29]

2 Computational Methods

2.1 Bayesian Molecular Design

The primary task of the Bayesian molecular design is to draw a set of samples from the posterior distribution $P(S \mid \boldsymbol{Y} \in U)$, which represents the conditional probability of observing a chemical structure S, given material properties $\boldsymbol{Y} = \{Y_i \mid i = 1,\ldots,m\}$, that lies in a target region U. In the iQSPR-X implementation, S is encoded as a SMILES string; i.e., $S = s_1 s_2 \ldots s_n$, where s_i is any valid character in SMILES. For example, phenol (C_6H_6O) can be represented by the SMILES string "C1=CC=C(C=C1)O", where C and O denote the carbon and oxygen atoms, respectively; "=" denotes a double bond; the two "1" digits denote the opening and closing of the ring structure; and the parentheses denote the beginning and ending of the branching component.

According to Bayes' theorem, a posterior distribution is proportional to the product of a likelihood function and a prior distribution:

$$P(S|\boldsymbol{Y} \in U) \;\propto\; P(\boldsymbol{Y} \in U|S)P(S)$$

where $P(\boldsymbol{Y} \in U \mid S)$ represents the likelihood function that evaluates the goodness-of-fit of S with respect to the given property requirement $\boldsymbol{Y} \in U$, and $P(S)$ represents the prior probability that S belongs to a predefined search space of SMILES strings. Thus, $P(S)$ will deliver a small or even zero probability when presented with an unfavorable or chemically unrealistic structure, thereby acting as a filter for such out-of-scope or invalid structures. In iQSPR-X, a sequential Monte Carlo algorithm proposed by Ikebata et al.[23] is implemented. This algorithm is somewhat similar to a genetic algorithm. With a given set of initial samples $\boldsymbol{S}_0 = \{S_i^0 \mid i = 1,\ldots,N\}$ of size N, the pre-trained prior is used as a generator to propose a new set of samples $\boldsymbol{S}_0'$. A fitness score is then assigned to each sample in $\boldsymbol{S}_0'$ using the likelihood, which is the evaluator in iQSPR-X. By resampling N samples from $\boldsymbol{S}_0'$ in proportion to the fitness scores, a refined set $\boldsymbol{S}_1$ is obtained and once again modified by the generator. This cycle is repeated T times to obtain a final sample set $\boldsymbol{S}_T$.

There are three important building blocks in this algorithm: the generator (prior), the evaluator (likelihood), and the descriptor $\varphi(S)$. When building models for the evaluator, we encode a chemical structure into a descriptor vector $\varphi(S)$ using, for example, a molecular fingerprinting algorithm. Using training instances $\{(\boldsymbol{Y}_k, S_k) \mid k = 1,\ldots,N_{data}\}$ on

the structure-property relationships, we then derive a model that describes the materials properties $\boldsymbol{Y}$ as a function of the descriptor $\varphi(S)$, defining $\hat{\boldsymbol{Y}}=\boldsymbol{\mu}(\varphi(S))$ with the trained model $\boldsymbol{\mu}$. Although iQSPR-X allows users to plug in customized functions for each building block, we also provide some commonly used functions internally, and these can be directly called from the package. For the descriptor, all available fingerprint types in RDKit[30] and the Python descriptor package Mordred[31] are available by default. Users can alternatively use a set of features extracted from pre-trained neural networks in the XenonPy model library, as described in the next section. For the evaluator, a Gaussian likelihood is given as a choice with any user-defined model $\mu_i(\varphi(S))$ and the standard deviation $\sigma_i(S)$, which represents the uncertainty of predicted properties:

$$P(\boldsymbol{Y}\in U|S)=\int_U\prod_{i=1}^{m}\frac{1}{\sigma_i(\phi(S))\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{Y_i-\mu_i(\phi(S))}{\sigma_i(\phi(S))}\right)^2\right)\mathrm{d}Y_1\ldots\mathrm{d}Y_m,$$

where U is the target region in the m dimensional space, and $\mu_i(\varphi(S))$ and $\sigma_i(\varphi(S))$ are the mean and standard deviation for the ith property, respectively, obtained from ML models with input $\varphi(S)$. For the generator, the extended n-gram model developed by Ikebata et al.[23] can be used by training it with any chemical structures given in SMILES. The model takes the form $P(S)=P(s_1)\prod^{n}{}_{i=2}\ P(s_i|s_{i-1},\ldots,s_1)$. Figure 1 summarizes the computational workflow of iQSPR-X.

