Generating Rationale for Molecular Prediction Using Reinforcement Learning

by

Benson Chen

B.S. in Engineering, School of Engineering and Applied Sciences, University of Pennsylvania (2016)
B.S. in Economics, The Wharton School, University of Pennsylvania (2016)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2019

Signature redacted

Author .....
Department of Electrical Engineering and Computer Science
October 22, 2018

Signature redacted

Certified by .....
Regina Barzilay
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Signature redacted

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Generating Rationale for Molecular Prediction Using Reinforcement Learning

by

Benson Chen

Submitted to the Department of Electrical Engineering and Computer Science
on October 22, 2018, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

This thesis studies generation of rationale for neural prediction problems using reinforcement learning. In particular, we focus on neural predictions in chemical property prediction tasks. We design a reinforcement learning agent that learns to incrementally extract the important regions of molecular graphs, and construct a predictor trained on only the selected regions. The ability for the model to predict a property based only on the partial graph exemplifies the importance of these substructures and therefore can be interpreted as rationales for the prediction task. We test our reinforcement learning model on several chemical datasets and show that our model can generate meaningful rationales while maintaining good predictive performances.

Thesis Supervisor: Regina Barzilay
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

I would like to thank Regina for being a great advisor and helping me tremendously on my PhD journey. In particular, she is always very receptive to listening to my ideas and provding thoughtful insights.

I would also like to thank Tommi for being a supportive and resourceful advisor. He has helped me reformulate my ideas, and refine them into better ones.

I would also like to thank all my labmates in the group, who have spent countless hours brainstorming and talking to me about ideas. Without the support from the people around me, my research journey would not be possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Recently, deep learning has been successfully applied to the development of predictive models relating chemical structures to physical or biological properties, outperforming existing methods [7, 15]. However, these gains in accuracy have come at the cost of interpretability. Often, complex neural models operate as black boxes, offering little transparency concerning their inner workings.

Interpretability plays a critical role in many areas including cheminformatics. Consider, for example, the problem of toxicity prediction. Over 90% of small molecule drug candidates entering Phase I trials fail due to lack of efficacy or due to adverse side effects. In order to propose a modified compound with improved properties, medicinal chemists must know which regions of the molecule are responsible for toxicity, not only the overall level of toxicity [2]. We call the key molecular substructures relating to the outcome *rationales*. In traditional cheminformatics approaches such as pharmacophore mapping, obtaining such a rationale behind the prediction is an intrinsic part of the model [23, 6, 13].

1.2 Contribution

In this paper, we propose a novel approach to incorporate rationale identification as an integral part of the overall property prediction problem. We assume access to the same training data as in the original prediction task, without requiring annotated rationales. At the first glance, the problem seems solvable using existing tools. For instance, attention-based models offer the means to highlight the importance of individual atoms for the target prediction. However, it is challenging to control how soft selections are exploited by later processing steps towards the prediction. In this sense, the soft weighting can be misleading. In contrast, hard selection confers the guarantee that the excluded atoms are not relied upon for prediction. The hard selection of substructures in a molecule is, however, a combinatorial problem. Prior approaches circumvent this challenge by considering a limited set of predefined substructures (typically of 1-6 atoms), like the ones encoded in some molecular fingerprints [6]. Ideally, we would like the model to derive these structures adaptively based on their utility for the target prediction task.

We formulate the problem of selecting important regions of the molecule as a reinforcement learning problem. The model is parametrized by a graph convolutional network (GCN) over a molecular graph in which the atoms and bonds are the nodes and edges of the graph, respectively. Different from traditional reinforcement learning methods that have a reward function provided by the environment, our model seeks to learn such a reward function alongside the reinforcement learning algorithm. More generally, our model works as a search mechanism for combinatorial sets, which readily expands to applications beyond chemistry or graphs.

Our iterative construction of rationales provides several advantages over standard architectures. First, sequential selection enables us to incorporate contextual features associated with past selections, as well as global properties of the whole molecule. Second, we can explicitly enforce desirable rationale properties (e.g., number of substructures) by including appropriate regularization terms in the reward function.

