Adjustive Linear Regression and Its Application to the Inverse QSAR

Jianshen Zhu[1], Kazuya Haraguchi[1][a], Hiroshi Nagamochi[1][b] and Tatsuya Akutsu[2][c]
[1]*Department of Applied Mathematics and Physics, Kyoto University, Kyoto 606-8501, Japan*
[2]*Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji 611-0011, Japan*

Keywords: Machine Learning, Linear Regression, Integer Programming, Linear Program, Cheminformatics, Materials Informatics, QSAR/QSPR, Molecular Design.

Abstract: In this paper, we propose a new machine learning method, called adjustive linear regression, which can be regarded as an ANN on an architecture with an input layer and an output layer of a single node, wherein an error function is minimized by choosing not only weights of the arcs but also an activation function at each node in the two layers simultaneously. Under some conditions, such a minimization can be formulated as a linear program (LP) and a prediction function with adjustive linear regression is obtained as an optimal solution to the LP. We apply the new machine learning method to a framework of inferring a chemical compound with a desired property. From the results of our computational experiments, we observe that a prediction function constructed by adjustive linear regression for some chemical properties drastically outperforms that by Lasso linear regression.

1 INTRODUCTION

In this paper, we design a new learning method, called "adjustive linear regression" in order to construct a function that predicts a chemical property of a given chemical compound and is further used for molecular design. We start with the background and related work.

Background and Related Work. Computational analysis of chemical compounds recently attracts attentions not only in chemo-informatics but also in bioinformatics and artificial intelligence because it may lead to discovery of novel materials and drugs. Indeed, various machine learning methods have been applied to the prediction of chemical activities from their structural data (Lo et al., 2018; Tetko and Engkvist, 2020). Recently, neural networks and deep-learning technologies have been extensively applied to this problem (Ghasemi et al., 2018).

Prediction of chemical activities from chemical structure data has also been studied for many years in the field of chemoinformatics, which is often referred to as *quantitative structure activity relationship* (QSAR) (Muratov et al., 2020). In addition to QSAR, extensive studies have been done on *inverse quantitative structure activity relationship* (inverse QSAR), which seeks for chemical structures having desired chemical activities under some constraints. In these studies, chemical compounds are usually treated as undirected graphs (called *chemical graphs*) and are further represented as *feature vectors*, each of which is a vector of real and/or integer numbers where its elements are called *descriptors*. One major approach in inverse QSAR is to infer feature vectors from given chemical activities and constraints and then reconstruct chemical structures from these feature vectors (Miyao et al., 2016; Ikebata et al., 2017; Rupakheti et al., 2015).

Artificial neural networks (ANNs) have also been extensively applied to inverse QSAR. For example, recurrent neural networks (Segler et al., 2017; Yang et al., 2017), variational autoencoders (Gómez-Bombarelli et al., 2018), grammar variational autoencoders (Kusner et al., 2017), generative adversarial networks (De Cao and Kipf, 2018), and invertible flow models (Madhawa et al., 2019; Shi et al., 2020) have been applied, where graph convolutional networks (Kipf and Welling, 2016) have been often utilized in such methods to directly handle chemical graphs. However, these methods do not yet guarantee optimal or exact solutions.

Recently, an exact approach has been proposed for

[a] https://orcid.org/0000-0002-2479-3135
[b] https://orcid.org/0000-0002-8332-1517
[c] https://orcid.org/0000-0001-9763-797X

Zhu, J., Haraguchi, K., Nagamochi, H. and Akutsu, T.
Adjustive Linear Regression and Its Application to the Inverse QSAR.
DOI: 10.5220/0010853700003123
In *Proceedings of the 15th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2022) - Volume 3: BIOINFORMATICS*, pages 144-151
ISBN: 978-989-758-552-4; ISSN: 2184-4305

inference of feature vectors from trained ANNs (Shi et al., 2021). This framework (Figure 1) consists of two phases: ANNs are trained using existing methods in the first phase and in the second phase, feature vectors are inferred from desired output values by solving mixed integer linear programming (MILP) that models an inverse problem of the ANN, which is the unique and novel point of this approach. Currently this approach has been further extended so that (i) not only feature vectors but also chemical graphs can be inferred by solving MILP, (ii) other types of prediction functions can be utilized, and (iii) any chemical graphs can be treated. A sophisticated method (Zhu et al., 2021a) has been studied in order to formulate a sparse MILP instance even for a complicated requirements in a topological specification.

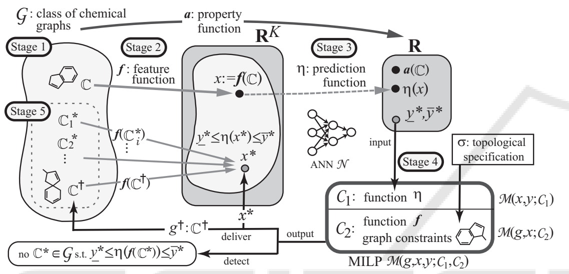
$\mathcal{G}$: class of chemical graphs
$\boldsymbol{a}$: property function
$\mathbf{R}^K$
$\mathbf{R}$
Stage 1
Stage 2
Stage 3
Stage 4
Stage 5
$\boldsymbol{f}$: feature function
η: prediction function
$\mathbb{C}$
$x := \boldsymbol{f}(\mathbb{C})$
$\boldsymbol{a}(\mathbb{C})$
$\eta(x)$
$\underline{y}^*, \overline{y}^*$
$\mathbb{C}_1^*$
$\mathbb{C}_2^*$
$\mathbb{C}^\dagger$
$f(\mathbb{C}_i^*)$
$f(\mathbb{C}^\dagger)$
$\underline{y}^* \leq \eta(x^*) \leq \overline{y}^*$
x^*
ANN $\mathcal{N}$
input
σ: topological specification
C_1: function η
C_2: function $\boldsymbol{f}$ graph constraints
$\mathcal{M}(x,y;C_1)$
$\mathcal{M}(g,x;C_2)$
MILP $\mathcal{M}(g,x,y;C_1,C_2)$
$g^\dagger : \mathbb{C}^\dagger$
deliver
output
detect
no $\mathbb{C}^* \in \mathcal{G}$ s.t. $\underline{y}^* \leq \eta(f(\mathbb{C}^*)) \leq \overline{y}^*$

Figure 1: An illustration of a framework for inferring a set of chemical graphs $\mathbb{C}^*$.