2.2 Generator: Extended *n*-gram Model

The role of the generator is to propose new candidate molecules modified from a set of initial molecules. We implemented the extended n-gram model as an internally available function in iQSPR-X. This model consists of two components: (1) a table that records the probability of observing a subsequent character given a substring and (2) a function that modifies a given SMILES string based on the stored n-gram probability table. The table can be trained by supplying a set of SMILES strings sampled from the desired search space. The maximum length of a substring to be considered and stored in the table is controlled by the "order" parameter. In the extended n-gram model, SMILES strings are internally tokenized into a list of characters. For example, "=O" and "%10" are considered as one character, and a terminal character is automatically added at the end of each string. When proposing a new candidate molecule, the modifier function deletes a random number of characters from the end of the SMILES string, and then elongates the shortened string based on the n-gram table. Because the representation of a molecule in SMILES is not unique, a reordering of the SMILES string is probabilistically performed to avoid constantly modifying the same part of the chemical structure.

In short, the most important parameters in modelling the generator include the probability required to trigger reordering, the range of the number of letters to be deleted, and the order parameter controlling the maximum length of a substring in training and sampling the n-gram model. Users can adjust these parameters based on the expected molecule size in the targeted search space. Although SMILES is a powerful representation of chemical structures, as exemplified by its ability to handle chirality using the "@" symbol, the non-uniqueness of SMILES representations may lead to subtle effects in certain usages. For example, the aromatic ring in phenol can be represented as "C1=CC=CC=C1" or "c1ccccc1." We recommend that users not mix different representations of the same molecular structure when training the extended n-gram model.

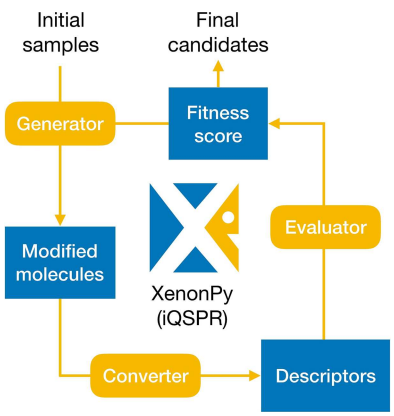

Figure 1. Computational workflow in iQSPR-X with three main building blocks that users can flexibly construct: the generator, the evaluator, and the converter that translates an input chemical structure into a descriptor vector.

2.3 Evaluator: Likelihood Function

The role of the evaluator is to provide a fitness score for a candidate molecule to estimate how likely the candidate possesses the desired properties. iQSPR-X allows users to write their own evaluator, which receives a list of molecules, converts them to a set of descriptors using a pre-set descriptor conversion function, and returns a list of corresponding log-likelihood values. A Gaussian likelihood function can also be used if users select a desired descriptor and provide an ML model that returns the mean and standard deviation for a given set of descriptors as input.

2.4 Pre-trained Neural Descriptors in XenonPy

One of the most distinctive features of our software is the availability in XenonPy of a comprehensive set of pre-

trained neural features for use as the descriptor $\varphi(S)$. The sampling efficiency of iQSPR-X is highly influenced by the reliability of the evaluator that predicts the material properties for any given chemical structure. Building such models from scratch is often time-consuming and requires a large set of training data, which is not available in many applications. XenonPy currently provides 140,000 pre-trained neural networks for the prediction of physical, chemical, electronic, thermodynamic, and mechanical properties of small organic molecules, polymers, and inorganic crystalline materials, with models for 15, 18, and 12 properties of these material types, respectively. The models are distributed as MXNet[32] (R) and/or PyTorch[33] (Python) model objects. The distributed API (application programming interface) allows users to query the XenonPy.MDL database. Users can directly use a retrieved model relevant to the target task, if available, or can re-train a pre-trained model on the target task using a transfer learning technique as described below. Transfer learning has significant potential to overcome the problem of limited materials property data, as demonstrated in our previous study,[26] for various materials science tasks. Other studies have also shown promising applications of transfer learning in materials informatics.[18,34–41]

In this study, we applied a specific type of transfer learning using pre-trained neural networks. For a target property, a neural network pre-trained on proxy properties is available in the library, where the source datasets are sufficiently large. If the two properties are physically or chemically interrelated, the pre-trained models can be expected to autonomously acquire common features relevant to the proxy properties. The features learned by solving the related tasks are partially transferable to the descriptor $\varphi(S)$ in a model constructed for the target task.

In general, earlier or shallower layers in a neural network tend to acquire general features to form the basis of the material descriptions, and only the last one or two layers identify specific features for the prediction of a source property. In iQSPR-X, we freeze the shallower layers for use as a feature extractor. A subnetwork $\varphi(S)$ of such a pre-trained model can be reused in the supervised learning of the target property. To simplify the implementation of the repetitious tasks of neural descriptor extraction, XenonPy provides users with an internal function to extract values from any hidden layer in a pre-trained neural network. With its large library of pre-trained models and wide range of built-in descriptors, XenonPy provides a strong foundation for flexibly arranging the necessary building blocks of the iQSPR algorithm.