1.3 Outline

- **Chapter 2** introduces related works on the foray into deep learning in the chemistry field, as well as interpretability models in deep learning.
- **Chapter 3** introduces the graph convolutional networks, the basis of deep learning models for chemical problems, as well as the reinforcement learning background for the models used in this work.
- **Chapter 4** details the rationale selection algorithms used in this work, including both a concurrent selection model, and the sequential selection model using reinforcement learning.
- **Chapter 5** presents the experimental setup, results and analysis on two chemistry property prediction datasets on solubility and the Ames test.
- **Chapter 6** summarizes the work of this thesis and highlights future directions for this work.

Chapter 2

Related Works

2.1 Deep Learning in Chemistry

One of the major shifts in chemical property prediction is towards the use of deep learning. The existing models fall into one of two classes. The first class of models is based on an expert-constructed molecular representation such as fingerprints that encapsulate substructures thought to be important and a range of molecular properties [29, 25]. These models are not well suited for extracting rationales because desired structures may not be part of the fingerprint. Moreover, it may be challenging to attribute properties recorded in fingerprints to specific substructures in the molecule. One would have to restrict the feature space of the fingerprint, which can harm the performance of the model.

The second class of models move beyond traditional molecular fingerprints, instead learning task-tailored representations through graph convolutional networks (GCN). Specifically, these networks learn a continuous representation of the molecule [7, 15], capturing an information-rich latent space of the molecules. While several complex architectures have been proposed [11], they can be boiled down to a form of message-passing networks, in which atoms can be viewed as passing messages to their neighbors at each layer [8]. Following this direction, our work is also based on learned molecular representations. Our focus, however, is on augmenting these models with rationales. As articulated in the introduction, the task is challenging due to the number of can-

didate substructures and the need to attribute properties aggregated in convolutions to individual atoms.

2.2 Learning Rationales

The topic of interpretability has recently gained significant attention [21, 16]. The proposed approaches cover a wide spectrum in terms of the desired rationales and the underlying methods. Work in this area include visualization of activations in the network [12, 22], and examination of the most influential data examples to provide interpretability [17]. Attention-based models have also been widely used to extract interpretability [1, 4, 24, 3, 27, 28]. Our work is mostly closely related to approaches focused on extractive rationales [20, 26]. [20] presents a model to extract parts of text as rationale, but their model does not readily generalize to graphs, and the sequential nature of our model can place a meaningful ordinal ranking over the atom selections.

Chapter 3

Background

3.1 Graph Convolutional Networks

Chemistry problems involving molecules are naturally well-suited as graph problems. Here, the atoms are the nodes, and the bond are the edges of the graph. Convolutional neural networks has been applied to image problems with much success [19], but they also naturally extend to graph problems. For each atom or node, the convolution operation iteratively aggregates neighboring features to creating successive hidden representations. This successive aggregation of neighboring information intrinsically propagates information to connected, but non-adjacent atoms within a graph.

One particular algorithm to conduct this manner of propagation is the message passing neural network [8]. This belief propagation algorithm is traditionally used to perform inference on graphical models, but also works well as a propagation paradigm for a molecular graph. In particular, this model uses the approximate "loopy" belief propagation model since molecular graphs are, in general, not trees, but contains cycles in the form of rings. Next, we will formally describe the graph convolutional network (GCN) model in detail.

Let a_i be the atom feature vector for atom i, and let $b_{i,j}$ be the bond feature vector for the bond between atoms $\{a_i, a_j\}$. We can define $m_{i,j}^l$ as the message from $a_i \rightarrow a_j$ at layer l. $m_{i,j}^l$ can then be recursively defined in the following manner:

$$m_{i,j}^0 = W_{m,i} \cdot [a_i; b_{i,j}] \tag{3.1}$$

$$m_{i,j}^l = \sigma\big(W_{m,h} \cdot \sum_{k \in n(i)-j} m_{k,i}^{l-1}\big) \tag{3.2}$$

Where $n(i)$ is the set of indices of the atoms neighboring atom i, $\{W_{m_i}, W_{m_h}\}$ are weight matrices, and σ is some non-linear activation function. The first message vectors are computed with the raw features, whereas the subsequent messages are iteratively computed from messages in the previous layer. Let h_i be the final atom embedding for an atom i, and let L be the total number of layers for this network. We define h_i as follows:

$$h_i = \sigma\big(W_{m,o} \cdot \sum_{k \in n(i)} [m_{k,i}^L; a_i]\big) \tag{3.3}$$