Contribution. In this paper, we develop a novel prediction function and its machine learning method that can be used in the above-mentioned framework. Let us compare linear regression and ANNs. The former uses a hyperplane to explain a given data set and the latter can represent a more complex subspace than a hyperplane. Importantly a best hyperplane that minimizes an error function can be found exactly in the former whereas a local optimum solution to an error function is constructed by an iterative procedure in the latter and different local optimum solutions often appear depending on how we have tuned many parameters in ANNs. Linear regression can be regarded as an ANN on an architecture with an input layer and an output layer of a single node with a linear activation function. We consider an ANN on the same architecture such that each node in the input and output layers is equipped with a set of activation functions. Given a data set, we consider a problem of minimizing an error function on the data set by choosing a weight of each arc, a bias of the output node and a best activation function for each node simultaneously. With some restriction on the set of activation functions and the definition of an error function, we show that such an minimization problem can be formulated as a linear program, which is much easier than an MILP to solve exactly. We call this new method "adjustive linear regression" and implemented it in the two phase framework. We compared adjustive linear regression with Lasso linear regression in constructing prediction functions for several chemical properties. From the results of our computational experiments, we observe that a prediction function constructed by adjustive linear regression for some chemical properties drastically outperforms that by Lasso linear regression. We refer to a full preprint version (Zhu et al., 2021b) for some details on notions, modeling of chemical compounds, the framework and feature function used in this paper.

Organization. The paper is organized as follows. Section 2 reviews the idea of prediction functions based on linear regression and ANNs and designs "adjustive linear regression", a new method for constructing a prediction function by solving a linear program to optimize a choice of weights/bias together with activation functions in an ANN with no hidden layers. Section 3 reviews a method, called a *two-layered model* for representing the feature of a chemical graph in order to deal with an arbitrary graph in the framework. Section 4 reports the results on some computational experiments conducted for the framework of inferring chemical graphs by using our new method of adjustive linear regression. Section 5 makes some concluding remarks.

2 CONSTRUCTING PREDICTION FUNCTIONS

Let $\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{Z}$ and $\mathbb{Z}_+$ denote the sets of reals, non-negative reals, integers and non-negative integers, respectively. For two integers a and b, let $[a,b]$ denote the set of integers i with $a \leq i \leq b$. For a vector $x \in \mathbb{R}^p$, the j-th entry of x is denoted by $x(j)$, $j \in [1,p]$.

2.1 Linear Prediction Functions

For an integer $K \geq 1$, define a feature space $\mathbb{R}^K$. Let $\mathcal{X} = \{x_1, x_2, \ldots, x_m\}$ be a set of feature vectors $x_i \in \mathbb{R}^K$ and let $a_i \in \mathbb{R}$ be a real assigned to a feature vector x_i. Let $A = \{a_i \mid i \in [1,m]\}$. A function $\eta : \mathbb{R}^K \to \mathbb{R}$ is called a *prediction function*. We wish to find a prediction function $\eta : \mathbb{R}^K \to \mathbb{R}$ based on a subset of $\{x_1, x_2, \ldots, x_m\}$ so that $\eta(x_i)$ is closed to the value a_i for many indices $i \in [1,m]$.

For a prediction function $\eta : \mathbb{R}^K \to \mathbb{R}$, define an error function $\mathrm{Err}(\eta; \mathcal{X}) \triangleq \sum_{i \in [1,m]} (a_i - \eta(x_i))^2$, and define the *coefficient of determination* $\mathrm{R}^2(\eta, \mathcal{X})$ to be $1 - \frac{\mathrm{Err}(\eta;\mathcal{X})}{\sum_{i\in[1,m]}(a_i - \widetilde{a})^2}$ for $\widetilde{a} = \frac{1}{m}\sum_{i \in [1,m]} a_i$.

Many methods have been proposed in order to find a prediction function η that minimizes the error function $\mathrm{Err}(\eta_{w,b};\mathcal{X})$ possibly without using all elements in $\mathcal{X}$; e.g., (Zou and Hastie, 2005).

For the feature space $\mathbb{R}^K$, a hyperplane is defined to be a pair (w,b) of a vector $w \in \mathbb{R}^K$ and a real $b \in \mathbb{R}$. A prediction function η is called *linear* if η is given by $\eta_{w,b}(x) = w \cdot x + b, x \in \mathbb{R}^K$ for a hyperplane (w,b). The linear regression is to find a hyperplane (w,b) that minimizes $\mathrm{Err}(\eta_{w,b};\mathcal{X}) = \sum_{i\in[1,m]}(a_i - (w \cdot x_i + b))^2$.

In many cases, a feature vector f contains descriptors that do not play an essential role in constructing a good prediction function. When we solve the minimization problem, the entries $w(j)$ for some descriptors $j \in [1,K]$ in the resulting hyperplane (w,b) become zero, which means that these descriptors were not necessarily important for finding a prediction function $\eta_{w,b}$. It is proposed that solving the minimization with an additional penalty term to the error function often results in a more number of entries $w(j) = 0$, reducing a set of descriptors necessary for defining a prediction function $\eta_{w,b}$. For an error function with such a penalty term, a Ridge function $\frac{1}{2m}\mathrm{Err}(\eta_{w,b};\mathcal{X}) + \lambda[\sum_{j\in[1,K]} w(j)^2 + b^2]$ (Hoerl and Kennard, 1970a; Hoerl and Kennard, 1970b) and a Lasso function $\frac{1}{2m}\mathrm{Err}(\eta_{w,b};\mathcal{X}) + \lambda[\sum_{j\in[1,K]} |w(j)| + |b|]$ (Tibshirani, 1996) are known, where $\lambda \in \mathbb{R}$ is a given real number. As a hybridization of Ridge linear regression and Lasso linear regression, a linear regression that minimizes an error function defined to be $\frac{1}{2m}\mathrm{Err}(\eta_{w,b};\mathcal{X}) + \lambda_2[\sum_{j\in[1,K]} w(j)^2 + b^2] + \lambda_1[\sum_{j\in[1,K]} |w(j)| + |b|]$ is called elastic net linear regression (Zou and Hastie, 2005), where $\lambda_1, \lambda_2 \in \mathbb{R}$ are given real numbers.

