平成30年度 修士論文

# 小規模データでも利用可能な深層生成モデルによる医薬品構造提案手法の開発

東京大学大学院工学系研究科化学システム工学専攻
船津・小寺研究室

37-176695 鈴木天音

# 目次

# 第 1 章

# 序論

## 1.1 背景

創薬の現場において、構造の探索は一般に化学者・薬学者の知見に基づいて行われている。一方で実験には一定の時間およびコストがかかることから、実験サイクルの最適化が求められている。

実際に実験を行うことなく活性を予測する手法として、化合物の特徴を表す記述子と活性値との関係を統計的にモデリングする定量的構造活性相関 (Quantitative Structure-Activity Relationships, QSAR) が存在する。一般的な QSAR は以下のステップに基づいて行われる。

1. 化学構造から特徴を抽出した記述子を計算
2. 記述子と活性値との間にモデルを構築

この方法に基づいて構築したモデルを利用すると、化学構造から活性値を予測することができる。

一方、活性値から対応する化学構造を得る枠組みを inverse-QSAR という。図 1.1 に QSAR と inverse-QSAR の流れを示す。望ましい活性を示すような化学構造を得る inverse-QSAR を考えた際には 2 つの問題点が存在する。1 つ目は一般的な QSAR モデルは複雑な非線形関数であるため、活性値から記述子を得るような逆関数が明示的に得られず、逆解析が困難なことである。もう 1 つは記述子が得られた場合でもその記述子を持つような化学構造を組み立てるのが困難であるということである。これらの問題を解決するため、宮尾らは Gaussian Mixture Model (GMM) を利用して逆関数を明示的に得る手法 [1] や Differencial Evolution を用いて非線形モデルに対して望ましい記述子を得る手法 [2] を提案している。

近年盛んに研究されているアプローチは、深層生成モデル (後述) を用い、記述子を計算することなく望ましい活性を持つ化学構造を直に得るという手法である。Gómez-Bombarelli らは Variational Autoencoder (VAE) [3, 4] を用いて、化学構造の線形表記の 1 種である SMILES から連続値の潜在表現を得る手法を提案した [5]。潜在変数空間中の 1 点を指定すればそこから直に SMILES 表記された化学構造を得ることができることを示したほか、

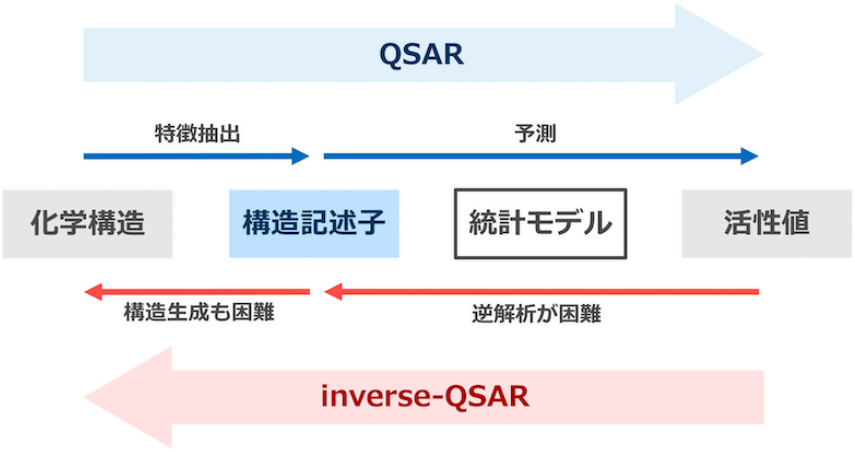



図 1.1 QSAR と inverse-QSAR

いくつかの物性について潜在変数空間を探索することで最適化が可能であることを確認している。Blaschke らは VAE に加え、Adversarial Autoencoders (AAE) [6] というモデルを採用した。深層生成モデルを学習させた後に Dopamine Receptor D2 (DRD2) のデータセットに対して実際に構造最適化を行い、活性が高いと思われる構造を取得することに成功している [7]。

## 1.2 本研究の目的と方針

本研究では、深層生成モデルを利用したいずれの先行研究も検証を大規模データで行っていることに着目した。Gómez-Bombarelli ら [5] は ZINC データベースから抽出した 250,000 件のデータを、Blaschke ら [7] は ExCAPE-DB から抽出した 350,422 件のデータを用いている。これらはいずれも教師付き（物性・活性値の測定されている）データである。一方でタンパク質を特定した場合、教師付きデータは数十件から数百件程度しか存在しないことも多い。教師付きデータが少ない場合は深層生成モデルはうまく学習することができないため、少数データに対して利用可能な深層生成モデルは未だ提案されていない。そこで、本研究では教師付きデータが少ない場合でも良好な構造提案が行える手法の提案を目的とした。

方針としては、VAE と AAE を半教師付き学習的に利用した。半教師付き学習とは、多数の教師なしデータを利用してデータの分布を学習し、学習した分布を用いて教師との関係を学習する方式である。半教師付き学習を利用すると、手元に実験値が存在するデータがわずかであっても、Web 上で公開されているオープンデータの化学構造と組み合わせて学習を行うことができる。

また、化学データに限らなければ Generative Adversarial Positive-Unlabeled Learning (PU-GAN) [8] という半教師付き学習の手法が存在する。これも化学データに適用し、結果を検証した。

# 第 2 章

# 手法

## 2.1 深層学習

本説では、深層学習の基礎的な用語について述べる。

### 2.1.1 ニューラルネットワーク

生物の神経回路網を模倣した統計モデルをニューラルネットワークと言い、多層のニューラルネットワークを利用してモデルを構築する手法を一般に深層学習と呼ぶ。ニューロンと呼ばれる単純な計算ユニットを多数連結させることでモデルに十分な自由度を持たせ、複雑な関数を表現することを可能にしている。入力が入ってくる 1 層目を入力層、出力を行う最終層を出力層、残りのすべての層を隠れ層と呼ぶ。

各層の全ニューロンが次の層の全ニューロンと結合を持っているニューラルネットワークを全結合ニューラルネットワークという。図 2.1 に全結合ニューラルネットワークの概略図を示す。全結合ニューラルネットワークの各層を全結合層 (Fully Connected Layer, Dense Layer) と呼ぶ。

ニューラルネットワークには結合の様式によって図 2.1 に示した全結合ニューラルネットワークのほか、画像データのモデリングに用いる畳み込みニューラルネットワーク (Convolution Neural Network)、系列データのモデリングに用いる再帰的ニューラルネットワーク (Recurrent Neural Network) などの種類がある。畳み込みニューラルネットワークの各層を畳み込み層 (Convolution Layer)、再帰的ニューラルネットワークの各層を再帰的層 (Recurrent Layer) とそれぞれ呼ぶ。

### 2.1.2 誤差逆伝播法

ニューラルネットワークを学習するためには、全ての層が持つパラメータを決定する必要がある。そこで、確率的最急降下法 (Stochastic Gradient Decent Method, SGD) を用いて最適化を行う。

SGD では、モデルを評価する損失関数 $E$ を設定し、各パラメータ $\theta$ における勾配 $\frac{\partial E}{\partial \theta}$ を

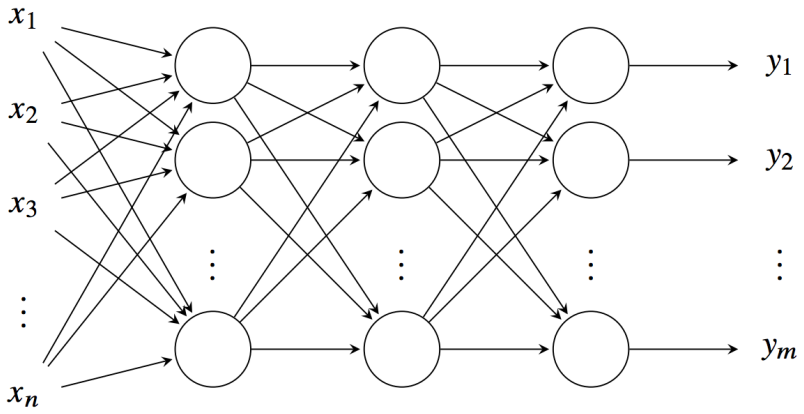



図 2.1 全結合ニューラルネットワークの概略図

求め、$E$ を最小化する方向に各パラメータ $\theta$ を少しずつ調整する。勾配 $\frac{\partial E}{\partial \theta}$ を求める手法として誤差逆伝播法 (Back Propagation) [9] が存在する。合成関数の微分における連鎖律を利用することで出力に近い層から順に勾配を計算することが可能である。

1. 各層を通してニューラルネットワークの予測値を得る（順伝播）
2. 予測値と実測値を元に損失関数 $E$ を計算する
3. 誤差逆伝播の結果を元に各層のパラメータ $\theta$ を調節する

SGD の拡張にあたる最適化手法として、Adagrad[10] や Adam[11] が存在する。

### 2.1.3 活性化関数

ニューラルネットワークは複数の層を通して入力と出力との関係をモデリングする。この際、非線形な関係を学習するために導入するのが活性化関数である。

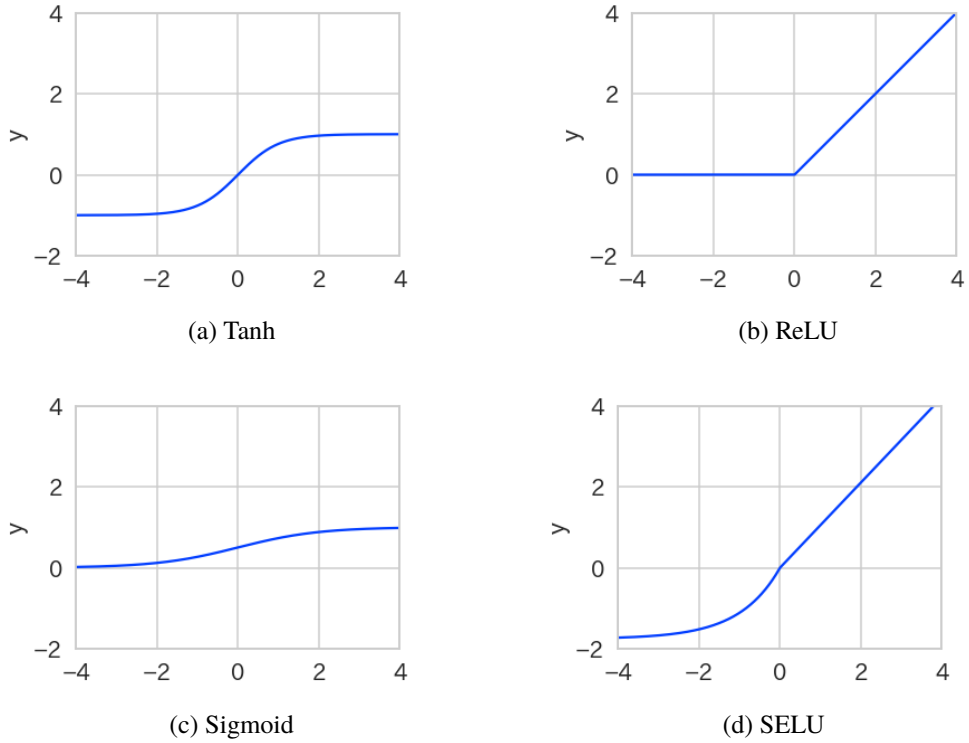
活性化関数は各層の出力にかける関数で、一般には図 2.2 に示すような Tanh や ReLU[12]、Sigmoid[13] や SELU[14] が用いられる。

### 2.1.4 多クラス分類への応用

ニューラルネットワークで多クラス分類を行う場合、出力が各クラスの確率として解釈できる必要がある。そこで、$N$ クラス分類を行う場合は出力層の次元を $N$ とし、Softmax と呼ばれる変換を施す (式 2.1)。

$$y_i = \frac{\exp(x_i)}{\sum_{j=1}^{N} \exp(x_j)} \tag{2.1}$$

$x_i$ は $[-\infty, \infty]$ を取るが、exp をかけることで全ての出力を正にすることができる。その




(a) Tanh　(b) ReLU

(c) Sigmoid　(d) SELU

図 2.2 種々の活性化関数

後、全クラスの出力の合計値を 1 にするように正規化を行うことで、各クラスの出力は確率として解釈できるようになる。

## 2.2　深層生成モデル

データ $x$ と、そのデータが所属するクラスを表すラベル $y$ から分類境界を直接モデリングし、クラス分類を行う手法を判別モデルという。判別モデルとしては Support Vector Machine (SVM) や Random Forest Classification など、多くの手法が提案されている。データ $x$ と実測値 $y$ から回帰曲面を直接モデリングし、回帰を行う手法を回帰モデルという。回帰モデルには Support Vector Regression (SVR) や Random Forest Regression などが含まれる。

一方、生成モデルではデータ $x$ が潜在変数 $z$ から生成されたと考える。そして、潜在変数の分布 $p(z)$ とデータの分布 $p(x)$ との変換規則 $p(x|z)$ をモデリングする。潜在変数の分布 $p(z)$ からサンプリングして生成モデルに通すと、$p(x)$、すなわちデータの分布に沿ってサンプルを生成することができる。

深層学習を利用しない生成モデルとしては、Gaussian Mixture Model (GMM) や Hidden Markov Model (HMM)、Latent Dirichlet Allocation (LDA) などが存在する。