Where $\{W_{m,o}\}$ is another weight matrix, and we add residual links from the input atom features to construct their embeddings. We aggregate the the individual atom embeddings of a molecule to create a molecular fingerprint: $m = W_{f,o} \cdot \sigma\big(W_{f,h} \cdot \sum_i f(h_i)\big)$. Where f is some aggregation function such as mean, sum or max.

3.2 Incorporating structure in GCNs

Sometimes we may prefer to incorporate structure into the graph convolution networks. In chemistry applications, groups of adjacent atoms are often focused as a single meaningful entity (called functional groups in chemistry). Likewise, we might wish to incorporate such structures in the neural model so that the model may more easily learn the patterns meaningful to humans.

One way to incorporate domain-specific knowledge is to aggregate the atom embeddings of specific important substructures. Suppose $S = \{s_1, s_2..s_n\}$ is the set of groups of atoms that constitute important substructures. We can construct node embedding g_i for group s_i:

$$g_i = \sigma\left(W_{s,o} \cdot \sum_{k \in s_i} h_k\right) \tag{3.4}$$

Where h_k are the atom embeddings from equation 3.3. To compute the embedding for the molecule, we can aggregate over the g_i or a combination of g_i and h_i (for atoms not associated with specific substructures).

3.3 Reinforcement Learning

A reinforcement learning problem can be described by a Markov Decision Process (MDP) characterized by the tuple $\{S, A, T, r\}$. Here, S describes the set of possible states, and A describes the set of possible actions, both of which can be either discrete or continuous. T is the transition operator that characterizes the probabilistic changes of the stochastic environment given the action. r is a function $S \times A \rightarrow \mathbb{R}$ that takes a state and action as input and outputs some scalar reward value.

In a reinforcement learning problem, the agent wants to takes actions that optimizes the sum of rewards. The agent learns a policy $\pi_\theta(s_t)$ that assigns a probability distribution over the action space for the current state s_t that maximizes the cumulative reward $\sum_t r(s_t, a_t)$. θ are the parameters that characterize the policy, such as the weights of neural network. The optimization problem can be succinctly written as:

$$\theta^* = \arg\max \sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim p_\theta(s_t,a_t)}[r(s_t, a_t)] \tag{3.5}$$

Where T is when the agent has hit a terminal state, or has reached the maximally allotted time step. Generally, the agent has no knowledge of the stochastic transitions of the environment, but can either explicitly learn a model for the transition probabilities of the environment (model-based reinforcement learning), or can indirectly incorporate information about the enviornment in its policy (model-free reinforcement learning). In this work, we will mostly focus on the latter, model-free reinforcement learning.

There are two main methods of conducting model-free reinforcement learning. The first tries to explicitly model the value of specific state-action pairs, and derives a policy from those values, such as in Q-learning. The second method tries to explicitly models a policy function, though can still use a value function to learn that policy function. These methods are commonly known as policy gradient methods and we will focus on these methods in this work. Policy gradient methods can be viewed as smooth versions of traditional Q-learning methods, and therefore are often easier to train [9].

In order to solve for equation 3.5, if we let $J(\theta) = \sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim p_\theta(s_t,a_t)}[r(s_t,a_t)]$, we can compute the gradients as:

$$\nabla J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \Big(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \Big) \Big(\sum_{t=1}^{T} R(s_{i,t},a_{i,t}) \Big) \tag{3.6}$$

Where the approximation is due to the fact that we are drawing samples, as the entire policy trajectory space is intractable. From this, we can further simplify the optimization problem:

$$\nabla J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) Q(s_{i,t},a_{i,t}) \tag{3.7}$$