2.2 ANNs for Linear Prediction Functions

It is not difficult to see that a linear prediction function η with a hyperplane (w,b) can be represented by an ANN $\mathcal{N}$ with an input layer $L_{\mathrm{in}} = \{u_1, u_2, \ldots, u_K\}$ of K input nodes and an output layer $L_{\mathrm{out}} = \{v\}$ of a single output node v such that the weight of an arc (u_j, v) from an input node u_j to the output node v is given by $w(j), j \in [1,K]$; the bias at node v is given by b; and the activation function at node v is linear. See Figure 2(a) for an illustration of an ANN $\mathcal{N}$ that represents a linear prediction function η with a hyperplane (w,b). Given a vector $x \in \mathbb{R}^K$, the ANN $\mathcal{N}$ outputs $y := \sum_{j\in[1,K]} w(j)x(j) + b$.

We consider an ANN $\mathcal{N}_\phi$ with the same architecture with the ANN $\mathcal{N}$ and introduce activation functions ϕ_j at nodes $u_j, j \in [1,K]$ and an activation function ϕ_0 at node v. Given a vector $x \in \mathbb{R}^K$, the ANN $\mathcal{N}_\phi$

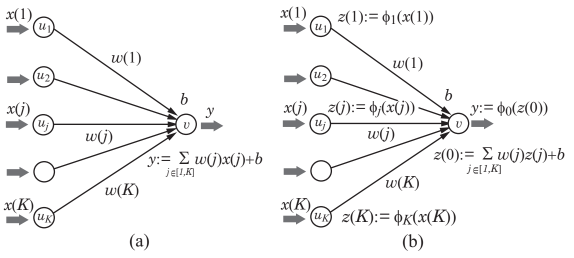

Figure 2: An illustration of the process in ANNs with no hidden layers: (a) An ANN $\mathcal{N}$ that represents a linear prediction function η with a hyperplane (w,b); (b) an ANN $\mathcal{N}_\phi$ with activation functions $\phi_j, j \in [0,K]$ at all nodes.

outputs $y := \phi_0(z(0))$ for $z(0) := \sum_{j\in[1,K]} w(j)z(j) + b$ and $z(j) := \phi_j(x(j)), j \in [1,K]$.

In a standard method of a prediction function $\eta_{\mathcal{N}_\phi}$ with the above ANN $\mathcal{N}_\phi$, we specify each activation function ϕ_j and determine weights w and a bias b by executing an iterative procedure that tries to minimize an error function between the real values a_i and the predicted values $\eta_{\mathcal{N}_\phi}(x_i)$.

2.3 Adjustive Linear Regression

In this paper, we design a new method of constructing a prediction function with the above ANN $\mathcal{N}_\phi$ so that (i) not only weights w and a bias b but also prediction functions ϕ_j are chosen so as to minimize an error function and (ii) the minimization problem is formulated as a linear programming problem.

We introduce a class Φ_j of functions for a choice of each activation function $\phi_j, j \in [0,K]$. When we choose a function $\phi_j \in \Phi_j$ for each $j \in [0,K]$ and a hyperplane (w,b), we define a prediction function $\eta_{\Psi,w,b}$ such that

$$\eta_{\Psi,w,b}(x) \triangleq \phi_0(\sum_{j\in[1,K]} w(j)(\phi_j(x(j))) - b)$$

for the set $\Psi = \{\phi_j \mid j \in [0,K]\}$ of the functions.

In this paper, we use a function $\xi(t) = ct + c't^2 + c''(1-(t-1)^2), 0 \le t \le 1$ for a function $\phi_j, j \in [1,K]$ or the inverse ϕ_0^{-1} of a function ϕ_0, where c, c' and c'' are nonnegative constant constants with $c + c' + c'' = 1$ which will be determined for each $j \in [0,K]$ by our method. Note that, for a domain $0 \le t \le 1$, $\xi(t)$ is a monotone increasing function such that $\xi(0) = 0$ and $\xi(1) = 1$ and admits an inverse function $\xi^{-1}(t)$.

We introduce a class Φ_j of functions in the following way.

1. Normalize the set $\{x_i(j) \mid x_i \in \mathcal{X}\}, j \in [1,K]$ and the set $\{a_i(j) \mid x_i \in \mathcal{X}\}$ so that the minimum and maximum in the set become 0 and 1.
2. For each index $j \in [0,K]$, define a class Φ_j of functions to be

$\Phi_j \triangleq \{c_0(j)t + c_1(j)t^2 + c_2(j)(1-(t-1)^2),$
$0 \le t \le 1 \mid c_q(j) \ge 0, q \in [0,2],$
$\sum_{q\in[0,2]} c_q(j) = 1\}, j \in [1,K],$
and define
$\widetilde{\Phi}_0 \triangleq \{c_0(0)t + c_1(0)t^2 + c_2(0)(1-(t-1)^2),$
$0 \le t \le 1 \mid c_q(0) \ge 0, q \in [0,2],$
$\sum_{q\in[0,2]} c_q(0) = 1\},$
$\Phi_0 \triangleq \{\xi^{-1}(t), 0 \le t \le 1 \mid \xi(t) \in \widetilde{\Phi}_0\}.$

To use linear programming, we measure an error of a prediction function η over a data set $\mathcal{X}$ by the sum of the absolute errors: $\mathrm{SAE}(\eta;\mathcal{X}) \triangleq \sum_{x_i\in\mathcal{X}} |a_i - \eta(x_i)|$.

Now our aim is to find a prediction function $\eta_{\Psi,w,b}$ that minimizes the sum of the absolute errors $\mathrm{SAE}(\eta_{\Psi,w,b};\mathcal{X})$ over all functions $\phi_0 \in \widetilde{\Phi}_0, \phi_j \in \Phi_j, j \in [1,K]$ and hyperplanes (w,b).