生成モデルに深層学習を用いた手法を深層生成モデルという。代表的な深層生成モデル

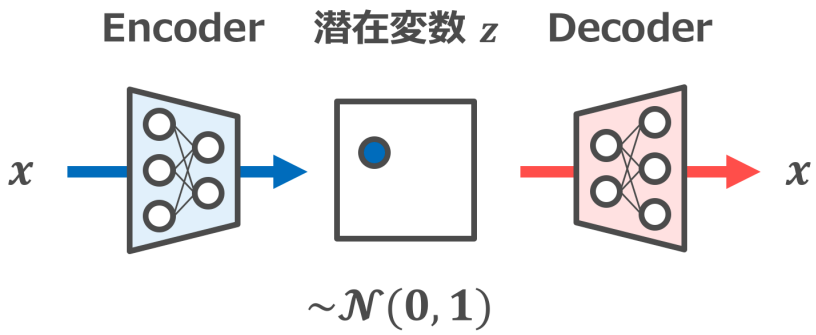



図 2.3 VAE の概略図

としては

- Variational Autoencoders (VAE)
- Generative Adversarial Networks (GAN)
- Adversarial Autoencoders (AAE)

などが挙げられる。

### 2.2.1 Variational Autoencoders (VAE)

図 2.3 に Variational Autoencoders (VAE)[3, 4] の概略図を示す。VAE とは、潜在変数に正規分布 $\mathcal{N}(0,1)$ を仮定した Autoencoders の一種である。

入力 $x$ から潜在変数 $z$ への変換を行う Encoder と $z$ から $x$ への変換を行う Decoder を同時に学習させる。すなわち、$p(z|x)$ を Encoder モデル $\mathrm{Enc}(z|x)$ で、$p(x|z)$ を Decoder モデル $\mathrm{Dec}(x|z)$ で近似させる。

$\mathrm{Enc}(z|x)$ の学習とは $\mathrm{Enc}(z|x)$ と $p(z|x)$ の Kullback-Leibler divergence を最小にすることであるから、

$$\begin{aligned}\mathrm{KL}[\mathrm{Enc}(z|x)||p(z|x)] &= \mathbb{E}[\log \mathrm{Enc}(z|x) - \log p(z|x)] \\ &= \mathbb{E}[\log \mathrm{Enc}(z|x) - \log p(x|z) - \log p(z)] + \log p(x) \\ &= \mathrm{KL}[\mathrm{Enc}(z|x)||p(z)] - \mathbb{E}[\log p(x|z)] + \log p(x)\end{aligned}$$

より、

$$\mathrm{KL}[\mathrm{Enc}(z|x)||p(z|x)] - \log p(x) = \mathrm{KL}[\mathrm{Enc}(z|x)||p(z)] - \mathbb{E}[\log p(x|z)] \tag{2.2}$$

が成り立つ。左辺第 1 項を最小化することは Encoder を学習することを表し、左辺第 2 項を最小化することは Autoencoders 全体を学習することを意味する。

よって、式 2.2 の右辺が VAE における損失関数 (Loss) となり、これを最小化することが学習の目的である。

#### ネットワークの構成法

Encoder は適当なニューラルネットワークで作成するが、その出力 $z$ は正規分布 $\mathcal{N}$ に従うと仮定する。この仮定に基づくと、Encoder は平均 $\mu(x)$ と分散 $\Sigma(x)$ を出力するネットワークとして設計することができる。

Decoder は $\mu(x)$ と $\Sigma(x)$ から 1 点をサンプリングし、そこから $x$ を再構成するネットワークとして設計すれば良い。

#### Reparametrization Trick

式 2.2 において、$\mathbb{E}[\log p(x|z)]$ を計算するためにはサンプリングが用いられる。しかし、ニューラルネットワーク中でサンプリングを行ってしまうと計算グラフが途切れてしまうため、誤差逆伝播法による学習を行うことができない。そこで用いられるのが Reparametrization Trick と呼ばれる手法である。

$\mathcal{N}(\mu, \Sigma)$ からサンプリングする代わりに $e \sim \mathcal{N}(0, 1)$ として $z = \mu(x) + e\Sigma(x)$ を計算する。このように計算を行うことで、サンプリングと等価な演算を行いながら誤差逆伝播法による学習も行うことが可能になる。

#### Conditional VAE (CVAE) への拡張

VAE の拡張として、潜在変数の生成に目的変数の情報を考慮できる Conditional VAE (CVAE)[3] が提案されている。Kingma らは CVAE を用いた M1+M2 モデルという深層生成モデルを提案した。M1+M2 モデルは目的変数の情報を考慮しない M1 モデルを予め大量の教師なしのデータで学習させておき、得られた潜在変数を用いて目的変数の情報を考慮する M2 モデルを僅かな教師付きデータと大量の教師なしデータを混ぜて学習するモデルである。このモデルは手書き文字認識タスク MNIST において、70,000 枚中わずか 100 枚にのみ正解を与える半教師付き学習で正解率 96% を達成した。このように、CVAE は半教師付き学習のフレームワークとして利用可能である。

### 2.2.2 Generative Adversarial Networks (GAN)

図 2.4 に Generative Adversarial Networks (GAN) [15] の概念図を示す。GAN とは、Generator と Discriminator という 2 つのモデルを敵対的に学習させることによってデータを生成させる枠組みである。Generator $G(z)$ は乱数 $z$ から実データと似たデータ $G(z)$ を生成する。Discriminator $D(x)$ は実データと Generator が作り出したデータを判別する。

GAN で用いられる目的関数は以下の式で表される。

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.3}$$

Discriminator を学習する際は Generator を固定した上で式 2.3 を最大化する。右辺第 1 項は実際のデータに対応した項で、この項を最大化するためには実データに対して Dicriminator

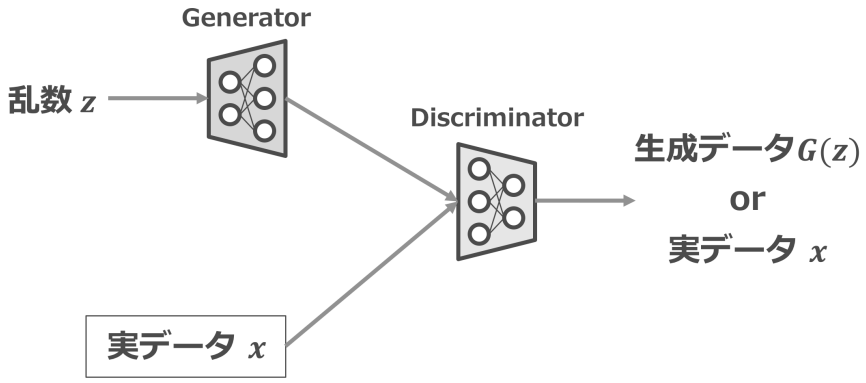



図 2.4 GAN の概念図

が 1 と判断すれば良い。右辺第 2 項は Generator が生成したデータに対応した項で、この項を最大化するためには Generator が生成したデータに対して 0 と出力すればよい。

Generator を学習する際は Discriminator を固定した上で式 2.3 を最小化すればよい。第 1 項は Generator に依存しないので第 2 項のみを考える。第 2 項を最小化するためには $D(G(z))$ を 1 に近づける、すなわち、Generator が生成した画像に 1 というラベルをつけて学習させればよい。

### 2.2.3 Adversarial Autoencoders (AAE)

Adversarial Autoencoders (AAE) [6] とは、オートエンコーダに対して GAN の枠組みで正則化を行う手法である。

VAE では Kullback-Leibler divergence を解析的に計算するために潜在変数を単純な正規分布と仮定する必要があったが、AAE では GAN の枠組みで正則化を行うことで潜在変数を任意の分布に近づけることができる。

#### AAE の概要

図 2.5 に AAE の概略図を示す。AAE は基本的にはオートエンコーダであるため、入力 $x$ を潜在変数 $z$ にエンコードしたり、$z$ を $x$ にデコードしたりする。オートエンコーダにおける Encoder を Generator $G$ として扱う。Discriminator $D$ はオートエンコーダとは別のネットワークとして定義し、与えられた $z$ が Generator を通した後の $q(z)$ 由来のものなのか、任意の分布 $p_z(z)$ からサンプリングしたものなのかを判別する。

目的関数は式 2.3 と類似しており、

$$\min_G \max_D V(D,G) = \mathbb{E}_{z\sim p_z(z)}[\log D(z)] + \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log(1 - D(G(x)))] \tag{2.4}$$

で表される。任意の分布 $p_z$ からサンプリングしたものを真のデータと見なしている点と、GAN とは $x$ と $z$ が逆になっている点に注意が必要である。

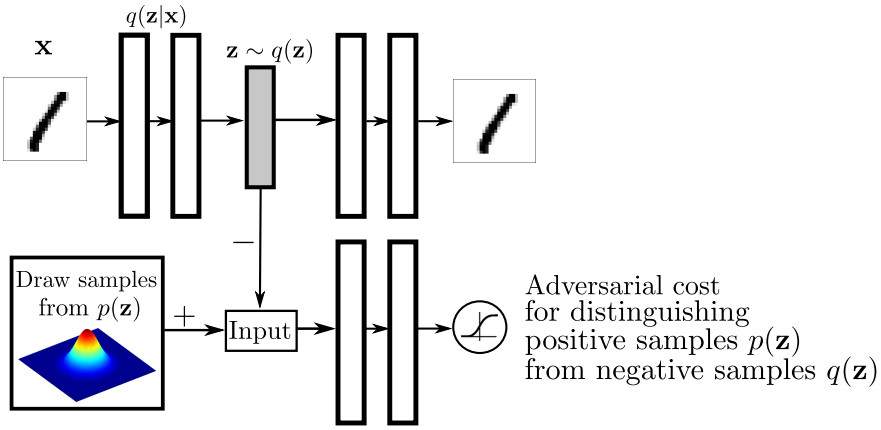



図 2.5 Adversarial Autoencoders の概略図 [6]

学習の際は、

1. オートエンコーダ部分を復元誤差を最小化するように学習させる
2. Discriminator を式 2.4 を最大化するように学習させる
3. Generator を式 2.4 を最小化するように学習させる

の順で行う。

式 2.4 を最小化するためには $\log(1 - D(G(x)))$ を最小化すればよく、$D(G(x))$ を 1 に近づければ良い。これは、Generator の生成結果を Discriminator が真のデータと予測することを意味する。したがって、Generator を学習させる過程で、潜在変数 $q(z)$ を $p(z)$ に近づけることができる。

#### AAE の半教師付き学習への応用

AAE を半教師付き学習に応用する場合のアーキテクチャを図 2.6 に示す。Disciriminator にクラスを表すラベルを入力することで、Discriminator がラベルの情報を使って分類することが可能になる。ラベルを参照する Discriminator を騙すため、Generator は同じラベルを持ったサンプルを近づけるように学習するようになる。

Discriminator に与えるラベルに「ラベルなし」というクラスを用意することで、AAE を半教師付き学習として利用することができる。本研究では、このアーキテクチャを利用した。

### 2.2.4 Positive-Unlabeled GAN (PUGAN)

Positive-Unlabeled GAN (PUGAN) [8] とは、GAN に positive-unlabeled learning の枠組みを組み合わせた手法である。positive-unlabeled learning 自体は古くから存在する考え方で、僅かな正例と大量のラベルなしデータを利用して機械学習を行うという発想である。

図 2.7 に PUGAN の概念図を示す。PUGAN は 3 つの Discriminator と 2 つの Generator

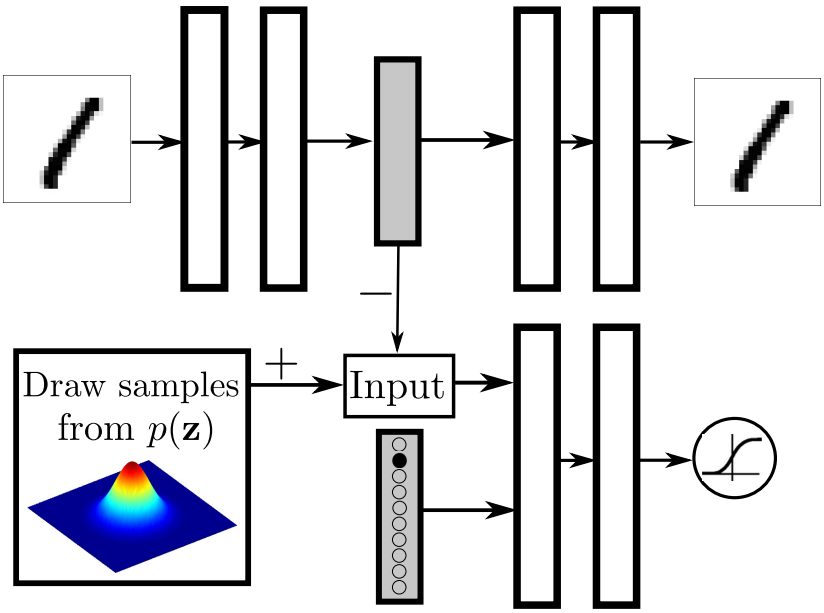



図 2.6 AAE を半教師付き学習へ応用する場合の構造 [6]

を持つ。それぞれの役割は以下の通りである。

- Positive Generator: 正例らしいデータを生成する
- Negative Generator: 負例らしいデータを生成する
- Positive Discriminator: 正例と Positive Generator が生成した正例を判別する
- Unlabeled Discriminator: ラベルなしデータと Generator が生成したデータを判別する
- Negative Discriminator: 正例と Negative Generator が生成した負例を判別する