3 Results and Discussion

3.1 Data

We used data from PG to illustrate the use of iQSPR-X based on an example motivated from a previous study on polymer design.[28] PG is an open database for polymeric properties that currently contains 854 polymers composed of nine types of atoms (H, C, O, N, S, F, Cl, Br, and I) with experimental data for three material properties (glass transition temperature, density, and solubility parameter) and computational data from density functional theory (DFT) for four material properties (bandgap (E_{gap}), refractive index, dielectric constant (ε_{tot}), and atomization energy). Using a subset of the data (4-block polymers composed of CH_2, NH, CO, C_6H_4, C_4H_2S, CS, and O), Mannodi-Kanakkithodi et al.[28] designed 6- to 12-block polymers with high ε_{tot} for insulator applications using ML models and a genetic algorithm. They were specifically interested in polymers with higher ε_{tot} and E_{gap}, and this goal was adopted in our example. The given data of the chemical structures S and their materials properties were used to train the generator and the evaluator. Here, we considered S to be the SMILES strings of the repeating polymer units. The connection points, i.e., the head and tail of a monomer, were denoted as "*".

In PG, the lowest-energy crystal structures of the polymers were used for the DFT calculation. For each polymer, E_{gap} was computed using a hybrid Heyd-Scuseria-Ernzerhof (HSE06) electronic exchange-correlation functional, and ε_{tot}, which is the sum of the electronic and ionic dielectric constants, was computed using density functional perturbation theory (DFPT). Mannodi-Kanakkithodi et al.[28] have detailed this computational procedure. As shown in Figure 2a, we observed an inverse relation between ε_{tot} and E_{gap}. Polymers containing thiophene (C_4H_2S) tended to reach high ε_{tot}, but generally had low E_{gap}. In contrast, polymers containing fluorine (F) atoms tended to reach high E_{gap}, but generally had low ε_{tot}. However, in contrast to the enrichment offered by either C_4H_2S or F atoms, polymers exhibiting high ε_{tot} and high E_{gap} tended to be composed of CH_2, NH, CO, C_6H_4 and O.[28] The design objective was to solve this nontrivial trade-off problem.

3.2 Training the Generator

In this study, we considered two ways to train the extended n-gram model. First, we used all 854 polymers in PG as a training set, which covered a wide variety of polymers. Second, we focused only on specific types of chemical structures that shared some common features, taken from other data sources. In practice, users may often be interested in designing a specific class of molecules. Here, we explored F-containing polymers with high ε_{tot} and E_{gap}. In particular, we focused on a training set containing the

fragment "C(F)(F)N," which was taken from PubChem.[2,42] Because of the extremely high diversity of chemical structures in PubChem, we used multiple steps to extract training molecules: (1) we screened molecules in PubChem continuously until 5,000 molecules having the desired fragment were found (screening a total of over 36,940,000 molecules); (2) we reduced the number of molecules to 3,860 that consisted only of C, O, N, F, and/or S atoms; (3) we finally extracted 2,485 molecules by filtering out those that had more than six F atoms or included more than one molecule in a single SMILES string (SMILES string with "."). The final training set was formed by the union of these selected PubChem molecules and the set of PG polymers. The order parameter controlling the length of a substring for training and sampling the *n*-gram tables was set to be 20 for both cases, after examining the distribution of the SMILES lengths of the molecules in PG (see Figure 2b). In the construction of a training set, duplicates of each SMILES string were generated by performing random reordering of the string for at most 15 times. This step is important to avoid the occurrence of unseen substring patterns during the generation of new molecules, considering that we set the reorder probability to be 0.5 during the molecular generation. Figure 3 illustrates how the two different generators modified molecules step-by-step starting from the same initial chemical structures. The generator trained on the PubChem molecules showed a stronger tendency to include F-containing fragments during the modification process.

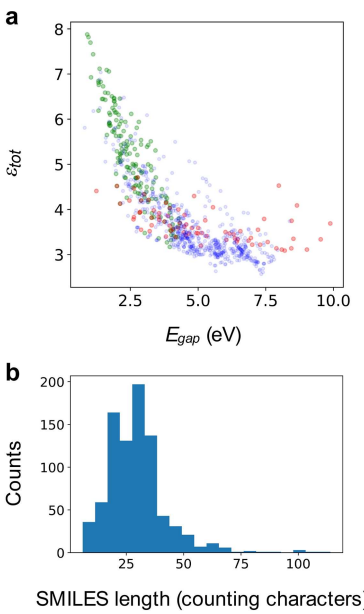

Figure 2. Summary of observed data in PG. (a) Joint distribution of ε_{tot} and E_{gap}. Red dots denote all polymers containing F atoms, green dots denote those having C_4H_2S as fragments, and blue dots denote all other polymers. (b) Histogram of the lengths of SMILES strings in PG.