Where $Q(s_{i,t},a_{i,t}) = \sum_{t'=t}^{T} r(s_{i,t},a_{i,t})$ is the sum of future rewards taking action $a_{i,t}$ in state $s_{i,t}$. Because at each time step, only the rewards in the future matters, we can reduce the variance of the gradients by only using the "rewards to go." Furthermore, in practice, subtracting a baseline value from the reward estimate greatly improves training by reducing variance, which leads us to replace $Q(s_{i,t},a_{i,t})$ with $Q(s_{i,t},a_{i,t}) - b(s_{i,t})$, where $b(s_{i,t})$ is some baseline value that is a function of only the state. Generally, $b(s_{i,t})$ is taken as a value estimate of the the state $V(s_{i,t})$, which leads to the following gradient estimate:

$$\nabla J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) (Q(s_{i,t},a_{i,t}) - V(s_{i,t})) \tag{3.8}$$

This is the general actor-critic model, wherein the actor is the policy function

parametrized as π_θ, and the critic is the value function $V(s_{i,t})$ that tries to estimate the value of being in specific states (and is independent of the action chosen). Here, $V(s_{i,t})$ is fit to the sum of sampled rewards with the following L_2 norm loss: $L_V = \frac{1}{2}\sum_i |V(s_{i,t}) - \sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'})|$. And the term $Q(s_{i,t}, a_{i,t}) - V(s_{i,t})$ is commonly referred to as the advantage function $A(s_{i,t}, a_{i,t})$, measuring the value of choosing action $a_{i,t}$ in state $s_{i,t}$ over all other actions. The advantage is useful to consider, because we want the policy to move based on the relative value of specific actions, and not their absolute values.

Chapter 4

Methods

In this chapter, we describe methods to extract rationales from deep learning models applied to property prediction tasks. First, we talk about generating rationales through a concurrent hard selection method, before introducing our sequential algorithm involving reinforcement learning.

4.1 Concurrent Rationale Selection

One way to generate rationales is to train a model to directly predict the important atoms by outputting a mask over the input atoms. A second model is trained to predict on these partial subgraphs. As in [20], we will refer to the former as the generator G and the later as the encoder E. Both G and E will use the graph convolution network described in section 3.1.

From the atom embeddings h_i of the generator G, we compute a mask m_i for each atom node as $m_i = W_{mask_o} \cdot \sigma(W_{mask_h} \cdot h_i)$. From this, we can apply the sigmoid operator to get a probability, $p_i = \frac{e^{m_i}}{1+e^{m_i}}$.

Ultimately, we want a hard selection over the atoms of the molecules, but since casting the probability to a 0-1 value is not differentiable, we apply the gumbel-softmax trick [10]. Using gumbel-softmax, we can compute the new probabilities as:

$$p'_i = \frac{e^{(\log(p_i)+g_i)/\tau}}{1 + e^{(\log(p_i)+g_i)/\tau}} \tag{4.1}$$

Where g_i is sampled from Gumbel$(0, 1)$. When the temperature parameter τ is large, the distribution is more uniform, and when the temperature parameter is small, the distribution is more peaky. During training, we use the soft output probabilities to mask the atom features, so that gradients pass through the mask. During evaluation, we cast the mask to a one-hot vector to enforce the hard selection. As mentioned earlier, both the generator and encoder models are parameterized by the GCN. While the generator differs in its output from the base GCN, the encoder differs in the input. Specifically, we alter equation 3.1 as follows:

$$m^0_{i,j} = W_{m,i} \cdot p'_j \cdot [a_i; b_{i,j}] \tag{4.2}$$

Additionally, the computation of the final atom embeddings is altered in a similar fashion, in which the atom features are scaled (zeroed in evaluation phase) based on p_i's.

$$h_i = \sigma\big(W_{m,o} \cdot \sum_{k \in n(i)} [m^L_{k,i}; p'_i \cdot a_i]\big) \tag{4.3}$$

Note that one feature of this formulation is that we only zero out the input features of atoms/bonds that have been masked out. We do not remove the node entirely from the graph. This way, the convolutional operations can still propagate information from different parts of the molecule. A consequence of this formulation is that the bond features are present as features in the input messages as long as one of its atoms has not been masked out. From the atom embeddings h_i, we can aggregate as in the GCN model to produce a molecular fingerprint m.