特徴的なのは正例である $x_p$ を 2 回入力として用いる点である。

生成を行う際は Positive Generator に乱数を入力することで正例らしいデータを生成することができる。

## 2.3 深層学習で文字列を扱う手法

### 2.3.1 文字列の埋め込み (Embedding)

文字列はそのままニューラルネットワークの入力にすることはできない。そこで、文字列の各要素を多次元の数値ベクトルに変換する必要がある。これを埋め込み (Embedding) という。一般的な文章の場合は各単語を、SMILES の場合は各文字を、それぞれ埋め込むことになる。

方法のひとつは図 2.8 に示す One-Hot Encoding である。SMILES に含まれる文字種を抽出した後、その各文字に対して該当する文字種のビットだけが 1 になっているダミー変数

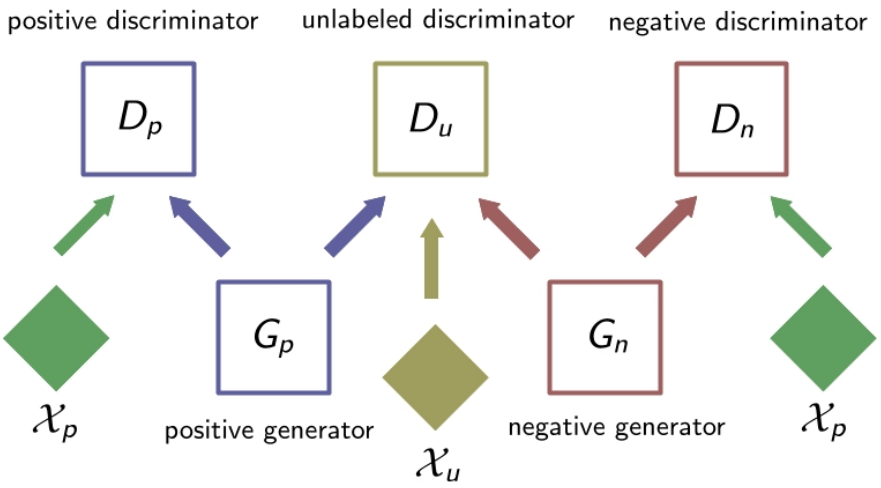



図 2.7 PUGAN の概略図 [8]


one-hot encoding

| | C | 1 | = | C | ( | C | = | N | N | 1 | ) | C | Br |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| = | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| ( | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| N | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| ) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Br | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

図 2.8 One-Hot Encoding の概念図

に変換する。データセットに含まれる文字種が $M$ 種類、SMILES の長さが $N$ 文字であった場合、ひとつの構造は $M \times N$ の行列で表す。実際には分子によって SMILES の長さが異なるため、データセット中で最も長い SMILES を持つ分子の長さを $N$ に設定し、足りない部分はすべて空白埋めを行って長さを揃える。One-Hot Encode されたベクトルは文字列と 1 対 1 対応するため、モデルの出力として One-Hot Encode されたベクトルを得られれば直に SMILES 文字列を得たことに相当する。

もうひとつの方法は多くのニューラルネットワークライブラリに実装されている Embeeding Layer を用いる方法である。One-Hot Encoding が多くの 0 を含むスパースな埋め込みになるのに対し、Embedding Layer を用いた場合は非零要素が少ない密ベクトルでの

埋め込みとなる。埋め込み方は Embedding Layer をランダムに初期化した後、誤差逆伝播法によってモデルと同時に学習させる。Embedding Layer を用いた場合も、モデルの出力としては One-Hot Encode されたベクトルを得ることを目指す。

本研究ではいずれの埋め込み方法も試し、性能の優劣を確認した。

### 2.3.2 Recurrent Neural Network (RNN)

系列データを扱うための深層学習モデルを総称して Recurrent Neural Network (RNN) と言う。RNN は内部状態を持っており、系列を順番に読み込み、内部状態をアップデートしながら予測を行う。

層の出力と入力を単純に接続し、前回の出力を考慮した計算を可能としたものを単純 RNN[16] と呼ぶ。単純 RNN は考慮するステップが多くなった場合に勾配消失や勾配爆発が起こるという問題があった。勾配消失や勾配爆発が起こると誤差逆伝播法による学習を行うことができない。そこで、Long Short-Term Memory (LSTM) [17–19] と Gated Recurrent Unit (GRU) [20] が提案されている。

LSTM は何度かのアップデートを経て性能が向上した手法である。Hochreiter らは LSTM に状態を記憶するメモリセルを用意し、メモリセルを書き換えるべきかを判断する入力ゲートとメモリセルの内容を出力すべきか判断する出力ゲートを提案した [17]。Gers らは系列の傾向が大きく変わった際に以前の情報を適切に忘れるために忘却ゲートを提案した [18]。Gers らは各ゲートの判断が系列のみに依存していることに着目し、メモリセル自体がゲートの判断に関与することを可能にする Peephole Connection を提案した [19]。他にもいくつかの改善が存在するが、多くのライブラリに実装されている LSTM はここまでの改善を反映したものである。

Cho らは LSTM の構造が複雑であることに注目し、ゲートを更新ゲートとリセットゲート 2 つに絞った GRU を提案した [20]。GRU はパラメータが少ないこともあり、特にデータ数が少ない状況で効果を発揮する。

本研究では LSTM と GRU について比較を行った。

### 2.3.3 Teacher Forcing

文字列を生成させるモデルの場合、先頭から順次生成させるのが一般的である。図 2.9 に文字列生成の概略を示す。最初は開始トークンを入力して各文字の確率を出力として得た後、確率最大の文字を 1 文字目とする。2 文字目以降は以前の出力を入力とする。以前の文字から次の文字を予測することになるため、文字列の後半になると誤差が蓄積され、学習が不安定になったり意味のある文字列が生成されなくなるという問題がある。

この問題に対処するため提案されたのが Teacher Forcing[21] である。図 2.10 に Teacher Forcing の概略を示す。Teacher Forcing を適用する場合、訓練時は予測された文字ではなく、正解の文字を次への入力とする。これにより学習が安定し、収束が早くなるメリット

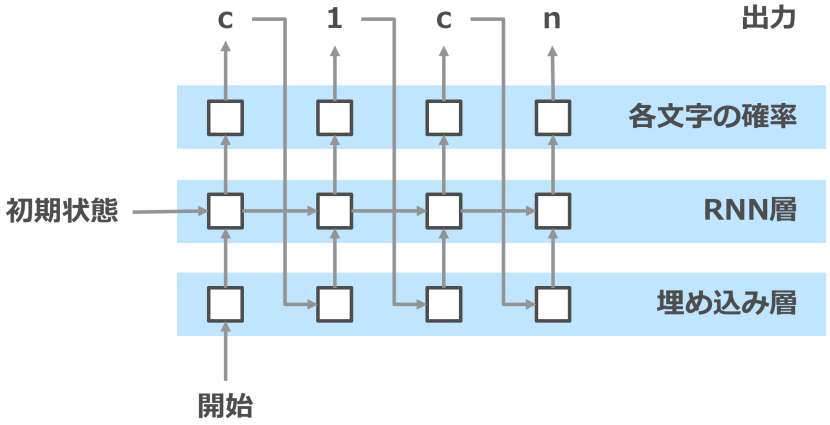



図 2.9 文字列生成の概略図

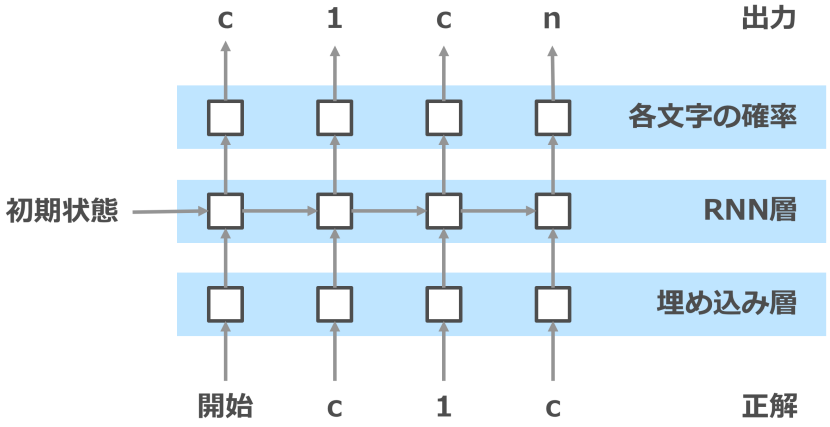



図 2.10 Teacher Forcing の概略図

がある。

Teacher Forcing のデメリットとしては、推論時には学習時と異なる状況で推論を行わなければいけないという点が挙げられる。この問題を改善するために、Scheduled Sampling[22] が提案されている。図 2.11 に Scheduled Sampling の概略を示す。Scheduled Sampling を利用する場合、正解の文字を確率 $p$ で、予測された文字を確率 $1 - p$ でサンプリングして次の入力とする。

Scheduled Sampling の確率を $p = 0$ とした場合は通常の学習、$p = 1$ とした場合は Teacher Forcing となる。一般には Scheduled Sampling も含めて Teacher Forcing と呼ぶため、本研究でもそれに倣った。

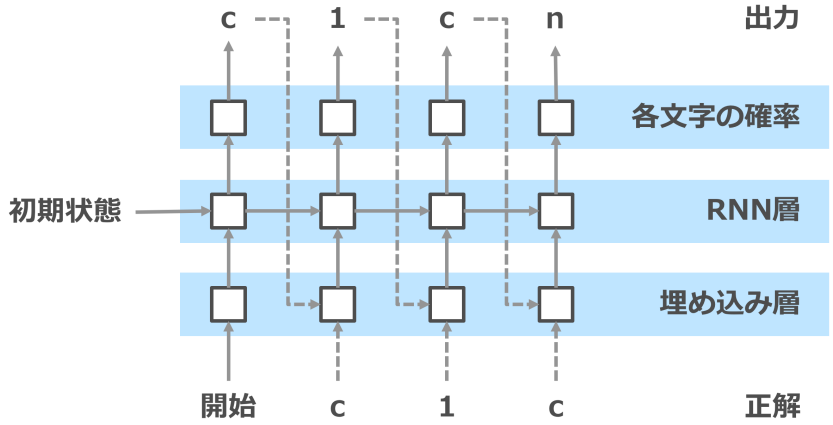



図 2.11 Scheduled Sampling の概略図

### 2.3.4 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN, 畳み込みニューラルネットワーク) とは、一般に画像解析の際に用いられる深層学習手法である。CNN は複数のフィルタを局所的にかけていくことで画像から特徴を抽出する。

近年、CNN は文字列解析にも用いられるようになっている。画像解析で用いられる CNN はフィルタが 2 次元であるが、文字列解析では 1 次元のフィルタを用いる。1 次元のフィルタを用いた CNN を Conv1D と表記することが多い。

CNN に特有のパラメータとして、カーネルサイズとフィルタ数がある。カーネルサイズはどの程度近傍の情報を利用するかというパラメータで、大きくするほど全体的な特徴を、小さくするほど局所的な特徴を取得するようになる。フィルタ数は近傍のとり方の種類数を表す。

## 2.4 深層生成モデルを化学データに適用した先行研究

本節では、深層生成モデルを化学データに適用した先行研究について概説する。

### 2.4.1 化合物から連続的な潜在表現を取り出す

Gómez-Bombarelli らは Variational Autoencoder (VAE) を用いて、化学構造の線形表記の 1 種である SMILES から連続値の潜在表現を得る手法を提案した [5]。これを chemical VAE と呼ぶ。

図 2.12 に chemical VAE の概念図を示す。chemical VAE は化学構造の SMILES 表記を入力とし、SMILES 表記から連続潜在変数を取り出すエンコーダと潜在変数から SMILES

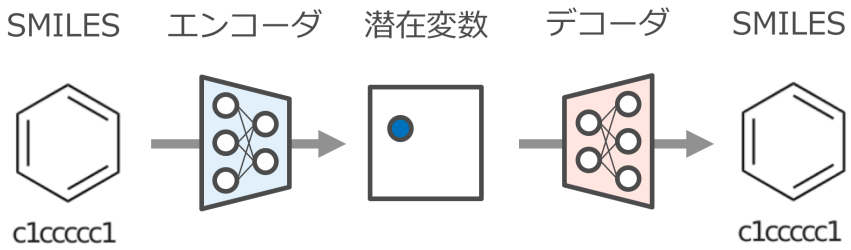



図 2.12 chemical VAE の概念図

表記を復元するデコーダを用意し学習を行う。エンコーダには文字列を畳み込む 1 次元 CNN を、デコーダには文字列を生成する RNN を用いることが多い。潜在変数空間中の 1 点を指定してデコーダに通すことで、対応する SMILES を直に得ることができる。

所望の活性値を持つ化学構造を得る場合、2 通りのアプローチが存在する。1 つは学習した潜在変数を特徴量とし、活性値との間で新たな統計モデルを構築する手法である。もう 1 つは学習した chemical VAE は構造の組み立てにのみ用い、活性値の予測は通常の QSPR と同様に記述子を計算して行うという手法である。いずれの場合でも、活性値との関係を学習するためには多量のデータが必要になるという問題点が存在する。

### 2.4.2 種々の深層生成モデルを比較する

Blaschke らは複数の深層生成モデルを比較し、AAE とベイズ最適化を組み合わせることで医薬品である確率が高く、新規な構造を生成できることを示している [7]。