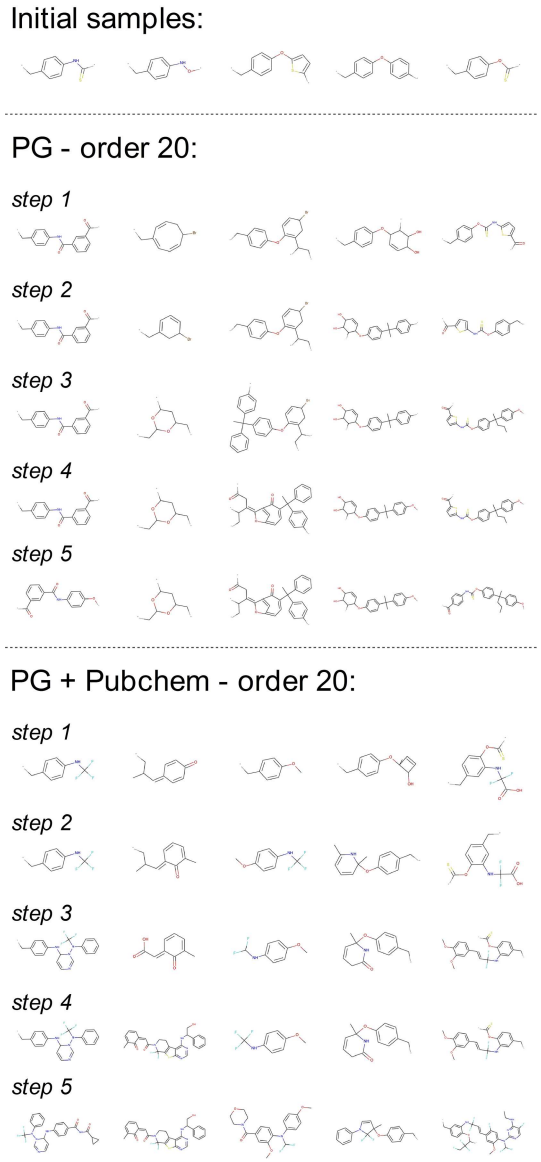

Figure 3. Modification of molecules using extended *n*-gram models trained with different datasets. The same five chemical structures in PG were successively modified five times according to generators that were trained on 854 polymers from PG (top) and with the 854 polymers from PG and 2,485 F-containing molecules from PubChem (bottom).

3.3 Training the Evaluator

We conducted a series of experiments to obtain a model for the Gaussian likelihood function. As a descriptor, we used pre-trained neural network models in XenonPy that were trained with 10 different types of fingerprints available in RDKit: atom pair and topological torsion fingerprints, Morgan fingerprints (both feature-based and not feature-based), basic fingerprints in RDKit, and five more that were obtained by adding the MACCS keys to the five listed fingerprints. With each of the 10 fingerprints, 100 randomly

constructed neural networks, each having six fully connected layers, were trained with either the ε_{tot} or E_{gap} datasets; the number of epochs was 2,000 and the dropout rate was 0.1. The dataset was randomly separated into training and validation sets at a ratio of 8:2. Figure 4 shows

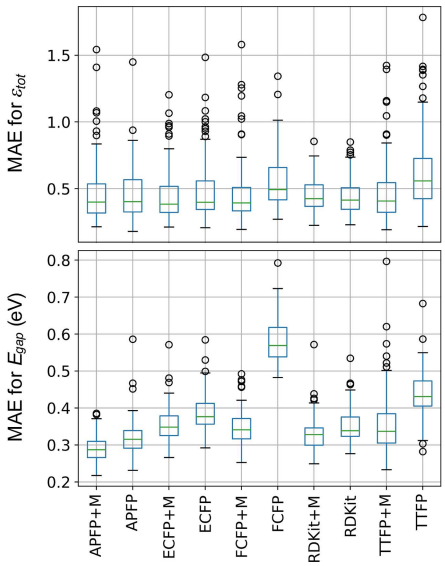

Figure 4. Box-plots of the MAEs across different fingerprint descriptors evaluated on the validation datasets of either ε_{tot} or E_{gap}. APFP denotes the atom pair fingerprints, ECFP denotes the non-feature-based radius-3 Morgan fingerprints, FCFP denotes the feature-based radius-3 Morgan fingerprints, TTFP denotes the topological torsion fingerprints, RDKit denotes the basic fingerprints in RDKit, and +M denotes the addition of the MACCS keys.

a comparison of the validated mean absolute errors (MAEs) across the different fingerprint descriptors. The atom pair fingerprints with the MACCS keys showed consistently high performance on both ε_{tot} and E_{gap}, and was therefore selected for use in the Bayesian molecular design.