To formulate this minimization problem as a linear programming problem, we predetermine the sign of $w(j)$ for each descriptor j in a hyperplane (w,b) that we will choose. Compute the correlation coefficient $\sigma(X_j,A)$ between $X_j = \{x_i(j) \mid i \in [1,m]\}$ and $A = \{a_i \mid i \in [1,m]\}$ and partition the set of descriptors into two sets $I^+ := \{j \in [1,K] \mid \sigma(X_j,A) \ge 0\}$ and $I^- := \{j \in [1,K] \mid \sigma(X_j,A) < 0\}$. We impose an additional constraint that $w(j) \ge 0, j \in I^+$ and $w(j) \le 0, j \in I^-$. Then the objective function is described as follows, where we rewrite each term $w(j), j \in I^+$ (resp., $-w(j), j \in I^-$) as $w'(j)$:

$$\sum_{i\in[1,m]} \Big|c_0(0)a_i + c_1(0)a_i^2 + c_2(0)(1-(a_i-1)^2)$$
$$- \sum_{j\in I^+} [w'(j)\big(c_0(j)x_i(j) + c_1(j)x_i(j)^2$$
$$+ c_2(j)(1-(x_i(j)-1)^2)\big)]$$
$$+ \sum_{j\in I^-} [w'(j)\big(c_0(j)x_i(j) + c_1(j)x_i(j)^2$$
$$+ c_2(j)(1-(x_i(j)-1)^2)\big)] - b\Big|.$$

We minimize this over all nonnegative reals $c_q(j)$, $q \in [0,2], j \in [1,K]$, nonnegative reals $w(j), j \in [1,K]$ and a real $b \in \mathbb{R}$ such that $\sum_{q\in[0,2]} c_q(j) = 1, j \in [1,K]$.

By introducing a penalty term for the weights $w(j), j \in [1,K]$, we consider the following problem which we call *adjustive linear regression* $\mathrm{ALR}(\mathcal{X},\lambda)$, where $w'(j)c_q(j), q \in [0,2]$ is rewritten as $w_q(j)$.

$$\text{min: } \frac{1}{2m}\sum_{i\in[1,m]} |c_0(0)a_i + c_1(0)a_i^2 + c_2(0)(1-(a_i-1)^2)$$
$$-\sum_{j\in I^+} [w_0(j)x_i(j) + w_1(j)x_i(j)^2 + w_2(j)(1-(x_i(j)-1)^2)]$$
$$+\sum_{j\in I^-} [w_0(j)x_i(j) + w_1(j)x_i(j)^2 + w_2(j)(1-(x_i(j)-1)^2)]$$
$$-b| + \lambda \sum_{j\in[1,K]} w_0(j) + \lambda|b|$$

subject to $c_0(0) + c_1(0) + c_2(0) = 1$.

We observe that adjustive linear regression is an extension of the Lasso linear regression except that the error function is the sum of absolute errors in the former and the sum of square errors in the latter. It is not difficult to see that the above minimization can be formulated as a linear program with $O(m+K)$ variables and constraints. In our experiment, we also penalize each weight $w_q(j), q \in [1,2]$ with the same constant λ in a similar fashion to Lasso linear regression..

We solve the above minimization problem to construct a prediction function $\eta_{\Psi,w,b}$. Let $c_q^*(0), q \in [0,2]$, $w_q^*(j), q \in [0,2], j \in [1,K]$ and b^* denote the values of variables $c_q(0), q \in [0,2]$, $w_q(j), q \in [0,2], j \in [1,K]$ and b in an optimal solution, respectively. Let K' denote the number of descriptors $j \in [1,K]$ with $w_0^*(j) > 0$ and $I_{K'}$ denote the set of $j \in [1,K]$ with $w_0^*(j) > 0$. Then we set
$w^*(j) := 0, j \in [1,K]$ with $w_0^*(j) = 0$,
$w^*(j) := w_0^*(j)/(w_0^*(j) + w_1^*(j) + w_2^*(j)), j \in I^+ \cap I_{K'}$,
$w^*(j) := -w_0^*(j)/(w_0^*(j) + w_1^*(j) + w_2^*(j)), j \in I^- \cap I_{K'}$,
$c_q^*(j) := w_q^*(j)/w^*(j), q \in [0,2], j \in I_{K'}$ and
$w^* := (w_0^*(1), w_0^*(2), \ldots, w_0^*(K)) \in \mathbb{R}^K$.

For a set Ψ^* of selected functions $\phi_j(t) = c_0^*(j)t + c_1^*(j)t^2 + c_2^*(j)(1-(t-1)^2), j \in I_{K'}$ with and $\phi_0(t)$ with $\phi_0^{-1}(t) = c_0^*(0)t + c_1^*(0)t^2 + c_2^*(0)(1-(t-1)^2)$ and a hyperplane (w^*, b^*), we construct a prediction function η_{Ψ^*,w^*,b^*}.

We propose the following scheme of executing ALR for constructing a prediction function and evaluating the performance.

1. Given a data set $\mathcal{X} = \{x_i \in \mathbb{R}^K \mid i \in [1,m]\}$ of normalized feature vectors and a set $A = \{a_i \in \mathbb{R} \mid i \in [1,m]\}$ of normalized observed values, we choose a real $\lambda > 0$ possibly from a set of candidates for $\lambda > 0$ so that the performance of a prediction function η_{Ψ^*,w^*,b^*} obtained from an optimal solution (Ψ^*, w^*, b^*) to the ALR $(\mathcal{X},\lambda)$ attains a criterion, where we may use cross-validation and the test coefficient of determination to know the performance.
2. With the real λ determined in 1, we evaluate the performance of a prediction function obtained with ALR based on cross-validation. We divide the entire set $\mathcal{X}$ into five subsets $\mathcal{X}^{(k)}, k \in [1,5]$. For each $k \in [1,5]$, we use the set $\mathcal{X} \setminus \mathcal{X}^{(k)}$ as a training data to construct a prediction function $\eta_{\Psi,w,b}$ with ALR $(\mathcal{X} \setminus \mathcal{X}^{(k)}, \lambda)$ and compute the coefficient of determination $\mathrm{R}^2(\eta_{\Psi,w,b};\mathcal{X}^{(k)})$.