用いているデータセットは EsCAPE-DB から抽出した Dopamine Receptor D2 (DRD2) データセットである。検証の流れは ChEMBL22 で VAE、AAE の各モデルを学習した後、得られた潜在変数空間と活性値との間で Support Vector Machine (SVM) を用いた活性予測モデルを構築する。その後活性予測モデルの出力を最大化するように潜在変数空間中でベイズ最適化を行う。

しかし、DRD2 は正例 7,218 件、負例 343,204 が存在する比較的大きなデータセットである。データセットが小さくなった場合、ベイズ最適化の目的関数となる活性予測モデルの外挿能力が低下するという問題が発生する。目的関数が信頼できない状況で行ったベイズ最適化の結果は同様に信頼することが難しい。AAE の潜在変数には活性の情報が反映されていないため、AAE の潜在変数から直接サンプリングを行って新規構造を生成することも困難である。

## 2.5 提案手法

本研究では、以下の 3 つの手法を半教師付き学習的に化学データに適用し、検証を行った。

- Variational Autoencoders (VAE) [3, 4]

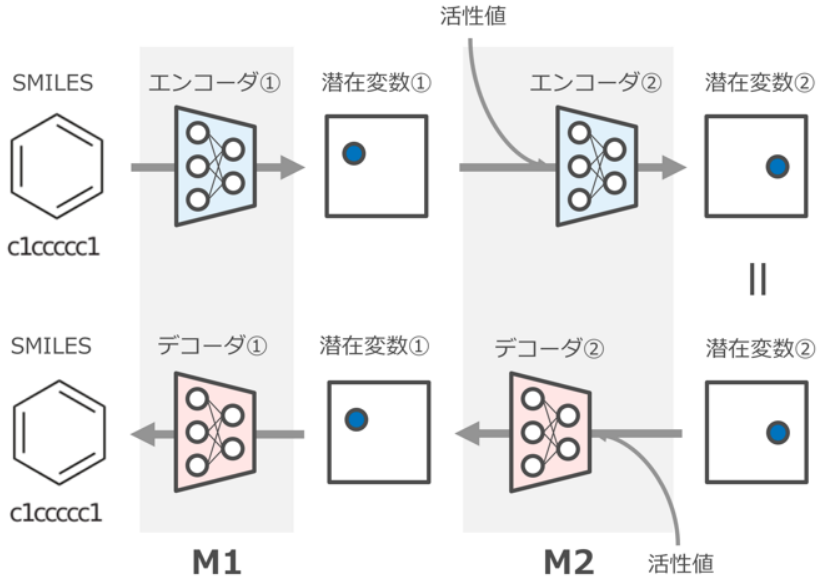



図 2.13 VAE を利用した半教師付き学習

- Adversarial Autoencoders (AAE) [6]
- Positive-Unlabeled GAN (PUGAN) [8]

本節では各手法を適用した場合について詳説する。

### 2.5.1 VAE を用いた高精度な活性予測

図 2.13 に提案手法の概念図を示す。M1 モデルでは活性値の情報のついていない大量の化学構造データから、SMILES と化学構造一般の特徴（潜在変数①）の対応付けを学習する。M2 モデルではわずかな活性値付きデータを混ぜて学習し、化学構造一般の特徴（潜在変数①）と活性値との関係を潜在変数②に落とし込む。

デコーダ②に活性値を通した場合、潜在変数②の情報と活性値の両方から最も尤もらしい SMILES を組み立てる構造生成器として利用する。デコーダ②に活性値を通さなかった場合、潜在変数②から活性値を予測する予測モデルとしても使用可能になるような設計を検討した。活性値のついていないデータは ChEMBL[23] や ZINC[24, 25] などのデータベースから容易に取得可能であり、今回は ChEMBL を利用した。実験化学者が各々で取得している少量の活性データなどと組み合わせて良好な予測・構造生成を行うことを目指した。

### 2.5.2 AAE を用いた構造提案

大量のデータを利用した場合、AAE で良好な構造提案ができることは Blaschke らの研究 [7] によって示されている。そこで、本研究の着眼点は少数データにおいて AAE が適用

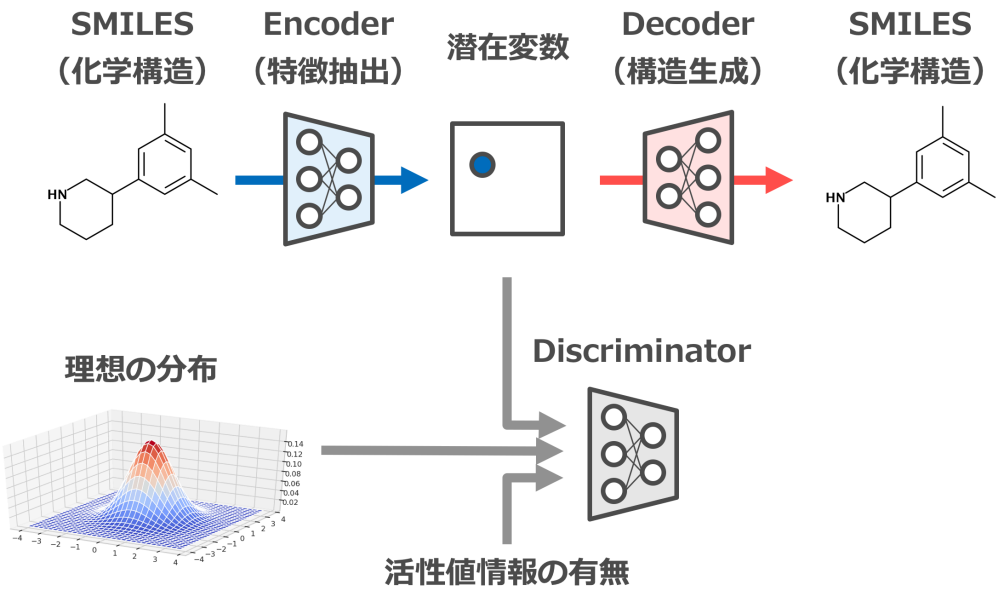



図 2.14 AAE を適用する場合の概略図

できるか確認することにある。

図 2.14 に AAE を適用する場合の概略図を示す。本研究では入力 $x$ に化学構造の線形表記である SMILES を利用し、SMILES を 2 次元のガウス分布に変換することを目指した。半教師付き学習を行うため、Discriminator に与えられた構造が活性値情報を持っているか否かの情報を与えた。Encoder は Decoder で正しく構造が復元できること、および Discriminator を誤分類させることを目指して学習を行った。このように学習を行うことで潜在変数 $z$ が上で挙げた望ましい性質を満たすようになった。

### 2.5.3 PUGAN を用いた構造提案

図 2.15 に PUGAN を適用する場合の概略図を示す。本研究では活性ありのデータセットとして ADRA2 データセットを、活性値情報なしのデータセットとして ChEMBL22 データセットをそれぞれ利用した。Positive Generator と Negative Generator の入力には一様乱数を利用した。

構造生成を行う場合は、Positive Generator に一様乱数を与えることでサンプリングを行うことができる。

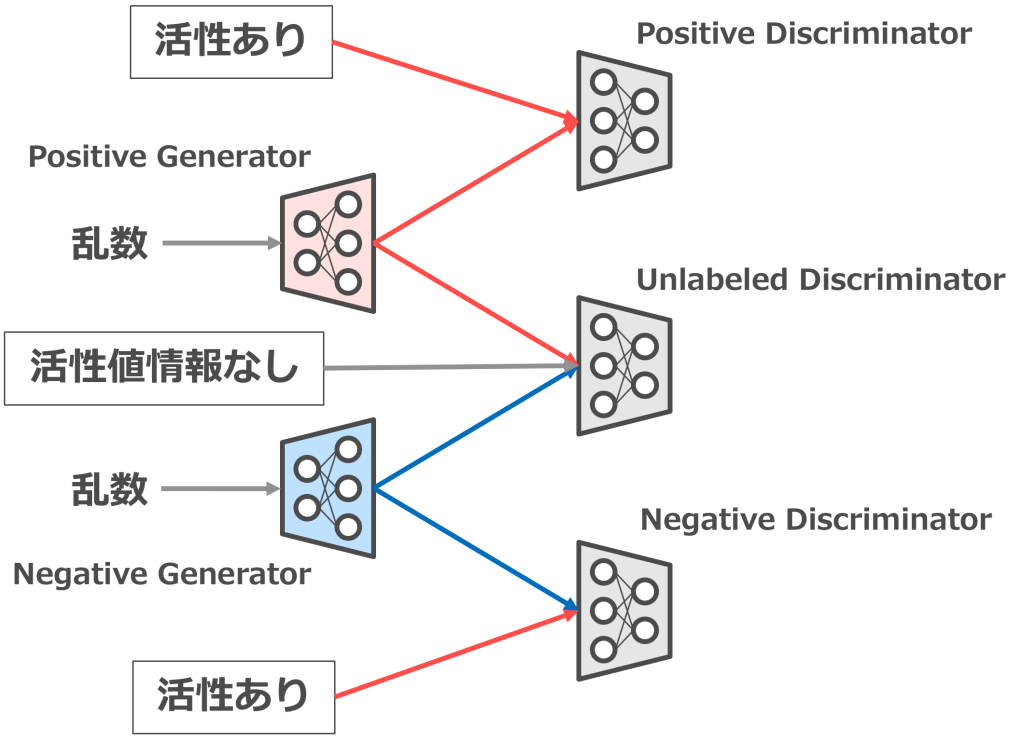



図 2.15 PUGAN を適用する場合の概略図

# 第 3 章

# ケーススタディ

## 3.1 データセット

データセットとして ChEMBL22 と ADRA2A を用いた (表 3.1)。

ChEMBL22 データセットは ChEMBL データベース [23] より取得した 1,300,000 件余りのデータからランダムに 500,000 件をサンプリングしてきた、活性値情報を持たないデータである。M1 モデルに対して教師なし学習を行うのに用い、SMILES から化学構造一般の特徴を得た。ADRA2A データセットは ChEMBL データベースより抽出されたヒト alpha-2A adrenergic receptor に対する阻害活性についてのデータであり、501 件存在する。M2 モデルに対して ChEMBL22 と合わせて半教師付き学習を行って活性値の特徴を得るのに用いた。

### 3.1.1 データセットの分割

ADRA2A のデータセットを訓練 401 構造、検証 100 構造に分割した。検証構造はモデルの訓練には用いず、最終的に生成された構造が検証構造と類似の骨格を持っているかを確認するのに用いた。

### 3.1.2 行った前処理

前処理として、キラリティ情報の削除、SMILES のカノニカライズ、原子数の制約、SMILES 文字列長の制限、重複の削除を行った。以下に詳細を示す。

表 3.1 用いたデータセット

| 名称 | データ数 | 活性値情報 | 活性 |
|---|---|---|---|
| ChEMBL 22 | 500,000 | × | — |
| ADRA2A | 501 | ○ | あり |

#### キラリティ情報の削除

本来、薬理活性においてキラリティは重要な役割を持つ。しかし、ChEMBL や ADRA2A のデータセットには本来キラリティを考慮しなければならないのにキラリティ情報が含まれていない分子が存在する。

このような状況においてキラリティ情報を利用することはモデルの学習に悪影響を及ぼす。そこで、本研究では全分子からキラリティの情報を削除することで対応した。

#### SMILES のカノニカライズ

SMILES 表記は開始原子の位置や環を辿る順番などによって複数の表記が存在しうる。そこで、本研究では SMILES を一意に定めるカノニカライズを施した。カノニカライズにはフリーのケモインフォマティクスライブラリである RDKit[26] を用いた。

#### 原子数の制約

SMILES の文字列を生成させる上で、小さすぎる分子はノイズとなる。そこで、水素以外の原子が 10 個未満の分子はデータセットから除外した。これは先行研究 [7] で用いられていた基準と同様である。

#### SMILES 文字列長の制限

本研究では、SMILES を生成させる際に RNN を利用した。RNN は多少の系列長の変化に対しては頑健なものの、極端に長い系列や極端に短い系列に対しては対応が難しい。図 3.1 に各データセットにおける SMILES の文字列長のヒストグラムを示す。横軸が SMILES での文字列長、縦軸が度数である。横軸で表示されている範囲が該当する文字列長の SMILES が存在する範囲である。ADRA2A が 120 文字までにすべての分子が収まっているのに対し、ChEMBL22 には極端に文字列長が長い分子が含まれていることが分かる。そこで、本研究では ADRA2A に合わせ、ChEMBL22 のうち 120 文字以下の分子についてのみ利用することとした。