The default model in the Gaussian likelihood function is set to be a Bayesian linear regression model. Users can directly train the model with their input data using a one-line Python script. In this paper, we considered three more approaches to constructing models that return μ and σ: (1) bagging to calculate a bootstrap variance σ for any deterministic model, (2) random forests combined with a jackknife method,[43] and (3) Bayesian linear models with neural descriptors extracted from pre-trained models.[26] In our example, we tested these different methods to select the best prediction model. Five-fold cross validation (CV) was performed on both ε_{tot} and E_{gap} and the model with the best prediction performance was selected.

For the bagging approach, the gradient boosting method in scikit-learn[44] was used and the training data in each fold of the 5-fold CV were further divided into 10 non-overlapping bags. Each bag produced a gradient boosting regressor under the default setting in scikit-learn. The mean and standard deviation of the predicted values from the 10 trained models were taken as μ and σ, respectively.

For the random forest approach, the *forestci* package was used along with the random forest method in scikit-learn to calculate μ and σ. The number of trees was set to be 500 and the "max_feature" option was selected to be "sqrt."

For the Bayesian linear regression with neural descriptors, we began by selecting pre-trained model from the model library in XenonPy for each of the two target properties. The 100 pre-trained neural networks of ε_{tot} and E_{gap} were modified such that the last hidden layers were connected to Bayesian linear regressors, and the prediction performances of the models were then evaluated by the 10-fold CV applied to the training data within each fold of the 5-fold CV. Each of the models of ε_{tot} and E_{gap} that achieved the overall lowest MAE was selected, and their last hidden layers were concatenated to form a new neural descriptor. This descriptor was used to replace the originally selected descriptor in the default Gaussian likelihood function. Finally, this evaluator was trained with the full training data within each fold of the five-fold CV.

Figure 5 shows the performance of each model on the five-fold CV for the ε_{tot} and E_{gap} datasets. The bagging approach with the gradient boosting model achieved the best overall performance and was therefore selected for the inverse design calculation.

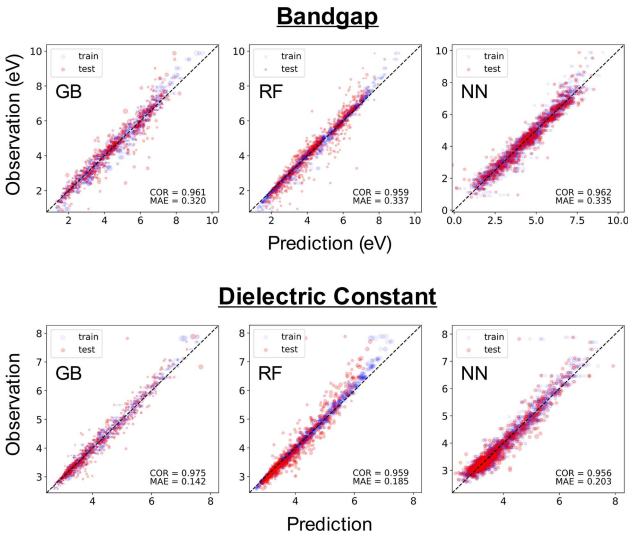

Figure 5. Prediction performance of different models on the five-fold CV for the ε_{tot} and E_{gap} datasets. GB denotes bagging with gradient boosting, RF denotes random forests with jackknife-based uncertainty quantification, and NN denotes pre-trained neural networks with their last hidden layers connected to Bayesian linear regressors.

3.4 Design Results

In this example, we set the target property region to be $\varepsilon_{tot} > 4.5$ and $E_{gap} > 5$ eV. Three rounds of iQSPR-X were executed using different setups to compare the effect of various components of the algorithm. The first run used the generator trained with molecules in PG, and in the inverse design calculation, 100 initial samples were randomly selected from the 854 molecules in PG. The second run used the same generator, but the 100 initial samples were randomly selected from a subset of the molecules in PG that had a relatively low ε_{tot} or E_{gap} ($\varepsilon_{tot} < 4$ or $E_{gap} < 4.5$ eV). The third run used the same initial samples as in the first run, but the generator was trained with the PG and PubChem molecules, as detailed in the previous section.

Other components of iQSPR-X were set to be the same for all three runs. The *n*-gram order parameter was set to be 20, with the range of the number of letters to be deleted set to be 1–10. The reordering probability was set to be 0.5. The descriptor was selected to be a concatenation of the atom pair fingerprints and the MACCS keys in RDKit. The evaluator was selected to be the Gaussian likelihood with 10 "bags" of gradient boosting models trained on 10-fold CV of the full PG datasets for ε_{tot} and E_{gap}. The mean function μ was given by the mean of the predictions from the 10 models. For practical purposes, the variance function σ^2 was composed of the bootstrap variance plus a tiny pre-set constant (0.04 for ε_{tot} and 0.09 for E_{gap}). To avoid trapping at a local region of the entire search space, an annealing schedule was applied: the likelihood scores were powered by a sequence of factors from 0 to 1, which corresponds to a sequential transformation of a distribution from a uniform distribution to the actual posterior distribution. From empirical evidence, a slow cooling schedule is recommended. We started with 20 steps of powers linearly increasing from 0 to 0.2, 10 additional steps linearly increasing from 0.2 to 0.4, and another 10 steps linearly increasing from 0.4 to 1. Finally, we performed another 60 steps with the power fixed at 1, and these samples were recorded as candidate molecules.