3 TWO-LAYERED MODEL

Let $\mathbb{C} = (H, \alpha, \beta)$ be a chemical graph (see (Zhu et al., 2021b) for the detail) and $\rho \geq 1$ be an integer, which we call a *branch-parameter*.

A *two-layered model* of $\mathbb{C}$ is a partition of the hydrogen-suppressed chemical graph $\langle\mathbb{C}\rangle$ into an "interior" and an "exterior" in the following way. We call a vertex $v \in V(\langle\mathbb{C}\rangle)$ of $\mathbb{C}$ an *exterior-vertex* if $\mathrm{ht}(v) < \rho$ (see (Zhu et al., 2021b) for the detail of $\mathrm{ht}(v)$). Let $V^{\mathrm{ex}}_{\rho}(\mathbb{C})$ denote the set of exterior-vertices. We call an edge $e \in E(\langle\mathbb{C}\rangle)$ of $\mathbb{C}$ *exterior-edge* if e is incident to an exterior-vertex. Let $E^{\mathrm{ex}}_{\rho}(\mathbb{C})$ denote the set of exterior-edges. Define $V^{\mathrm{int}}_{\rho}(\mathbb{C}) \triangleq V(\langle\mathbb{C}\rangle) \setminus V^{\mathrm{ex}}_{\rho}(\mathbb{C})$ and $E^{\mathrm{int}}_{\rho}(\mathbb{C}) \triangleq E(\langle\mathbb{C}\rangle) \setminus E^{\mathrm{ex}}_{\rho}(\mathbb{C})$. We call a vertex in $V^{\mathrm{int}}_{\rho}(\mathbb{C})$ (resp., an edge in $E^{\mathrm{int}}_{\rho}(\mathbb{C})$) an *interior-vertex* (resp., *interior-edge*). The set $E^{\mathrm{ex}}_{\rho}(\mathbb{C})$ of exterior-edges forms a collection of connected graphs each of which is regarded as a rooted tree T rooted at the vertex $v \in V(T)$ with the maximum $\mathrm{ht}(v)$. Let $\mathcal{T}^{\mathrm{ex}}_{\rho}(\langle\mathbb{C}\rangle)$ denote the set of these chemical rooted trees in $\langle\mathbb{C}\rangle$. The *interior* $\mathbb{C}^{\mathrm{int}}_{\rho}$ of $\mathbb{C}$ is defined to be the subgraph $(V^{\mathrm{int}}_{\rho}(\mathbb{C}), E^{\mathrm{int}}_{\rho}(\mathbb{C}))$ of $\langle\mathbb{C}\rangle$.

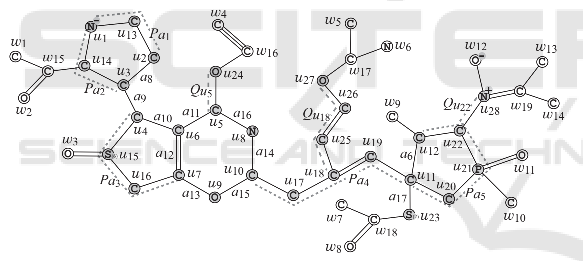
Figure 3: An illustration of a hydrogen-suppressed chemical graph $\langle\mathbb{C}\rangle$ obtained from a chemical graph $\mathbb{C}$ by removing all the hydrogens, where for $\rho = 2$, $V^{\mathrm{ex}}_{\rho}(\mathbb{C}) = \{w_i \mid i \in [1,19]\}$ and $V^{\mathrm{int}}_{\rho}(\mathbb{C}) = \{u_i \mid i \in [1,28]\}$.

Figure 3 illustrates an example of a hydrogen-suppressed chemical graph $\langle\mathbb{C}\rangle$. For a branch-parameter $\rho = 2$, the interior of the chemical graph $\langle\mathbb{C}\rangle$ in Figure 3 is obtained by removing the set of vertices with degree 1 $\rho = 2$ times; i.e., first remove the set $V_1 = \{w_1, w_2, \ldots, w_{14}\}$ of vertices of degree 1 in $\langle\mathbb{C}\rangle$ and then remove the set $V_2 = \{w_{15}, w_{16}, \ldots, w_{19}\}$ of vertices of degree 1 in $\langle\mathbb{C}\rangle - V_1$, where the removed vertices become the exterior-vertices of $\langle\mathbb{C}\rangle$.

For each interior-vertex $u \in V^{\mathrm{int}}_{\rho}(\mathbb{C})$, let $T_u \in \mathcal{T}^{\mathrm{ex}}_{\rho}(\langle\mathbb{C}\rangle)$ denote the chemical tree rooted at u (where possibly T_u consists of vertex u) and define the ρ-*fringe-tree* $\mathbb{C}[u]$ to be the chemical rooted tree obtained from T_u by putting back the hydrogens originally attached with T_u in $\mathbb{C}$. Let $\mathcal{T}(\mathbb{C})$ denote the set of ρ-fringe-trees $\mathbb{C}[u], u \in V^{\mathrm{int}}_{\rho}(\mathbb{C})$. Figure 4 illustrates the set $\mathcal{T}(\mathbb{C}) = \{\mathbb{C}[u_i] \mid i \in [1,28]\}$ of the 2-fringe-trees of the example $\mathbb{C}$ with $\langle\mathbb{C}\rangle$ in Figure 3.

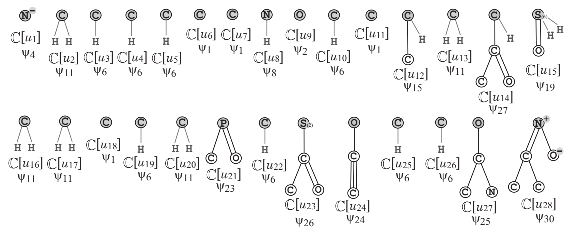
Figure 4: The set $\mathcal{T}(\mathbb{C})$ of 2-fringe-trees $\mathbb{C}[u_i], i \in [1,28]$ of the example $\mathbb{C}$ with $\langle\mathbb{C}\rangle$in Figure 3, where the hydrogens attached to non-root vertices are omitted in the figure.