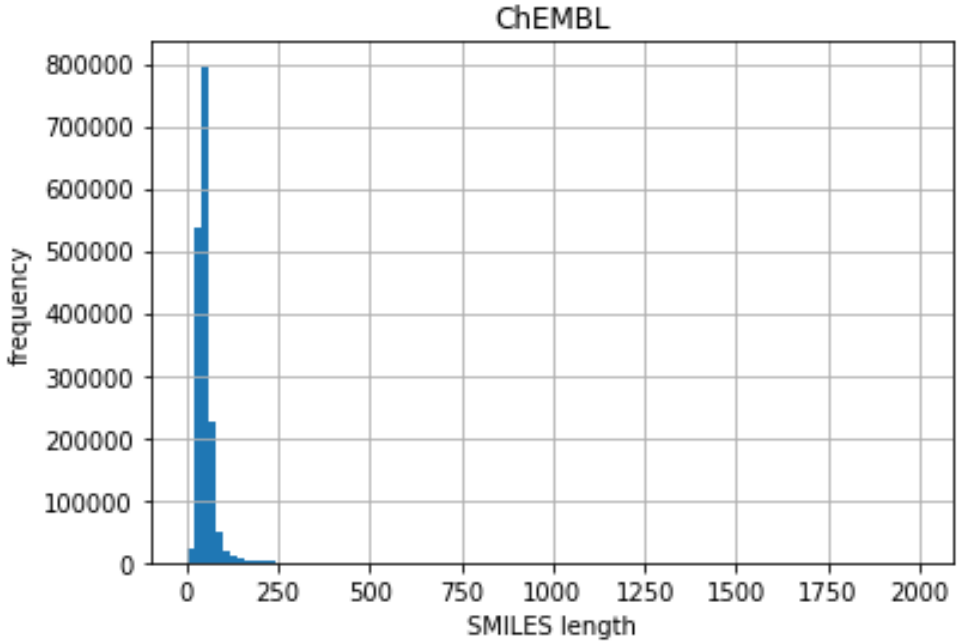
#### 重複の削除

ChEMBL22 中に存在する ADRA2A の構造をすべて削除した。

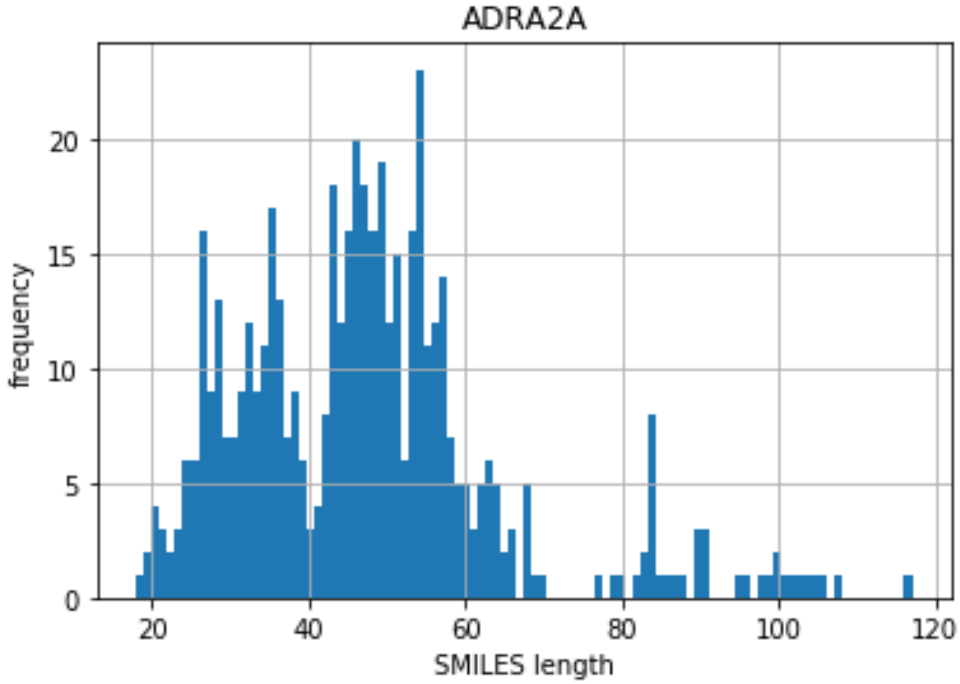
## 3.2 VAE を用いた高精度な活性予測

本節では VAE を適用した際の結果を示す。

VAE を利用して活性値の回帰を行ったあと、構造生成を行った。




(a) ChEMBL22




(b) ADRA2A

図 3.1 各データセットにおける SMILES 文字列長

表 3.2 VAE で検討したパラメータ

| | 項目 | パラメータ候補 |
|---|---|---|
| 層の種類 | Encoder の入力層 | Embedding, One-Hot Encoding |
| | Encoder の隠れ層 | LSTM, GRU, Conv1D |
| | Encoder の出力層 | Reparametrization |
| | Decoder の入力層 | Dense |
| | Decoder の隠れ層 | LSTM, GRU |
| | Decoder の出力層 | Softmax |
| 各層のパラメータ | Embedding の次元数 | 8, 16, 32, 64 |
| | Dense の次元数 | 32, 64, 128, 256 |
| | LSTM の次元数 | 16, 32, 64, 128 |
| | GRU の次元数 | 16, 32, 64, 128 |
| | Conv1D のチャネル数 | 8, 16, 32, 64 |
| | Conv1D のフィルタサイズ | 5, 6, 7, 8, 9, 10 |
| | 活性化関数 | ReLU, SELU |
| 最適化 | バッチサイズ | 512 |
| | 最適化手法 | Adam |
| | 学習率 | 1e-3 |

### 3.2.1 使用したアーキテクチャ

表 3.2 に VAE のアーキテクチャで検討したパラメータを示す。Dense は通常の全結合層、Conv1D は 1 次元の畳み込み層、GRU・LSTM は RNN の一種、Reparametrization 層は入力を平均 $\mu$ と対数分散 $\log \sigma^2$ に変換する層、Softmax は各文字の出力を確率に変換する層をそれぞれ表す。

表 3.3 に採用されたパラメータを示す。パラメータ候補のうち、層の種類はそれぞれの中で全探索するグリッドサーチを行った。層のパラメータについては、次元数とフィルタサイズ、活性化関数についてはそれぞれ全探索を、フィルタサイズについては組み合わせが膨大になるため実験の途中からは先行研究 [5] と同様のパラメータを利用した。Conv1D 層の活性化関数には学習が速く、かつ精度も向上した SELU[14] を採用した。一方、Dense、GRU の各層には一般的な区分線形関数 ReLU を採用した。

### 3.2.2 比較した手法

表 3.4 に比較した手法の概要を示す。一般的な機械学習手法として、Morgan Fingerprint で特徴抽出をして Random Forest[27] でモデルを構築したもの（FP+RF）、RDKit Descriptor

表 3.3 モデルのアーキテクチャ

| | 層の種類 | 層数 | パラメータ | 活性化関数 |
|---|---|---|---|---|
| Encoder | Embedding | 1 | 次元数 16 | — |
| | Conv1D | 3 | チャネル数 32, フィルタサイズ [9, 9, 10] | SELU |
| | Dense | 2 | 次元数 32 | ReLU |
| | Reparametrization | — | 1 | — |
| Decoder | Dense | 1 | 次元数 32 | ReLU |
| | GRU | 3 | 次元数 32 | ReLU |
| | Dense | 1 | 次元数 32 | ReLU |
| | Dense | 1 | — | Softmax |

表 3.4 比較した手法

| 名称 | 特徴抽出 | モデル |
|---|---|---|
| FP+RF | Morgan Fingerprint | Random Forest |
| desc+RF | RDKit Descriptor | Random Forest |
| 提案手法 (VAE) | VAE | VAE |

で特徴抽出をして Random Forest でモデルを構築したもの（desc+RF）を使用した。

Random Forest[27] は広く用いられている非線形回帰手法で、複数の回帰木のアンサンブルを行うことで良い予測精度を示すことが知られている。

Morgan Fingerprint と RDKit Descriptor はいずれもオープンソースのケモインフォマティクスライブラリである RDKit[26] で計算した。

5-fold クロスバリデーションの予測精度が最も高くなるハイパーパラメータを探索した結果、Morgan Fingerprint は半径 2、ビット数 512 を採用し、Random Forest の回帰木の数は 900 を採用した。

### 3.2.3 結果と考察

**予測精度の比較**

表 3.5 に予測精度の比較結果を示す。$R^2$ は決定係数と呼ばれ、説明変数が目的変数をどれだけ説明しているかを表す指標である。範囲は $[-\infty, 1]$ を取り、1 に近いほど予測精度が良いことを表す。MAE は絶対予測誤差の平均であり、範囲は $[0, \infty]$ を取る。0 に近いほど予測精度が良いことを示す。添字の val はクロスバリデーションの精度、test はテストデータに対する精度を表す。テストデータに対する予測精度を確認すると、既往手法に比べて提案手法ではいずれの評価指標においても良好な予測ができたことが分かった。

以上の結果は、教師付きデータが少ない状況下において提案手法が構造と活性値との関

表 3.5 予測精度の比較

| 名称 | $R^2_{\mathrm{val}}$ | $\mathrm{MAE}_{\mathrm{val}}$ | $R^2_{\mathrm{test}}$ | $\mathrm{MAE}_{\mathrm{test}}$ |
|---|---|---|---|---|
| FP+RF | 0.832 | 0.213 | 0.853 | 0.196 |
| desc+RF | 0.727 | 0.276 | 0.790 | 0.252 |
| 提案手法 (VAE) | 0.882 | 0.209 | 0.903 | 0.190 |

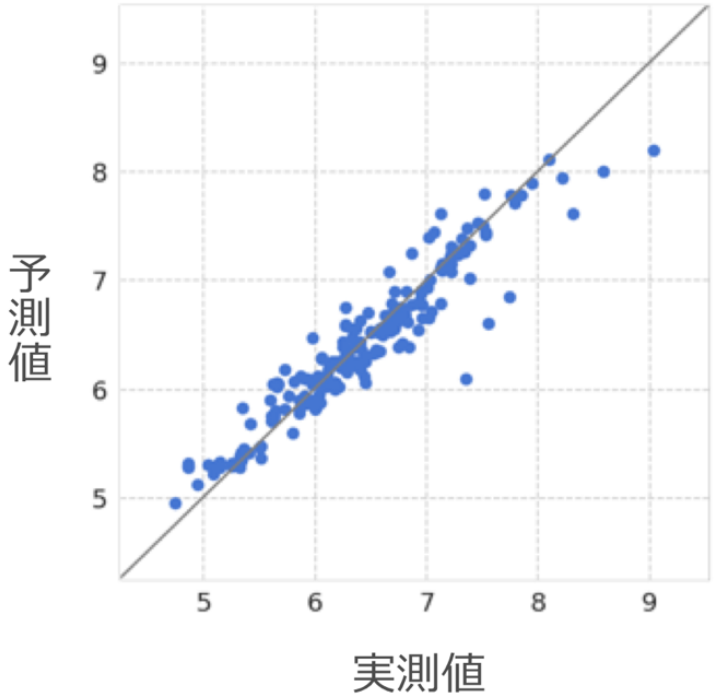


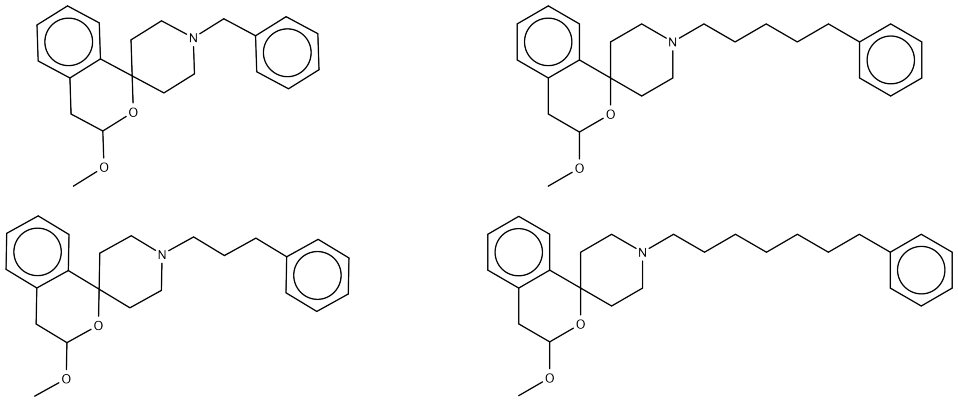

図 3.2 実測値と予測値の関係

係性を良くモデリングできたことを示すものである。図 3.2 にテストデータに対する実測 pKi と予測 pKi の関係を示す。広い範囲において良好な予測ができていることが分かった。これは、活性値情報のない ChEMBL22 のデータによって SMILES の文法が学習され、わずかな教師データでも活性値との関係を学習できたことによると考えられる。

**構造生成**

良好な予測ができたため、潜在変数のサンプリングによって構造生成を試みた。

図 3.3 に構造生成の結果の一例を示した。これらは構造生成結果を目視で確認し、特徴的と感じた生成構造である。訓練データに用いた構造と比較して、炭素鎖を伸ばしただけのものや、末端原子を置換しただけのものが多く生成された。これらの構造は構造式上は新規構造ではあるが、骨格に全く変化がないため、創薬を意識した際に有用とは言えない。



(a) 訓練データ　　(b) 生成構造

図 3.3 炭素鎖を伸ばしただけの生成構造

考えられる原因は 2 点存在する。

1 点目はモデルが M1、M2 の 2 段階からなるということである。潜在変数から復元を行う過程では復元誤差が存在する。復元を 2 段階にすると 2 段階分の復元誤差が乗るため、正しい SMILES が生成されにくくなる。復元誤差が小さくなりやすい構造変換は単純な構造変換であるため、炭素鎖の伸長や末端原子の置換といった単純な変換を施しただけの構造が多く生き残ったと考えられる。

2 点目は活性を連続値で表現したことである。活性の有無という 2 値分類での評価であれば、予測にはある程度幅があっても問題ない。しかし、連続値として評価した場合は許容される誤差の範囲が狭くなってしまう。活性値の予測を行う際は、誤差が小さくても構わない。しかし、構造生成の場合はある程度誤差を許容しなければ十分に構造変化を行うことができないと考えられる。