Movies S1, S2, and S3 in the supplementary materials demonstrate how the candidate molecules proposed in each step of the sequential Monte Carlo approach the target region. In the first run, one of the initial samples was observed to reach the target region, and a number of the samples continued to explore structures similar to that molecule, whereas other samples pursued alternative possibilities. In the end, the best proposed candidate molecules converged to molecules similar to those found in the previous study[28] (see Figure 6).

In contrast, with a significantly different set of initial samples, the second run struggled to converge to candidate molecules similar to the first run. Instead, it became trapped at molecules with complex ring structures (see Figure 7). For an intractable trade-off problem such as this example, a small finite set of samples cannot support full

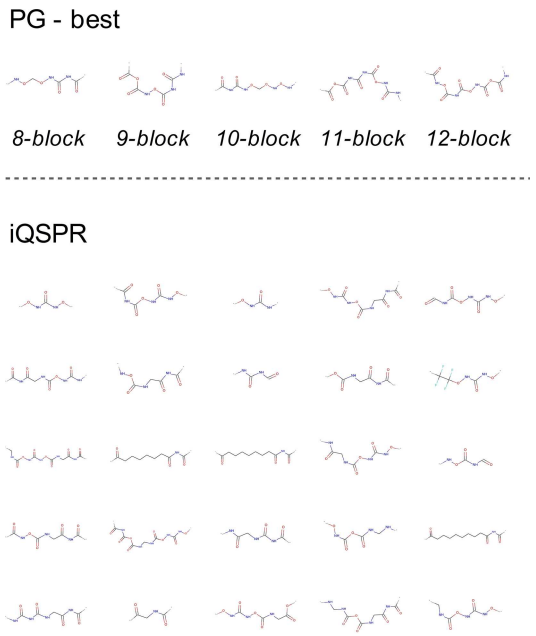

Figure 6. Comparison of the best candidate molecules from a previous study[28] and the top 25 candidate molecules generated from iQSPR-X. The optimal combinations of 8 to 12 building blocks that were proposed in the previous study are shown as a comparison.

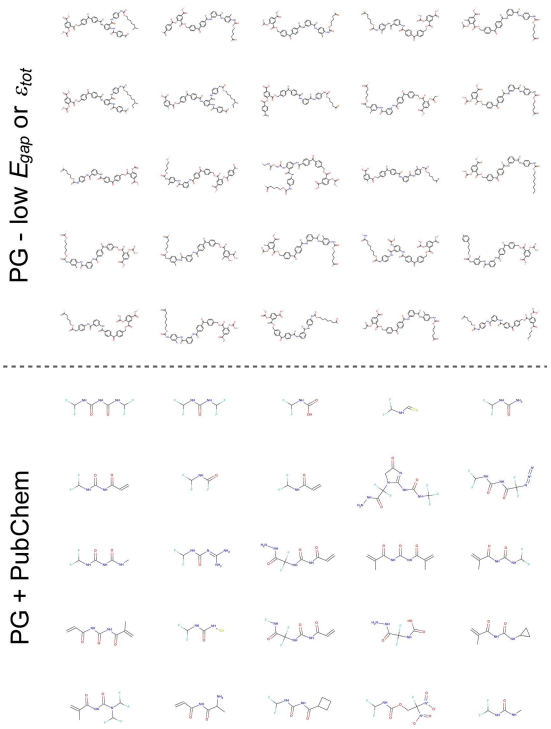

Figure 7. Comparison of the top 25 candidate molecules generated from iQSPR-X with initial samples randomly drawn from polymers in PG with $\varepsilon_{tot} < 4$ or $E_{gap} < 4.5$ eV and the top 25 candidate molecules generated using an extended *n*-gram model trained with samples from both PG and PubChem.

exploration of the search space. Increasing the number of samples is an intuitive yet computationally intensive solution. An alternative is to adjust parameters in the iQSPR-X algorithm, such as the initial samples, the *n*-gram order, the number of letters to be deleted, and so on.

iQSPR-X can also be used to search intensively for a specific molecular subspace. In the third run, the generator showed a clear tendency to attach F-containing fragments to chemical structures after being trained with thousands more samples from PubChem. As a result, we observed the frequent appearance of molecules with relatively higher E_{gap} during the sequential Monte Carlo iterations. The best candidate molecules were composed of the F-containing fragments with different combinations of the CH_2, NH, and CO blocks (see Figure 7).