Topological Specification. A topological specification is described as a set of the following rules proposed (Shi et al., 2021):

(i) a *seed graph* G_{C} as an abstract form of a target chemical graph $\mathbb{C}$;
(ii) a set $\mathcal{F}$ of chemical rooted trees as candidates for a tree $\mathbb{C}[u]$ rooted at each interior-vertex u in $\mathbb{C}$; and
(iii) lower and upper bounds on the number of components in a target chemical graph such as chemical elements, double/triple bonds and the interior-vertices in $\mathbb{C}$.

See (Zhu et al., 2021b) for a full description and example of topological specification.

4 RESULTS

We implemented our method of Stages 1 to 5 (see (Zhu et al., 2021b) for the detail) for inferring chemical graphs under a given topological specification and conducted experiments to evaluate the computational efficiency. We executed the experiments on a PC with Processor: Core i7-9700 (3.0GHz; 4.7 GHz at the maximum) and Memory: 16 GB RAM DDR4. We used scikit-learn version 0.23.2 with Python 3.8.5 for executing linear regression with Lasso function or constructing an ANN. To solve an LP or MILP instance, we used CPLEX version 12.10.

Results on Phase 1. We implemented Stages 1, 2 and 3 in Phase 1 as follows.

We have conducted experiments of adjustive linear regression and for 37 chemical properties of monomers (resp., ten chemical properties of polymers) and we found that the test coefficient of determination R^2 of ALR exceeds 0.6 for 28 properties of monomers: isotropic polarizability (ALPHA) and boiling point (BP), critical pressure (CP); critical temperature (CT); heat capacity at 298.15K (CV);

dissociation constants (Dc); electron density on the most positive atom (EDPA); flash point (Fp); energy difference between the highest and lowest unoccupied molecular orbitals (Gap); heat of atomization (Ha); heat of combustion (Hc); heat of formation (Hf); energy of highest occupied molecular orbital (Homo); heat of vaporization (Hv); isobaric heat capacities in liquid phase (IhcL); isobaric heat capacities in solid phase (IhcS); Kováts retention index (KvI); octanol/water partition coefficient (Kow); lipophilicity (Lp); energy of lowest unoccupied molecular orbital (Lumo); melting point (Mp); optical rotation (OptR); refractive index (Rf); solubility (Sl); surface tension (SfT); internal energy at 0K (U0); viscosity (Vis); and vapor density (Vd) and that the test coefficient of determination R^2 of ALR exceeds 0.8 for eight properties of polymers: experimental amorphous density (AmD); characteristic ratio (ChaR); dielectric constant(DeC); heat capacity liquid (HcL); heat capacity solid (HcS); mol volume (MlV); refractive index (RfId); and glass transition (Tg), where we include the result of property permittivity (Prm) for a comparison with Lasso linear regression and ANN.

Stage 1. We set a graph class $\mathcal{G}$ to be the set of all chemical graphs with any graph structure, and set a branch-parameter ρ to be 2.

For each of the properties, we first select a set Λ of chemical elements and then collect a data set D_π on chemical graphs over the set Λ of chemical elements.

Table 1 shows the size and range of data sets that we prepared for each chemical property in Stage 1, where we denote the following: $|\Lambda|$: the size $|\Lambda|$ of Λ used in the data set D_π;

$|D_\pi|$: the size of data set D_π over Λ for the property π; and K: the number of descriptors in a feature vector $f(\mathbb{C})$.

Stage 2. We used the feature function defined in our chemical model without suppressing hydrogen. We standardize the range of each descriptor and the range of property values $a(\mathbb{C}), \mathbb{C} \in D_\pi$.

Stage 3. For each chemical property π, we select a penalty value λ_π for a constant λ in ALR$(\mathcal{X}, \lambda)$ by conducting linear regression as a preliminary experiment.

We conducted an experiment in Stage 3 to evaluate the performance of the prediction function based on cross-validation. For a property π, an execution of a *cross-validation* consists of five trials of constructing a prediction function as follows.

Tables 1 and 2 show the results on Stages 2 and 3 for the properties on monomers and polymers, respectively, where we denote the following: time: the average time (sec.) to construct a prediction function with ALR by solving an LP with $O(|D_\pi|+K)$ variables and constraints over all 50 trials in ten cross-validations; ALR: the median of test R^2 over all 50 trials in ten cross-validations for prediction functions constructed with ALR; LLR: the median of test R^2 over all 50 trials in ten cross-validations for prediction functions constructed with Lasso linear regression; and ANN: the median of test R^2 over all 50 trials in ten cross-validations for prediction functions constructed with ANNs (see (Zhu et al., 2021b) for the details of constructing a prediction function with ANNs).

Table 1: Results in Phase 1 for monomers.