これを受け、以降の実験では、活性の評価は有無の 2 値分類で行った。

## 3.3 AAE を用いた構造提案

本節では AAE を適用した際の結果を示す。

AAE を適用して構造生成を行い、生成された構造が活性を持つ確率の分布や、生成された構造の骨格について検討した。

### 3.3.1 使用したアーキテクチャ

表 3.6 に AAE のアーキテクチャで検討したパラメータを示す。パラメータのうち、層の種類・層の数・隠れ層の数は全ての候補の組み合わせを試すグリッドサーチを行った。Conv1D のカーネルサイズについては先行研究と同様の値を利用した。

表 3.7 に採用されたパラメータを示した。最適化手法には Encoder、Decoder には Adam

表 3.6 AAE で検討したパラメータ

| | 項目 | パラメータ候補 |
|---|---|---|
| 層の種類 | Encoder の入力層 | Embedding, One-Hot Encoding |
| | Encoder の隠れ層 | Conv1D |
| | Encoder の出力層 | Tanh |
| | Decoder の入力層 | Dense |
| | Decoder の隠れ層 | LSTM, GRU |
| | Decoder の出力層 | Softmax |
| | Discriminator の入力層 | Dense |
| | Discriminator の隠れ層 | Dense |
| | Discriminator の出力層 | Softmax |
| 各層のパラメータ | Embedding の次元数 | 16 |
| | Dense の次元数 | 32, 64, 128 |
| | LSTM の次元数 | 16, 32, 64 |
| | GRU の次元数 | 16, 32, 64 |
| | Conv1D のチャネル数 | 16, 32, 64 |
| | Conv1D のフィルタサイズ | 9, 10 |
| | 活性化関数 | ReLU, SELU |
| 最適化 | バッチサイズ | 256, 512 |
| | 最適化手法 | SGD, Adam |
| | 学習率 | 1e-3, 5e-4, 2.5e-4, 1e-4, 5e-5 |

を学習率 $5 \times 10^{-4}$ で用いた。Discriminator には通常の SGD を用い、学習率は $2.5 \times 10^{-4}$ を採用した。Encoder、Decoder と Discriminator の間で最適化手法を変えている理由には、Encoder と Decoder の学習が難しく、Discriminator の学習が容易だという背景がある。

GAN の枠組みはひとつの目的関数を Encoder・Decoder・Discriminator の 3 つが使用する。Discriminator の学習が著しく進んでしまうと、Encoder と Decoder の学習が進まなくなる。そこで、Encoder と Decoder には収束が早い Adam を、Discriminator には単純な SGD を利用した。

### 3.3.2 生成構造の評価方法

生成された構造の評価には、生成された構造が活性を持つ確率および生成された構造が持つ骨格を考慮した。

生成された構造が活性を持つ確率は、Random Forest で構築した判別モデルの予測確率を利用した。Random Forest の回帰木の数は 900 とし、記述子には Morgan Fingerprint を利用した。Morgan Fingerprint の半径は 2、ビット数は 512 とした。

表 3.7 モデルのアーキテクチャ

| | 層の種類 | 層数 | パラメータ | 活性化関数 |
|---|---|---|---|---|
| Encoder | Embedding | 1 | 次元数 16 | — |
| | Conv1D | 3 | チャネル数 64, フィルタサイズ [9, 9, 10] | SELU |
| | Dense | 2 | 次元数 32 | ReLU |
| | Dense | 1 | — | Softmax |
| Decoder | Dense | 1 | 次元数 32 | ReLU |
| | GRU | 2 | 次元数 64 | ReLU |
| | Dense | 2 | 次元数 32 | ReLU |
| | Dense | 1 | — | Softmax |
| Discriminator | Dense | 3 | 次元数 32 | ReLU |
| | Dense | 1 | — | Softmax |

Random Forest で構築した判別モデルの精度を 5-fold クロスバリデーションで評価したところ、AUC = 0.692 であった。この値自体は高くはないため、予測確率の値そのものを信用するのは難しい。しかし、予測確率の分布はある程度信頼できると考え、これを評価値のひとつとした。

生成構造の構造式としての評価は、検証用データとして訓練には用いていない既知医薬品が持つ分子骨格と比較することで行った。

### 3.3.3 結果と考察

#### 潜在変数の可視化

図 3.4 に、ChEMBL22、ADRA2A それぞれを AAE の潜在変数空間中にプロットした結果を示す。ChEMBL22 は潜在変数空間中で 2 次元のガウス分布全体に射影されたのに対し、ADRA2A は偏った場所に射影された。これは、AAE によって活性のある分子同士が近づくように学習が行われたことを示す。ADRA2A 付近からサンプリングを行い、Decoder に通せば活性のある新規構造を生成することができた。

#### 生成構造の活性予測値分布

図 3.5 に ChEMBL22、ADRA2A それぞれの分布からサンプリングして得た構造それぞれに対し、Random Forest を用いて予測した活性予測値のヒストグラムを示す。横軸が活性のある確率を、縦軸が正規化された度数を表す。ADRA2A 付近からサンプリングした構造は、ChEMBL22 付近からサンプリングした構造に比べ活性予測値がより高い傾向にあった。

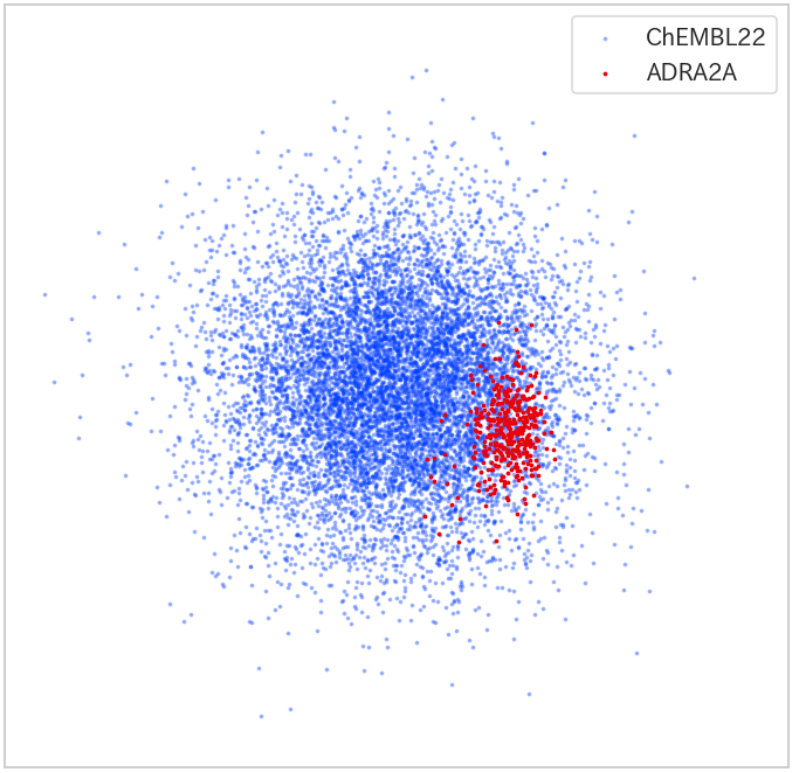



図 3.4 潜在変数への ChEMBL22 と ADRA2A の射影

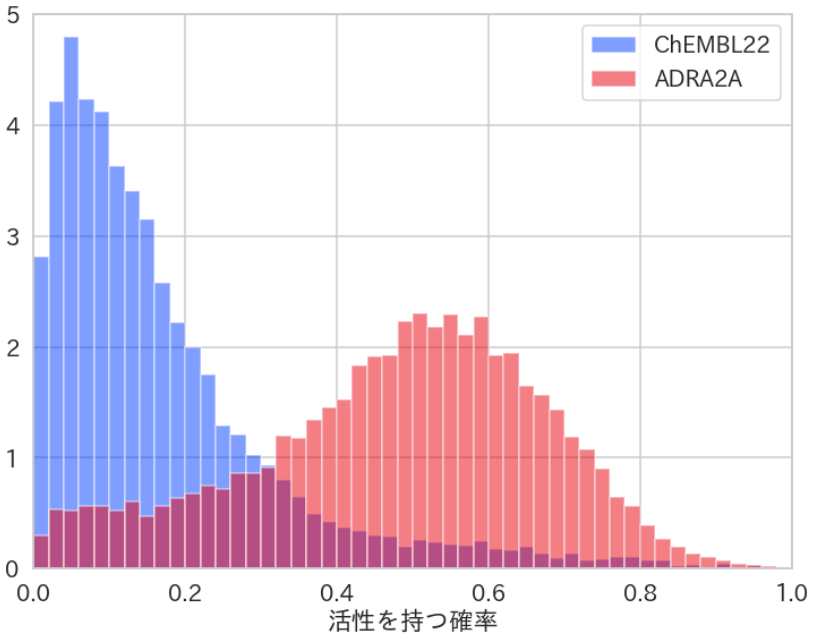



図 3.5 AAE で生成された構造の活性予測値分布

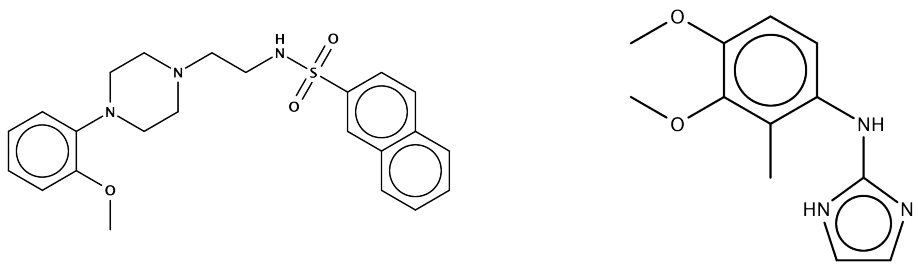
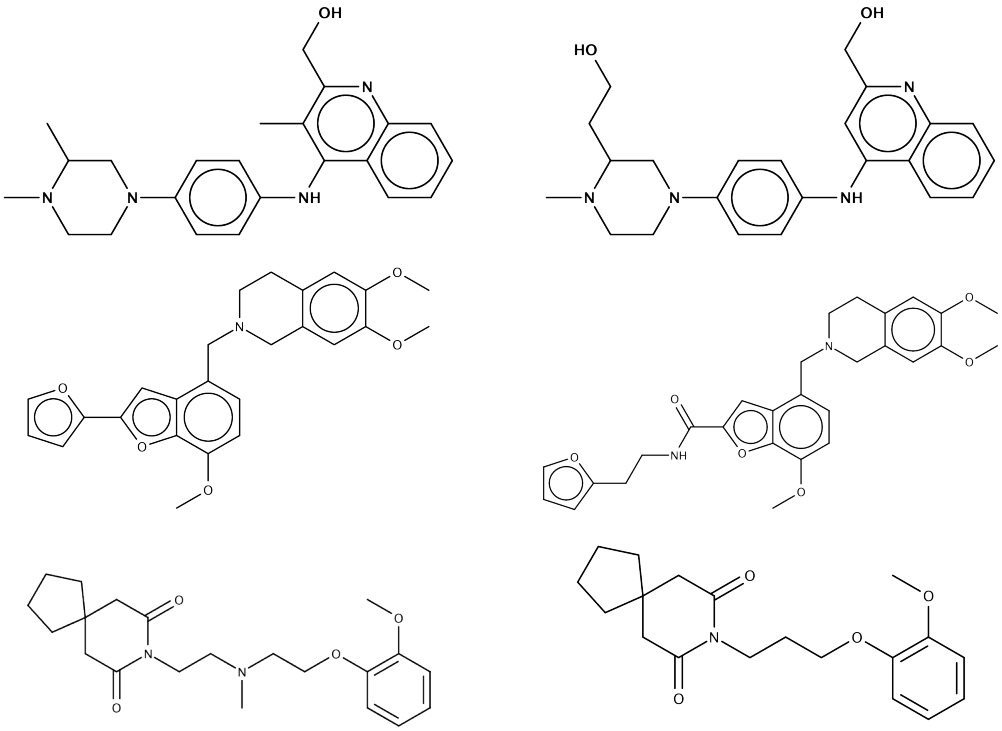


図 3.6 検証構造と同一の構造を生成できたパターン



(a) 検証構造　　(b) 生成構造

図 3.7 検証構造と同様の骨格を持った構造を生成できたパターン

**生成構造の例**

図 3.6, 3.7 に提案手法によって生成された構造の一例を示す。これらは Random Forest によって活性のある確率が 0.8 以上とされた構造である。図 3.6 は検証構造と全く同一の構造を生成できたパターン、図 3.7 は検証構造と同様の骨格を持った分子を生成できたパターンである。生成された構造それぞれについて、半径 2 の Morgan Fingerprint で類似度を計算し、最も類似している検証構造と併せて示した。提案手法によって新規医薬品構造候補を生成できたことが確認された。

3.2 節に比べて良い結果が出た理由として、AAE 自体の特徴と段階の少なさの 2 点が挙げられる。

VAE は正則化のロスを解析的に計算する必要があるため、各サンプルに対応するのは分布の平均と分散である。すなわち、潜在変数で同じ点であっても、複数のサンプルの分布が重なりあう可能性がある。1 つの点が複数の構造に対応する場合、復元誤差が大きくなり、学習が不安定になる。一方、AAE は正則化に Discriminator を利用するため、各サンプルは 1 点と対応させれば十分である。1 サンプルと 1 点が対応するため、Decoder の学習がスムーズに進むと考えられる。