4 Conclusions

iQSPR-X is an ML engine for generating a target-specific molecular library. XenonPy provides an all-in-one ML-based materials design platform, in which descriptor calculations, property prediction models for high throughput screening, molecular library generators, and inverse design algorithms are all present as independent modules that users can either take as pre-existing functions in XenonPy or build flexibly to accommodate their own needs in conjunction with other major ML and materials informatics Python packages. Moreover, transfer learning offers further capability and convenience to the ML technique. By implementing the iQSPR algorithm with the XenonPy platform, users can fully enjoy the benefits of a wide range of pre-existing functions and models that will greatly simplify the process of establishing the Bayesian inverse design algorithm. Detailed tutorials for each component are available at the website of XenonPy.[25]

In this paper, we demonstrated some basic functionalities of iQSPR-X by applying it to the task of designing polymers exhibiting high ε_{tot} and E_{gap}. We showed how changes to the setup of iQSPR-X, such as the initial sample sets and the generator, might influence the outcome of the computational workflow. One of our runs identified chemical structures that were similar to the best candidate molecules proposed in the original study. Furthermore, we demonstrated that by including a focused set of molecules in the training process of the generator, we were able to guide the algorithm to search a particular subspace in the large molecule space. Moreover, although users can quickly start the inverse design process using the default functions and setups, the true potential of the algorithm can be realized by building customized modules for a variety of tasks in materials science. The XenonPy project aims to gather contributions from various users in diverse fields of materials and data science. Contributors are highly welcome to share and implement their own codes in XenonPy as off-the-shelf modules.

5 Supplementary Materials

1) Movie S1-PG_basic.avi: This video shows the evolution of the material property values of the proposed candidate molecules in each step of the XenonPy-iQSPR iterations for the first run of our example. Blue dots denote the original data from PG, and red dots denote the proposed candidate molecules, with the radius of the dots proportional to the sum of the predicted variances of ε_{tot} and E_{gap}. Beta refers to the power value used in the annealing schedule.
2) Movie S2-PG_lowVal.avi: This video is the same type as Movie S1, for the second run of our example.
3) Movie S3-PG_Pubchem.avi: This video is the same type as Movie S1, for the third run of our example.

Conflict of Interest

None declared.

Acknowledgements

This work was supported in part by the Materials Research by Information Integration Initiative (MI^2I) of the Support Program for Starting Up Innovation Hub from the Japan Science and Technology Agency (JST). R.Y. acknowledges financial support from a Grant-in-Aid for Scientific Research (B) 15H02672, a Grant-in-Aid for Scientific Research (A) 19H01132 from the Japan Society for the Promotion of Science (JSPS), JST CREST Grant Number JPMJCR19I1, Japan, and JSPS KAKENHI Grant Number JP19H05820. S.W. gratefully acknowledges financial support from JSPS KAKENHI Grant Number JP18 K18017.

References

[1] R. S. Bohacek, C. McMartin, W. C. Guida, *Med. Res. Rev.* **1996**, *16*, 3–50.
[2] S. Kim, P. A. Thiessen, E. E. Bolton, J. Chen, G. Fu, A. Gindulyte, L. Han, J. He, S. He, B. A. Shoemaker, J. Wang, B. Yu, J. Zhang, S. H. Bryant, *Nucleic Acids Res.* **2016**, *44*(D1), D1202–1213.
[3] N. D. Austin, N. V. Sahinidis, D. W. Trahan, *Chem. Eng. Res. Des.* **2016**, *116*, 2–26.
[4] K. G. Joback, G. Stephanopoulos, *Proc. FOCAPD, Snowmass, CO*, **1989**, 363–387.
[5] D. J. Wales, H. A. Scheraga, *Science* **1999**, *285*, 1368–1372.
[6] D. Douguet, E. Thoreau, G. Grassy, *J. Comput.-Aided Mol. Des.* **2000**, *14*, 449–466.
[7] X. Hu, D. N. Beratan, W. Yang, *J. Chem. Phys.* **2008**, *129*, 064102.
[8] T. Miyao, H. Kaneko, K. Funatsu, *J. Comput.-Aided Mol. Des.* **2016**, *30*, 425–446.
[9] T. Miyao, H. Kaneko, K. Funatsu, *J. Chem. Inf. Model.* **2016**, *56*, 286–299.