π	$\|\Lambda\|$	$\|D_\pi\|$	K	time	ALR	LLR	ANN
Alpha	10	977	297	3.00	0.953	0.961	0.888
Bp	4	370	184	1.42	0.816	0.599	0.765
Bp	7	444	230	2.02	0.832	0.663	0.720
Cp	4	125	112	0.15	0.650	0.445	0.694
Cp	6	131	119	0.12	0.690	0.555	0.727
Ct	4	125	113	0.24	0.900	0.037	0.357
Ct	6	132	121	0.28	0.895	0.048	0.356
Cv	10	977	297	4.57	0.966	0.970	0.911
Dc	7	161	130	0.35	0.602	0.574	0.622
EDPA	3	52	64	0.06	0.999	0.999	0.992
Fp	4	36	183	1.31	0.719	0.589	0.746
Fp	7	424	229	1.92	0.684	0.571	0.745
Gap	10	977	297	4.77	0.755	0.784	0.795
Ha	4	115	115	0.29	0.998	0.997	0.926
Hc	4	255	154	0.74	0.986	0.946	0.848
Hc	7	282	177	0.84	0.986	0.951	0.903
Hf	3	82	74	0.05	0.982	0.987	0.928
Homo	10	977	297	4.95	0.689	0.841	0.689
Hv	4	95	105	0.19	0.626	-13.7	-8.44
IhcL	4	770	256	3.24	0.987	0.986	0.974
IhcL	7	865	316	1.98	0.989	0.985	0.971
IhcS	7	581	192	1.72	0.971	0.985	0.971
IhcS	11	668	228	2.21	0.974	0.982	0.968
KvI	3	52	64	0.05	0.838	0.677	0.727
Kow	4	684	223	3.13	0.964	0.953	0.952
Kow	8	899	303	4.95	0.952	0.927	0.937
Lp	4	615	186	1.81	0.844	0.856	0.867
Lp	8	936	231	3.37	0.807	0.840	0.859
Lumo	10	977	297	2.75	0.833	0.841	0.860
Mp	4	467	197	1.78	0.831	0.810	0.799
Mp	8	577	255	2.99	0.807	0.810	0.820
OptR	4	147	107	0.24	0.876	0.825	0.919
OptR	6	157	123	0.27	0.870	0.825	0.878
Rf	4	166	142	0.24	0.685	0.619	0.521
Sl	4	673	217	1.21	0.784	0.808	0.848
Sl	8	915	300	2.33	0.828	0.808	0.861
SfT	4	247	128	0.67	0.847	0.927	0.859
U0	10	977	297	2.40	0.995	0.999	0.890
Vis	4	282	126	0.37	0.911	0.893	0.929
Vd	4	474	214	2.24	0.985	0.927	0.912
Vd	7	551	256	2.28	0.980	0.942	0.889

Table 2: Results in Phase 1 for polymers.

π	$\lvert\Lambda\rvert$	$\lvert D_\pi\rvert$	K	time	ALR	LLR	ANN
AmD	4	86	83	0.09	0.933	0.914	0.885
AmD	7	93	94	0.10	0.917	0.918	0.823
ChaR	3	24	56	0.02	0.904	0.650	0.616
ChaR	4	27	67	0.03	0.835	0.431	0.641
DeC	7	37	72	0.04	0.918	0.761	0.641
HcL	4	52	67	0.06	0.996	0.990	0.969
HcL	7	55	81	0.05	0.992	0.987	0.970
HcS	4	54	75	0.07	0.963	0.968	0.893
HcS	7	59	92	0.09	0.983	0.961	0.880
MlV	4	86	83	0.10	0.998	0.996	0.931
MlV	7	93	94	0.09	0.997	0.994	0.894
Prm	4	112	69	0.09	0.505	0.801	0.801
Prm	5	131	73	0.09	0.489	0.784	0.735
RfId	5	91	96	0.15	0.953	0.852	0.871
RfId	7	124	124	0.21	0.956	0.832	0.891
Tg	4	204	101	0.23	0.923	0.902	0.883
Tg	7	232	118	0.54	0.927	0.894	0.881

From Tables 1 and 2, we see that ALR performs well for most of the properties in our experiments, The performance by ALR is inferior to that by LLR or ANN for some properities such as Gap, Homo, Lumo, OptR, Sl, SfT and Prm, whereas ALR outperforms LLR and ANN for properties Bp, Ct, Hv, KvI, Vd, ChaR, RfId and Tg. It should be noted that ALR drastically improves the result for properties Ct and Hv.

Results on Phase 2. To execute Stages 4 and 5 in Phase 2, we used a set of seven instances $I_{\rm a}$, $I_{\rm b}^i, i \in [1,4]$, $I_{\rm c}$ and $I_{\rm d}$ based on the seed graphs prepared in (Shi et al., 2021).

Stage 4. We executed Stage 4 for heat of vaporization (Hv).

Table 3 shows the computational results of the experiment in Stage 4 for the two properties, where we denote the following:

$\underline{y}^*$, $\overline{y}^*$: lower and upper bounds $\underline{y}^*, \overline{y}^* \in \mathbb{R}$ on the value $a(\mathbb{C})$ of a chemical graph $\mathbb{C}$ to be inferred; I-time: the time (sec.) to solve the MILP in Stage 4; and n: the number $n(\mathbb{C}^\dagger)$ of non-hydrogen atoms in the chemical graph $\mathbb{C}^\dagger$ inferred in Stage 4.

The result suggests that ALR is useful to infer relatively large size chemical graphs from given chemical properties. Note that hydrogen atoms can be recovered after getting hydrogen-suppressed chemical graphs.

Stage 5. We executed Stage 5 to generate a more number of target chemical graphs $\mathbb{C}^*$, where we call a chemical graph $\mathbb{C}^*$ a *chemical isomer* of a target chemical graph $\mathbb{C}^\dagger$ of a topological specification σ if $f(\mathbb{C}^*) = f(\mathbb{C}^\dagger)$ and $\mathbb{C}^*$ also satisfies the same topological specification σ. We computed chemical isomers $\mathbb{C}^*$ of each target chemical graph $\mathbb{C}^\dagger$ inferred in Stage 4. We execute the algorithm due to (Shi et al., 2021) to generate chemical isomers of $\mathbb{C}^\dagger$ up to 100 when the number of all chemical isomers exceeds 100.

Table 3: Results of Stages 4 and 5 for Hv.

inst.	$\underline{y}^*$, $\overline{y}^*$	I-time	n	D-time	$\mathbb{C}$-LB	#$\mathbb{C}$
$I_{\rm a}$	145, 150	24.9	37	0.0632	2	2
$I_{\rm b}^1$	190, 195	146.6	35	0.121	30	30
$I_{\rm b}^2$	290, 295	188.8	46	0.154	604	100
$I_{\rm b}^3$	165, 170	1167.2	45	36.8	7.5×10^6	100
$I_{\rm b}^4$	250, 255	313.7	50	0.166	2208	100
$I_{\rm c}$	285, 290	102.5	50	0.016	1	1
$I_{\rm d}$	245, 250	351.9	40	5.53	3.9×10^5	100

The algorithm can compute a lower bound on the total number of all chemical isomers $\mathbb{C}^\dagger$ without generating all of them.