As it is, however, this generator/encoder combination will tend towards an optimum where it selects all the atoms due to the fact that the model can never do worse (on training data), when it has more features available. Therefore, we have to add regularization to force the model to choose only the important atoms to add.

In order to penalize the model for selecting too many atoms, we enforce the following penalty, defining $\bar{p} = \frac{1}{N}\sum_{i=1}^{N} p'_i$:

$$L_{selection} = c_{selection} \cdot \log(\bar{p}) + (1 - c_{selection}) \cdot \log(1 - \bar{p}) \tag{4.4}$$

Where $c_{selection} \in [0, 1]$ is a hyperparameter that determines how much of the molecule should be selected (essentially a cross entropy term between the average selection rate in the molecule and a target selection rate).

Let $L_{predict}$ be the loss of the prediction task (cross entropy for classification and mean-squared error for regression). The total loss is then:

$$L = L_{predict} + \lambda_{selection} \cdot L_{selection} \tag{4.5}$$

One caveat occurs when we use nodes (section 3.2) instead of atoms for this formulation. Because a single atom can belong to multiple rings (each of which is a separate node), it can be unclear how the mask for that atom is determined. We use a max heuristic to determine the inclusion of an atom. That is, suppose atom a_i belongs to the set of nodes $\{n_1...n_k\}$, with associated probabilities $\{p_1...p_k\}$. The mask m_i for a_i is computed as $m_i = \max(p_1...p_k)$. In the hard selection case, as long as one of the nodes has not been selected out, the atom's input features will be preserved.

4.2 Sequential Rationale Selection

The next algorithm we will discuss is using reinforcement learning to sequentially select the rationale from molecules. This method works in an iterative fashion in the following manner. Fixing a prediction model, an agent learns to select for the most number of atoms in the molecule without changing the prediction too much. Then, fixing the agent, the prediction model will be retrained on the selections made by the agent.

Both the prediction model and the reinforcement learning agent are parametrized

by GCNs. The reinforcement learning algorithm, at each time step, tries to select one atom to remove from the molecule. As before, removing an atom only implies that the input features are zeroed out. Therefore, at each time step, the action space is the set of atoms that have still not been selected out of the molecule. Additionally, there is a stop action that the agent considers. When the stop action is selected, the episode is terminated. Using the atom embeddings h_i from equation 3.3, we compute the action logits as follows:

$$\alpha_i = W_{\alpha_o} \cdot \sigma\Big(W_{\alpha_h} \cdot h_i\Big) \tag{4.6}$$

These action values are computed as a function of only the atom embeddings, which only captures a semi-local representation. The stop action, however, would likely benefit from knowing information about the whole molecular state, so the stop action α_S is defined with respect to the molecular fingerprint m.

$$\alpha_S = W_{S_o} \cdot \sigma\Big(W_{S_h} \cdot m\Big) \tag{4.7}$$

The action-distribution is a softmax distribution over the available atoms (atoms not selected) and the stop action.

$$\rho_i = \frac{e^{\alpha_i}}{e^{\sum_i \alpha_i + \alpha_S}} \tag{4.8}$$

The value v_t at time step t, used to control the variance in this policy gradient method, also uses the molecular fingerprint, as the value describes the state as a whole, independent of individual atom actions. Therefore, similar to equation 4.7, $v_t = W_{V_o} \cdot \sigma\Big(W_{V_h} \cdot m\Big)$

Perhaps the most important aspect of this model is the reward function that we want to construct such that the agent can learn to pick out the unimportant parts of the molecules. The reward function has three main aspects, listed in table 4.1.

As an arbitrary baseline, the reward for choosing any action but the stop action is 1. If the agent chooses the stop action, or all the atoms have been selected out, the episode terminates, and a reward of 0 is received upon termination. Next, we want to

Name	Reward Value	Description
Default Reward	$+1$	The default reward for the agent not terminating (choosing a valid atom), so that the agent will want to select out as many atoms as possible.
Distance to previous prediction	$\lvert\hat{p}_t - \hat{p}_{t-1}\rvert$	The distance of the new prediction should be close to the previous prediction
Distance to optimal prediction	$\lvert\hat{p}_t - \hat{p}_0\rvert$	The distance of the new prediction should be close to the the prediction on the whole molecule.