段階の数という点に着目すると、提案手法で用いた VAE は M1、M22 段階のモデルであり、3.2 節で述べたように復元誤差が蓄積する問題があると考えられる。AAE では 1 段階で教師付きデータと教師なしデータを学習させることができるため、復元誤差が乗る余地が少ない。以上の理由により、VAE よりも AAE の方が良い結果を示したと考えられる。

### 3.3.4 課題と解決の方針

#### 骨格の多様性の低さ

AAE によって生成される構造には、訓練データに類似の構造が含まれるという欠点があった (図 3.6)。AAE によって妥当な構造生成はできているものの、生成された構造の骨格の多様性については課題が残るということが分かった。

これは活性値情報が存在するデータの少なさに由来すると考えられる。活性値情報が存在するデータが少ない場合、活性を落とさずに行うことができる構造変化のパターンの数が少なくなる。構造変化のパターンが少ないため、骨格から変化させた構造を生成することが難しい。逆に考えれば、構造変化のパターンを増やすためには活性値情報付きのデータを増やす必要がある。タンパクを絞って活性値情報を増やすのは難しいため、類似するタンパクに対するリガンドデータを用いてマルチモーダル的に学習を行うことで構造変化のパターンを増やすことができると期待される。

副作用の検討も同様の枠組みでできると考えられる。結合してほしくないタンパクに対するリガンドデータを学習データに加え、Discriminator にラベルを与えることで潜在変数空間上で望ましい構造と望ましくない構造を分離することができると考えられる。

#### GAN の枠組み全体に共通する学習の難しさ

また、GAN の枠組みは一般に学習が難しいことが知られている。本研究の場合も同様で、3.3.1 節に示した通り、最適化手法ひとつとっても Encoder、Decoder、Discriminator すべての学習がうまく進むようにパラメータを調節する必要がある。各ネットワークのパラメータについても同様で、パラメータを変更すると学習がうまく進まなかったり、場合によってはパラメータを変更していなくても学習が失敗することさえある。一般にランダム性を含んだモデルの場合、乱数のシード値を固定することで再現性を担保することができる。しかし、深層学習モデルを複数の GPU を用いて学習を行った場合、乱数のシード値

表 3.8 PUGAN で検討したパラメータ

| | 項目 | パラメータ候補 |
|---|---|---|
| 層の種類 | Generator の入力層 | Dense |
| | Generator の隠れ層 | LSTM, GRU |
| | Generator の出力層 | Softmax |
| | Discriminator の入力層 | Embedding |
| | Discriminator の隠れ層 | Dense |
| | Discriminator の出力層 | Softmax |
| 各層のパラメータ | Embedding の次元数 | 16 |
| | Dense の次元数 | 32, 64 |
| | LSTM の次元数 | 16, 32, 64 |
| | GRU の次元数 | 16, 32, 64 |
| | Conv1D のチャネル数 | 16, 32, 64 |
| | Conv1D のフィルタサイズ | 9, 10 |
| | 活性化関数 | ReLU, SELU, Tanh |
| 最適化 | バッチサイズ | 512 |
| | 最適化手法 | SGD, Adam |
| | 学習率 | 5e-4, 2.5e-4, 1e-4 |

を固定していても厳密な再現を取ることが難しい。

GAN の学習を安定させるテクニックは近年多く提案されている [28, 29] が、本研究では試せていない。これらのテクニックを適用することで改善が見込める可能性がある。

## 3.4 PUGAN を用いた構造提案

本節では PUGAN を適用した際の結果を示す。

PUGAN を適用して構造生成を行い、生成された構造が活性を持つ確率の分布や、生成された構造の骨格について検討した。

### 3.4.1 使用したアーキテクチャ

表 3.8 に PUGAN のアーキテクチャで検討したパラメータを示す。

表 3.9 に採用されたパラメータを示す。最適化手法は Positive Generator と Negative Generator には Adam を利用し、学習率は $5 \times 10^{-4}$ を 採用した。Positive Discriminator と Unlabeled Discriminator、Negative Discriminator の 3 つには SGD を利用し、学習率は $2.5 \times 10^{-4}$ を 採用した。GAN に基づくモデルでは Generator の学習率は高く、Discriminator の学習率は低く設定することが多いが、最適な学習率の組み合わせを 5 つのネットワー

表 3.9 モデルのアーキテクチャ

| | 層の種類 | 層数 | パラメータ | 活性化関数 |
|---|---|---|---|---|
| Positive Generator | Dense | 1 | 次元数 64 | Tanh |
| | GRU | 3 | 次元数 64 | ReLU |
| | Dense | 1 | 次元数 32 | ReLU |
| | Dense | 1 | — | Softmax |
| Negative Generator | Dense | 1 | 次元数 64 | Tanh |
| | GRU | 3 | 次元数 64 | ReLU |
| | Dense | 1 | 次元数 32 | ReLU |
| | Dense | 1 | — | Softmax |
| Positive Discriminator | Embedding | 1 | 次元数 16 | — |
| | Conv1D | 2 | チャネル数 32, フィルタサイズ [9,9] | ReLU |
| | Dense | 1 | 次元数 32 | ReLU |
| | Dense | 1 | — | Softmax |
| Unlabled Discriminator | Embedding | 1 | 次元数 16 | — |
| | Conv1D | 2 | チャネル数 32, フィルタサイズ [9,9] | ReLU |
| | Dense | 1 | 次元数 32 | ReLU |
| | Dense | 1 | — | Softmax |
| Negative Discriminator | Embedding | 1 | 次元数 16 | — |
| | Conv1D | 2 | チャネル数 32, フィルタサイズ [9,9] | ReLU |
| | Dense | 1 | 次元数 32 | ReLU |
| | Dense | 1 | — | Softmax |

クに対して調整するのは難しい。そこで、AAE の学習において用いたパラメータを転用することで探索を省略した。

### 3.4.2 生成構造の評価方法

生成された構造の評価には 3.3 節と同様、生成された構造が活性を持つ確率および生成された構造が持つ骨格を考慮した。

生成された構造が活性を持つ確率には 3.3.2 節と同様、Random Forest で構築した判別モデルの予測確率を利用した。ハイパーパラメータも同様に回帰木の数は 900 とし、記述子には Morgan Fingerprint を利用した。Morgan Fingerprint の半径は 2、ビット数は 512 とした。

生成構造の構造式としての評価は、検証用データとして訓練には用いていない既知医薬品が持つ分子骨格と比較することで行った。

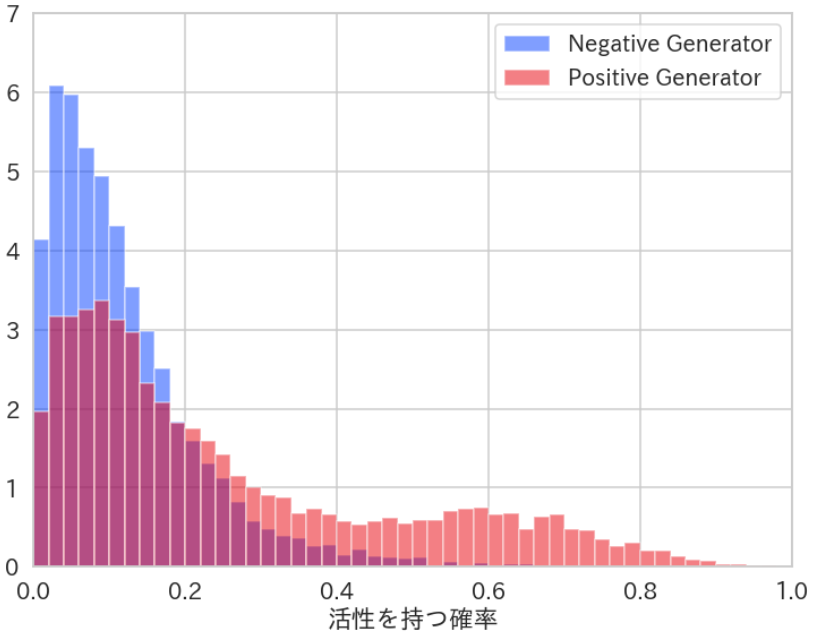



図 3.8 PUGAN で生成された構造の活性予測値分布

### 3.4.3 結果と考察

**生成構造の活性予測値分布**

図 3.8 に Positive Generator と Negative Generator が生成した構造の活性予測値のヒストグラムを示した。横軸が活性のある確率を、縦軸が正規化された度数を表す。図 3.5 と比較すると AAE に比べ PUGAN を適用した場合は生成された構造群の間に大きな違いがないことが分かった。

**生成構造の例**

図 3.9 に PUGAN の生成構造のうち、検証構造と同一の構造を生成できたものを示す。図 3.10 に PUGAN の生成構造のうち、検証構造と類似する骨格を持っているものの一部を示す。PUGAN を利用した場合も、医薬品類似の化合物を生成できることが分かった。

### 3.4.4 課題と解決の方針

図 3.11 に PUGAN の生成構造のうち、適切でないと思われる構造の例を示す。これらの構造は活性が高い構造によく含まれる部分構造を骨格に複数付加することによって生成していると考えられる。

このように適切でないと思われる構造が生成される原因として考えられるのは、Discriminator が弱いということである。GAN を元にした手法を学習する際は学習が容易な

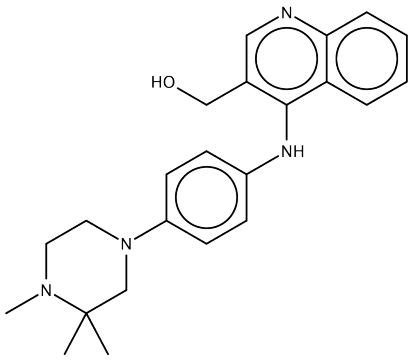
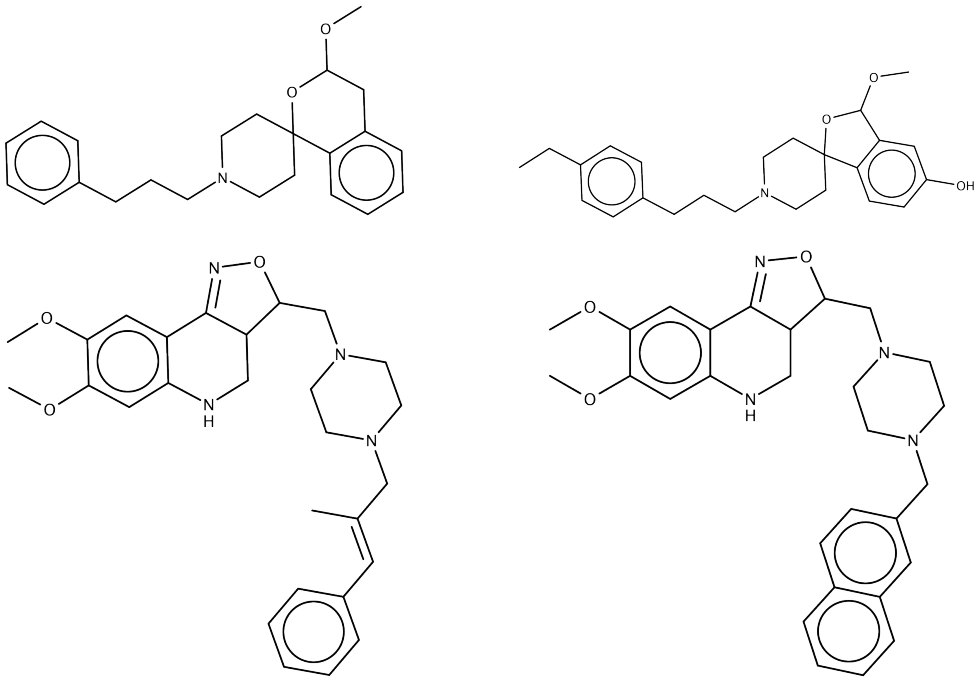



図 3.9 検証構造と同一の構造を生成できたパターン




(a) 検証構造　　(b) 生成構造

図 3.10 検証構造と同様の骨格を持った構造を生成できたパターン

Discriminator を意図的に弱く設計することでスムーズに学習を進めることが多い。しかし、今回の場合のように同じ置換基が大量についている構造を分類するためには Discriminator を少し強く設計しても良いかもしれない。

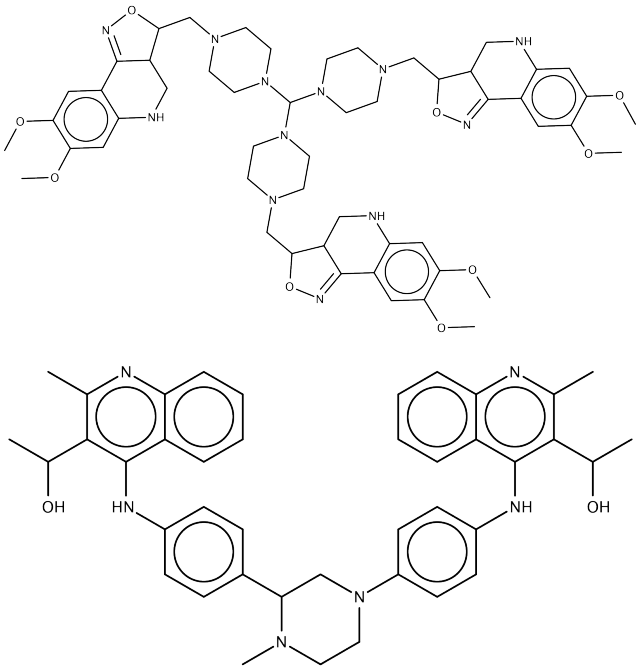


図 3.11 適切でないと思われる構造の例

表 3.10 1,000 件サンプリングした際の各モデルの結果

| モデル | 正しい SMILES の割合 | 検証構造と一致した構造数 |
|---|---|---|
| VAE | 35.4% | 0 件 |
| AAE | 77.3% | 6 件 |
| PUGAN | 53.1% | 1 件 |