[10] B. Sanchez-Lengeling, A. Aspuru-Guzik, *Science* **2018**, *361*, 360–365.
[11] D. Weininger, *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 31–36.
[12] M. J. Kusner, B. Paige, J. M. Hernández-Lobato, *PMLR* **2017**, *70*, 1945–1954.
[13] A. Kadurin, S. Nikolenko, K. Khrabrov, A. Aliper, A. Zhavoronkov, *Mol. Pharmaceutics* **2017**, *14*, 3098–3104.
[14] J. Lim, S. Ryu, J. W. Kim, W. Y. Kim, *J. Cheminf.* **2018**, *10*, 31.
[15] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, A. Aspuru-Guzik, *ACS Cent. Sci.* **2018**, *4*, 268–276.
[16] B. Sanchez-Lengeling, C. Outeiral, G. L. Guimaraes, A. Aspuru-Guzik, *ChemRxiv* **2017**, 10.26434/chemrxiv. 5309668.v3.
[17] X. Yang, J. Zhang, K. Yoshizoe, K. Terayama, K. Tsuda, *Sci. Technol. Adv. Mater.* **2017**, *18*, 972–976.
[18] M. H. S. Segler, T. Kogej, C. Tyrchan, M. P. Waller, *ACS Cent. Sci.* **2018**, *4*, 120–131.
[19] W. Jin, R. Barzilay, T. Jaakkola, *PMLR* **2018**, *80*, 2323–2332.
[20] H. Kajino, *arXiv* **2018**, 1809.02745.
[21] N. Brown, M. Fiscato, M. H. S. Segler, A. C. Vaucher, *J. Chem. Inf. Model.* **2019**, *59*, 1096–1108.
[22] N. Yoshikawa, K. Terayama, M. Sumita, T. Homma, K. Oono, K. Tsuda, *Chem. Lett.* **2018**, *47*, 1431–1434.
[23] H. Ikebata, K. Hongo, T. Isomura, R. Maezono, R. Yoshida, *J. Comput.-Aided Mol. Des.* **2017**, *31*, 379–391.
[24] D. Schwalbe-Koda, R. Gómez-Bombarelli, *arXiv* **2019**, 1907.01632.
[25] XenonPy, *https://xenonpy.readthedocs.io/en/latest/, last accessed on July 20, 2019.*
[26] H. Yamada, C. Liu, S. Wu, Y. Koyama, S. Ju, J. Shiomi, J. Morikawa, R. Yoshida, *ACS Cent. Sci.* **2019**, *online pre-release*.
[27] S. Wu, Y. Kondo, M.-A. Kakimoto, B. Yan, H. Yamada, I. Kuwajima, G. Lambard, K. Hongo, Y. Xu, J. Shiomi, C. Schick, J. Morikawa, R. Yoshida, *npj Comput. Mater.* **2019**, *5*, 66.
[28] A. Mannodi-Kanakkithodi, G. Pilania, T. D. Huan, T. Lookman, R. Ramprasad, *Sci. Rep.* **2016**, *6*, 20952.
[29] C. Kim, A. Chandrasekaran, T. D. Huan, D. Das, R. Ramprasad, *J. Phys. Chem. C* **2018**, *122*, 17575–17585.
[30] G. Landrum, *http://www.rdkit.org, last accessed on July 20,* **2019**.
[31] H. Moriwaki, Y.-S. Tian, N. Kawashita, T. Takagi, *J. Cheminf.* **2018**, *10*, 4.
[32] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, Z. Zhang, *arXiv* **2015**, 1512.01274.
[33] PyTorch, *https://pytorch.org/, last accessed on July 20,* **2019**.
[34] M. L. Hutchinson, E. Antono, B. M. Gibbons, S. Paradiso, J. Ling, B. Meredig, *arXiv* **2017**, 1711.05099.
[35] H. Oda, S. Kiyohara, K. Tsuda, T. Mizoguchi, *J. Phys. Soc. Jpn.* **2017**, *86*, 123601.
[36] R. Jalem, K. Kanamori, I. Takeuchi, M. Nakayama, H. Yamasaki, T. Saito, *Sci. Rep.* **2018**, *8*, 5845.
[37] T. Yonezu, T. Tamura, I. Takeuchi, M. Karasuyama, *Phys. Rev. Mater.* **2018**, *2*, 113802.
[38] B. Kailkhura, B. Gallagher, S. Kim, A. Hiszpanski, T. Y.-J. Han, *arXiv* **2019**, 1901.02717.
[39] E. D. Cubuk, A. D. Sendek, E. J. Reed, *J. Chem. Phys.* **2019**, *150*, 214701.
[40] X. Li, Y. Zhang, H. Zhao, C. Burkhart, L. C. Brinson, W. Chen, *Sci. Rep.* **2018**, *8*, 13461.
[41] M. Kaya, S. Hajimirza, *Sci. Rep.* **2019**, *9*, 5034.
[42] S. Kim, J. Chen, T. Cheng, A. Gindulyte, J. He, S. He, Q. Li, B. A. Shoemaker, P. A. Thiessen, B. Yu, L. Zaslavsky, J. Zhang, E. E. Bolton, *Nucleic Acids Res.* **2019**, *47*, D1102–1109.
[43] S. Wager, T. Hastie, B. Efron, *J. Mach. Learn. Res.* **2014**, *15*, 1625–1651.
[44] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.

Received: August 16, 2019
Accepted: October 14, 2019
Published online on November 5, 2019