Table 4.1: Table of the different components of the reward function. The three components encourages the agent to select out atoms as many as possible such that the predictor does not change its prediction too much.

ensure that the model only selects out the important atoms. Therefore we add two regularization-type terms to the reward function.

Suppose for a given molecule and a given prediction network, $\hat{p}_t$ is the prediction of the network at time t. Then, $\hat{p}_0$ is the prediction given the full molecule, without selecting out any atoms. The first regularization term that we add is a term that computes the distance of the current prediction to the "optimal" prediction for the molecule, which is the prediction using the full molecule: $|\hat{p}_t - \hat{p}_0|$. Additionally, we don't want each subsequent prediction (after removing an atom) to be very different from the previous prediction $|\hat{p}_t - \hat{p}_{t-1}|$. In all, the reward can be summarized as:

$$r_t = \begin{cases} 0 & a_t \text{ is the stop action} \\ 1 - \lambda_{opt} \cdot |\hat{p}_t - \hat{p}_0| - \lambda_{prev} \cdot |\hat{p}_t - \hat{p}_{t-1}| & \text{otherwise} \end{cases}$$

Where $\{\lambda_{opt}, \lambda_{prev}\}$ are hyperparameters controlling the degree of contribution for their respect components. When the λs are large, the agent will prefer to select less atoms, because each more atoms will likely cause greater changes in the prediction network.

After training the agent, fixing the prediction network constant, we fix the agent and train the prediction network on the partial selections of the agent. Namely, at each time step t, we obtain some state $\boldsymbol{o}_t$ which describes which atoms have been

selected out. We can then train the prediction network on each of these states that the agent generates. We choose to train on all states, and not simply the final and/or intial states, because this allows the predictor network to provide a smoother training signal/reward to the agent when the agent is trained. Since the intermediate states yield predictions close to the optimal (as controlled by the rewards), the model should not have a lot of difficulty learning a representation that is semi-agnostic to slight variations to the provided partial molecule. The full algorithm and details can be found in algorithm 1.

Algorithm 1 Sequential rationale selection using reinforcement learning

Initialize predictor model $\boldsymbol{P}$ with parameters $\boldsymbol{\theta}_P$, and pretrain predictor model.
Initialize RL agent parameters $\boldsymbol{\theta}_\pi$ and $\boldsymbol{\theta}_v$ for the actor and critic parameters respectively.
for k = 1, 2 ... N **do**
  repeat
    Sample molecule m
    Initialize step counter t and state $\boldsymbol{o}_0$
    repeat
      Compute action probabilities $\rho_{t,i} = \pi(a_i|\boldsymbol{o}_t; \boldsymbol{\theta}_\pi)$ and value $v_t = V(\boldsymbol{o}_t; \boldsymbol{\theta}_v)$
      Sample action j from ρ_t, and construct new state with $\boldsymbol{o}_t[j] = 0$
      Compute reward r_t by running $\boldsymbol{P}$ with $\boldsymbol{o}_t$
    until terminal o_t or $t == T$

$$R = \begin{cases} 0 & \text{for terminal } \boldsymbol{o}_T \\ v_{T+1} & \text{for non-terminal } \boldsymbol{o}_T \end{cases}$$

    for $t = \{T, T-1 ... 1\}$ **do**
      $R \leftarrow r_t + \gamma R$
      Update $\boldsymbol{\theta}_v$ through value loss: $L_v = (R - v_t(\boldsymbol{o}_t; \boldsymbol{\theta}_v))^2$
      Update $\boldsymbol{\theta}_\pi$ through policy loss: $L_\pi = \log \pi(a_{t,i}|\boldsymbol{o}_t; \boldsymbol{\theta}_\pi)(R - v_t(\boldsymbol{o}_t; \theta_v))$
    end for
  until convergence
  repeat
    Sample molecule m with label l
    Initialize step counter t and state $\boldsymbol{o}_0$
    repeat
      Compute action probabilities $\rho_{t,i} = \pi(a_i|\boldsymbol{o}_t; \boldsymbol{\theta}_\pi)$
      Sample action j from ρ_t, and construct new state with $\boldsymbol{o}_t[j] = 0$
      Update $\boldsymbol{\theta}_P$ for molecule m with label l using state $\boldsymbol{o}_t$
    until terminal o_t or $t == T$
  until convergence
end for

Chapter 5

Experiments

5.1 Experimental Setup

We test our model on several chemistry datasets. The first of which is the solubility dataset , the second is the AMES toxicity dataset.