Table 3 shows the computational results of the experiment in Stage 5 for property Hv, where we denote the following: D-time: the running time (sec.) to execute the dynamic programming algorithm in Stage 5 to compute a lower bound on the number of all chemical isomers $\mathbb{C}^*$ of $\mathbb{C}^\dagger$ and generate all (or up to 100) chemical isomers $\mathbb{C}^*$; $\mathbb{C}$-LB: a lower bound on the number of all chemical isomers $\mathbb{C}^*$ of $\mathbb{C}^\dagger$; and #$\mathbb{C}$: the number of all (or up to 100) chemical isomers $\mathbb{C}^*$ of $\mathbb{C}^\dagger$ generated in Stage 5. The result suggests that ALR is useful not only for inference of chemical graphs but also for enumeration of chemical graphs.

5 CONCLUDING REMARKS

In this paper, we proposed a new machine learning method, adjustive linear regression (ALR), which has the following feature: (i) ALR is an extension of the Lasso linear regression except for the definition of error functions; (ii) ALR is a special case of an ANN except that a choice of activation functions is also optimized differently from the standard ANNs and the definition of error functions; and (iii) ALR can be executed exactly by solving the equivalent linear program with $O(m+K)$ variables and constraints for a set of m data with K descriptors. Even though ALR is a special case of an ANN with non-linear activation functions, we still can read the relationship between cause and effect from a prediction function due to the simple structure of ALR.

In this paper, we used a quadratic function for a set Ψ of activation functions ϕ. We can use many different functions such as sigmoid function and ramp functions, where the non-linearity of a function does not affect to derive a linear program for ALR. The proposed method/system is available at GitHub https://github.com/ku-dml/mol-infer.

REFERENCES

De Cao, N. and Kipf, T. (2018). Molgan: An implicit generative model for small molecular graphs. arXiv:1805.11973.

Ghasemi, F., Mehridehnavi, A., Pérez-Garrido, A., and Pérez-Sánchez, H. (2018). Neural network and deep-learning algorithms used in qsar studies: merits and drawbacks. *Drug Discovery Today*, 23:1784–1790.

Gómez-Bombarelli, R., Wei, J., Duvenaud, D., Hernández-Lobato, J., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T., Adams, R., and Aspuru-Guzik, A. (2018). Automatic chemical design using a data-driven continuous representation of molecules. *ACS Cent. Sci.*, 4:268–276.

Hoerl, A. and Kennard, R. (1970a). Ridge regression: Applications to nonorthogonal problems. *Technometrics*, 12(1):69–82.

Hoerl, A. and Kennard, R. (1970b). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Ikebata, H., Hongo, K., Isomura, T., Maezono, R., and Yoshida, R. (2017). Bayesian molecular design with a chemical language model. *J. Comput. Aided Mol. Des.*, 31:379–391.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv:1609.02907.

Kusner, M., Paige, B., and Hernández-Lobato, J. (2017). Grammar variational autoencoder. In *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*, volume 70, pages 1945–1954, Sydney, NSW, Australia. JMLR.org.

Lo, Y.-C., Rensi, S., Torng, W., and Altman, R. (2018). Machine learning in chemoinformatics and drug discovery. *Drug Discovery Today*, 23:1538–1546.

Madhawa, K., Ishiguro, K., Nakago, K., and Abe, M. (2019). Graphnvp: an invertible flow model for generating molecular graphs. arXiv:1905.11600.

Miyao, T., Kaneko, H., and Funatsu, K. (2016). Inverse QSPR/QSAR analysis for chemical structure generation (from y to x). *J. Chem. Inf. Model.*, 56:286–299.

Muratov, E. N., Bajorath, J., Sheridan, R., Tetko, I., Filimonov, D., Poroikov, V., Oprea, T., Baskin, I., Varnek, A., Roitberg, A., Isayev, O., Curtarolo, S., Fourches, D., Cohen, Y., Aspuru-Guzik, A., Winkler, D., Agrafiotis, D., Cherkasov, A., and Tropsha, A. (2020). QSAR without borders. *Chemical Society Reviews*, 49(11):3525–3564.

Rupakheti, C., Virshup, A., Yang, W., and Beratan, D. (2015). Strategy to discover diverse optimal molecules in the small molecule universe. *J. Chem. Inf. Model.*, 55:529–537.

Segler, M., Kogej, T., Tyrchan, C., and Waller, M. (2017). Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS Cent. Sci.*, 4:120–131.

Shi, C., Xu, M., Zhu, Z., Zhang, W., Zhang, M., and Tang, J. (2020). GraphAF: a flow-based autoregressive model for molecular graph generation. arXiv:2001.09382.

Shi, Y., Zhu, J., Azam, N. A., Haraguchi, K., Zhao, L., Nagamochi, H., and Akutsu, T. (2021). An inverse qsar method based on a two-layered model and integer programming. *International Journal of Molecular Sciences*, 22:2847.

Tetko, I. and Engkvist, O. (2020). From big data to artificial intelligence: chemoinformatics meets new challenges. *J. Cheminformatics*, 12:74.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *J. R. Statist. Soc. B*, 58:267–288.

Yang, X., Zhang, J., Yoshizoe, K., Terayama, K., and Tsuda, K. (2017). ChemTS: an efficient python library for de novo molecular generation. *STAM*, 18:972–976.

Zhu, J., Azam, N. A., Haraguchi, K., Zhao, L., Nagamochi, H., and Akutsu, T. (2021a). An improved integer programming formulation for inferring chemical compounds with prescribed topological structures. In *The 34th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE 2021)*, pages 197–209, Kuala Lumpur, Malaysia. Springer.

Zhu, J., Haraguchi, K., Nagamochi, H., and Akutsu, T. (2021b). Adjustive linear regression and its application to the inverse qsar. Department of Applied Mathematics and Physics, Kyoto University, Technical Report, TR:2021-002 http://www.amp.i.kyoto-u.ac.jp/tecrep.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *J. R. Statist. Soc. B*, 67(2):301–320.