## 3.5 提案手法同士の比較

表 3.10 に分布に従って 1,000 件サンプリングした際に正しい SMILES が生成する割合と、検証構造と一致する構造が生成した件数を示す。AAE が正しい SMILES が生成する割合、検証構造と一致した数ともに最も良い性能を示すことが分かった。

先行研究 [7] で示されている正しい SMILES の割合は 77%〜78% 程度であるため、AAE を利用した際の正しい SMILES の割合 77.3% は妥当な数字であると言える。

# 第 4 章

# 結論

## 4.1 まとめ

大量の教師なしデータと少量の教師付きデータを組み合わせた半教師付き学習を化学データに適用した。深層生成モデルには VAE、AAE、PUGAN の 3 種類を用いた。

3.2 節では VAE を適用した。活性値の回帰を行い、Random Forest での教師付き学習と比較して良好な予測を行うことができた。一方で VAE で構造生成を行った場合は炭素鎖を伸ばしたり末端の原子の置換を行ったりなど、本質的でない構造変化を施した構造が多く生成されることが分かった。これは活性を実数値で評価したことで、許容される活性値の変化が小さくなってしまったことに原因があると考えられる。

3.3 節では AAE を適用した。潜在変数の可視化によって、ChEMBL22 の分布にくらべて ADRA2A のデータセットを限られた領域に押し込めることができ、その付近からサンプリングしてきた構造はより活性を持ちやすいことを示した。これは化学空間全体を探索するのに比べ、提案手法を利用すると効率的な医薬品構造が設計できることを意味する。構造生成では検証構造 (訓練データには含んでいないが、医薬品と判明している構造) と完全に同一もしくは同様の骨格を持っている構造を生成することができた。提案手法が新規医薬品構造を提案する能力を持っていることを示した。

3.4 節では PUGAN を適用した。PUGAN はラベルが付いていないデータを「活性がある可能性もあるし、ない可能性もある」と明示的に扱うことができるため、半教師付き学習として有用であることを期待していた。しかし、生成された構造には SMILES として妥当でない構造が多く含まれ、うまく学習が進んでいないことが示唆された。原因としては PUGAN のモデルの複雑さが考えられる。PUGAN は 2 つの Generator と 3 つの Discriminator が含まれる複雑なモデルであり、それらを協調して学習を進めなければならない。そのため、細かいパラメータの変更やデータセットの性質によって大きく学習が不安定になってしまう。今回の検証の範囲では AAE を上回る結果を得ることはできなかった。

3.5 節では各手法の比較を行った。SMILES として妥当な構造が生成された割合、活性の予測値が 0.5 を越えた構造の割合、検証用データと全く同じ構造を生成した割合を比較し、AAE が最も良い性能を発揮することを示した。

## 4.2 今後の展望

今後の展望としては、生成される構造の多様性を増すことが挙げられる。本研究ではデータが少ない状況でも利用可能な手法を提案した。しかし、活性値情報付きのデータが少ないと活性を失わずに構造変化させるパターンを学習することが難しい。この問題に対処するには、他のタンパクに対する活性値情報付きのデータセットを併せて利用し、マルチモーダル的に学習を行うことが考えられる。学習できる構造変化のパターンを増やすことで生成構造の多様性を増すことができると期待される。

また、今回は構造生成を SMILES を生成することで行った。SMILES を生成するにはモデルに SMILES の文法を学習させる必要がある。一方で近年ではグラフを直接生成する手法も提案されている。グラフの生成モデルを化学データに適用した場合についても検討を行うべきであろう。

# 付録 A

# 実装に関する補足情報

## A.1 使用ライブラリ

実験に用いたライブラリと各々のバージョンを以下に示す。

- pytorch: 0.4.1 [30]
- keras: 2.1.3 [31]
- scikit-learn: 0.19.1 [32]
- rdkit: 2017.09.3.0 [26]

深層学習ライブラリとしては実験の初期には Keras[31] を、以降は PyTorch[30] を利用した。Keras は既にライブラリに組み込まれている手法を手早く実装するのに向いているが、GAN などの特殊な学習をさせる手法を実装するのには不向きである。PyTorch は Python の作法に従って柔軟にネットワークを定義することができ、複雑な学習方法にも対応可能である。本研究では PyTorch のバージョンは 0.4.1 を用いたが、現在では 1.0.0 が公開されている。今後生成モデルに関連する実験を行う際は 1.0.0 以降を用いるのが望ましい。

# 謝辞

本研究のディレクションにあたって指導をいただきました船津公人教授に心より感謝いたします。談話会などで先生の質疑に答えているうちにするすると研究の方向性がまとまっていく感覚はなかなか心地よいものでした。講義や講演などでお忙しいなか、本当にありがとうございました。

小寺正明准教授には主に隔週のミーティングでアドバイスを多くいただきました。実験者としての視点も併せて伝えていただいたことで一層のブラッシュアップへとつながったと感じています。修士 2 年になってからのアカデミックな文章の指導は先生が中心となってくださり、たいへんためになりました。

田中健一助教には日々のディスカッションでお世話になりました。各種統計手法に始まり、研究の進め方、まとめ方、伝え方に至るまで。学部 4 年生からの 3 年間で多くのことを学ばせていただきました。

秘書の岡村さんには日々の生活を支えてくださったことに対して感謝の意を述べたいと思います。何度も何度も「ホワイトボードのマーカーください」とお願いに行ってすみませんでした。ホワイトボードのマーカーは箱でストックしていただけると幸いです。

船津・小寺研究室の研究員、学生のみなさま、本当にありがとうございました。先輩方には考えをぶつける楽しさを、後輩たちには知識を伝える喜びを、それぞれ学ぶことができました。この研究室で過ごした 3 年間は私の宝物です。

同期の菅原くん、井上くん、見内くん、Seol さん、天野くん、松本くん、菱沼さん、楽しかったです。4 月以降もまたどこかで会いましょう。菅原くんは 4 月からもよろしく。

最後に、私の歩む道を静かに見守り、支えてくださった両親に深く感謝の意を示して、謝辞とさせていただきます。

# 参考文献


1. Miyao, T., Kaneko, H. & Funatsu, K. Inverse QSPR/QSAR Analysis for Chemical Structure Generation (from y to x). *J. Chem. Inf. Model.* **56,** 286–299. ISSN: 15205142. https://gateway.itc.u-tokyo.ac.jp:11030/doi/pdfplus/10.1021/acs.jcim.5b00628 (2016).
2. Miyao, T., Funatsu, K. & Bajorath, J. Exploring differential evolution for inverse QSAR analysis. *F1000Research* **6,** 1285. https://f1000research.com/articles/6-1285/v1 (July 2017).
3. Kingma, D. P., Rezende, D. J., Mohamed, S. & Welling, M. Semi-Supervised Learning with Deep Generative Models. *arXiv.org* **cs.LG,** 1–9. ISSN: 10495258. arXiv: 1406.5298. http://arxiv.org/abs/1406.5298 (June 2014).
4. Doersch, C. Tutorial on Variational Autoencoders. *arXiv.org*. arXiv: 1606.05908. http://arxiv.org/abs/1606.05908 (June 2016).
5. Gómez-Bombarelli, R. *et al.* Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Cent. Sci.* **4,** 268–276. ISSN: 2374-7943. arXiv: 1610.02415. http://arxiv.org/abs/1610.02415%20http://pubs.acs.org/doi/10.1021/acscentsci.7b00572 (Feb. 2018).
6. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I. & Frey, B. Adversarial Autoencoders. *arXiv.org*. arXiv: 1511.05644. http://arxiv.org/abs/1511.05644 (Nov. 2015).
7. Blaschke, T., Olivecrona, M., Engkvist, O., Bajorath, J. & Chen, H. Application of Generative Autoencoder in De Novo Molecular Design. *Mol. Inform.* **37,** 1700123. ISSN: 1868-1751. arXiv: 1711.07839. http://doi.wiley.com/10.1002/minf.201700123%20http://www.ncbi.nlm.nih.gov/pubmed/29235269%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5836887 (Jan. 2018).
8. Hou, M., Chaib-Draa, B., Li, C. & Zhao, Q. Generative Adversarial Positive-Unlabelled Learning. arXiv: arXiv:1711.08054v2. https://arxiv.org/pdf/1711.08054.pdf.

9. Rumelhart, D., Hinton, G. & Williams, R. Learning representations by back-propagating errors. *Nature* **323,** 533–536. ISSN: 0028-0836. http://www.nature.com/doifinder/10.1038/323533a0 (1986).
10. C. Duchi, J., Hazan, E. & Singer, Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res.* **12,** 2121–2159 (2011).
11. Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. *arXiv.org*. arXiv: 1412.6980. http://arxiv.org/abs/1412.6980 (Dec. 2014).
12. Agarap, A. F. Deep Learning using Rectified Linear Units (ReLU). *arXiv.org*. arXiv: 1803.08375. http://arxiv.org/abs/1803.08375 (Mar. 2018).
13. Han, J. & Moraga, C. in, 195–201 (Springer, Berlin, Heidelberg, 1995). http://link.springer.com/10.1007/3-540-59497-3%7B%5C_%7D175.
14. Klambauer, G., Unterthiner, T., Mayr, A. & Hochreiter, S. Self-Normalizing Neural Networks. *arXiv.org*. arXiv: 1706.02515. http://arxiv.org/abs/1706.02515 (June 2017).
15. Goodfellow, I. J. *et al.* Generative Adversarial Networks. *arXiv.org*. arXiv: 1406.2661. http://arxiv.org/abs/1406.2661 (June 2014).
16. Cruse, H. & Holk. *Neural networks as cybernetic systems* 167. ISBN: 0865776725. https://dl.acm.org/citation.cfm?id=525023 (G. Thieme Verlag, 1996).
17. Hochreiter, S. & Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **9,** 1735–1780. ISSN: 0899-7667. http://www.mitpressjournals.org/doi/10.1162/neco.1997.9.8.1735 (Nov. 1997).
18. Gers, F. A., Schmidhuber, J. & Cummins, F. Learning to Forget: Continual Prediction with LSTM. *Neural Comput.* **12,** 2451–2471. ISSN: 0899-7667. http://www.mitpressjournals.org/doi/10.1162/089976600300015015 (Oct. 2000).
19. Gers, F. & Schmidhuber, J. *Recurrent nets that time and count* in *Proc. IEEE-INNS-ENNS Int. Jt. Conf. Neural Networks. IJCNN 2000. Neural Comput. New Challenges Perspect. New Millenn.* (IEEE, 2000), 189–194 vol.3. ISBN: 0-7695-0619-4. http://ieeexplore.ieee.org/document/861302/.
20. Cho, K. *et al.* Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. arXiv: 1406.1078. http://arxiv.org/abs/1406.1078 (June 2014).
21. Ranzato, M., Chopra, S., Auli, M. & Zaremba, W. Sequence Level Training with Recurrent Neural Networks. arXiv: 1511.06732. http://arxiv.org/abs/1511.06732 (Nov. 2015).
22. Bengio, S., Vinyals, O., Jaitly, N. & Shazeer, N. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. arXiv: 1506.03099. http://arxiv.org/abs/1506.03099 (June 2015).

23. Gaulton, A. *et al.* ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic Acids Res.* **40,** D1100–7. ISSN: 1362-4962. http://www.ncbi.nlm.nih.gov/pubmed/21948594%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3245175 (Jan. 2012).
24. Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S. & Coleman, R. G. ZINC: A Free Tool to Discover Chemistry for Biology. *J. Chem. Inf. Model.* **52,** 1757–1768. ISSN: 1549-9596. http://pubs.acs.org/doi/10.1021/ci3001277 (July 2012).
25. Sterling, T. & Irwin, J. J. ZINC 15 – Ligand Discovery for Everyone. *J. Chem. Inf. Model.* **55,** 2324–2337. ISSN: 1549-9596. http://pubs.acs.org/doi/10.1021/acs.jcim.5b00559 (Nov. 2015).
26. Landrum, G. *RDKit: Open-source cheminformatics* http://www.rdkit.org (2018).
27. Breiman, L. Random Forests. *Mach. Learn.* **45,** 5–32. ISSN: 08856125. http://link.springer.com/10.1023/A:1010933404324 (2001).
28. Salimans, T. *et al.* Improved Techniques for Training GANs. arXiv: 1606.03498. http://arxiv.org/abs/1606.03498 (June 2016).
29. Neyshabur, B., Bhojanapalli, S. & Chakrabarti, A. Stabilizing GAN Training with Multiple Random Projections. *arXiv.org.* arXiv: 1705.07831. http://arxiv.org/abs/1705.07831 (May 2017).
30. Paszke, A. *et al.* Automatic differentiation in PyTorch (2017).
31. Chollet, F. *et al. Keras* 2015. https://keras.io.
32. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **12,** 2825–2830 (2011).