- **Solubility Dataset:** The solubility dataset contains 1,116 molecules and their aqueous solubility as measured in log Mol/L by [5]. While solubility is very much a macro property of the whole molecule, there are simple functional groups that can often greatly alter the solubility of a molecule. We use this dataset to determine if our model can pick out specific hydrophobic/hydrophillic functional groups that are important to this property.

- **Ames Toxicity Dataset:** The Ames dataset contains 4,325 molecules with their Ames test outcomes. This is a test for mutagenicity of DNA in bacteria. While this property intrinsically has a variety of factors that can cause the mutagenicity phenomenon, certain functional groups can have high implications for this property as noted by [14]

We first present several baseline models, including simple regression models and GCN results using both atoms and nodes (ref section). This establishes the upper bound performance that we expect from these models, as rationale models generally

Feature	Values
Atom Symbol	one-hot vector
Formal Charge	$\{-2, -1, 0, +1, +2\}$
Bond Type	{single, double, triple, aromatic}

Table 5.1: The set of features used in the graph convolutional models.

are expected to perform worse, due to the removal of input features. Our baselines consist of regression with fingerprints, and the concurrent selection rationale model.

Regression Baseline The regression baselines uses Morgan fingerprints with radius 3 from RDKit [18]. Morgan fingerprints present a simple way to represent a molecule by have one-hot encodings of substructural features. Radius of 3 means that these substructure features are found by a depth-first search with maximum depth of 3. We use linear regression for the solubility dataset, and logistic regression for the Ames dataset. Both regressions use a L_2 penalty on the weight norms for regularization.

Graph Convolutional Network Baseline The GCN baseline uses the features in table 5.1. There are two atom features (symbol and formal charge) and one bond feature (bond type). We choose to minimize the feature space, as most other atomic features should be inferrable from these features. Furthermore, having fewer features can allow the model to overfit less, especially with regards to the relatively small solubility dataset. All other neural models also use the same features.

Concurrent Rationale Selection The concurrent rationale selection model (Section 4.1) is trained with atoms and nodes (Section 3.2). For this model, we would like to investigate the performance of the model when differing fractions of molecules are chosen. This is controlled by the hyperparameter $c_{selection}$. Because we don't want the model to have to select exactly $c_{selection}$ of the atoms (which would hugely bias the performance of the model), we can only loosely control the percent of atoms selected. For these models, we vary $c_{selection} \in [0, 1]$, but fix $\lambda_{selection} = 1$, and the temperature

Model	Solubility (RMSE)	Ames (AUC)
Regression	1.166	0.780
GCN	**0.684**	0.814
GCN + nodes	0.699	**0.824**

Table 5.2: Table of the baseline performances of regression models and graph convolutional network with and without nodes.

parameter at 1 as well. We use Adam optimizer with a learning rate of $5e^{-4}$.

Sequential Rationale Selection The sequential rationale selection model using reinforcement learning (Section 4.2) is also trained with both atoms and nodes. The percent of atoms selected for this model is controlled by λ_d, which modifies the importance of the different reward contributions of predictions. When λ_d is high, the RL agent is less likely to choose more atoms, because differences in predictions are more pronounced. Conversely, when λ_d is low, the RL agent is likely to choose more atoms. For the solubility dataset, we try values of $\lambda_d \in [1.0, 2.5]$, and for the Ames dataset, we test values of $\lambda_d \in [0.5, 1.5]$. The networks are trained with Adam optimizer with a learning rate of $5e^{-4}$.

5.2 Experimental Results and Analysis

From table 5.2, we can see that the GCN greatly outperforms the regression model on the solubility task, but is much closer to the regression model for the Ames dataset. The greater performance of the nueral model is expected, as the fingerprint regression models are not nearly as expressive. For instance, one property of solubility is that larger compounds tend to have lower solubility; however, the regression model cannot easily learn such a relationship through the fingerprint inputs. Moreover, the fingerprint inputs are one-hot feature vectors of substructures. However, the existence of multiples of a specific functional group (such as the hydrophillic hydroxyl group), often exhibits additive effects to solubility.

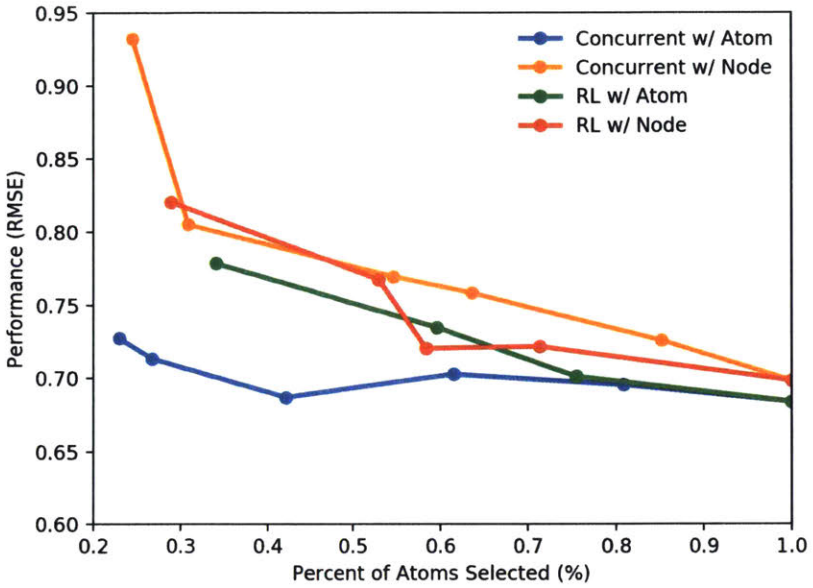
For the solubility dataset, using nodes instead of atoms observes a slight drop

in performance, while for the Ames dataset, using nodes has a positive impact on performance. Generally, for these two problems, using nodes instead of atoms doesn't seem to affect the performance greatly. Next, we look at the performance of our rationale models, both the concurrent and sequential selection models modeled with atoms and nodes.

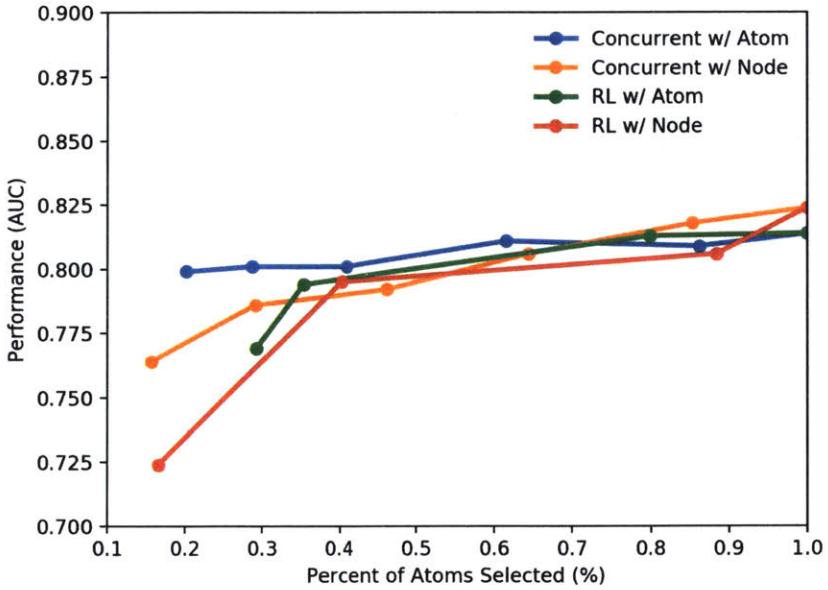
Figure 5-1 graphs the performance of the four models against the percent of atoms selected. We observe the trend that, as the fraction of atoms selected goes down, the performance of the models also decreases. As with before, the models using nodes tend to perform slightly worse than their model counterparts that selects atoms instead. We hypothesize that this is due to the fact that forcing the model to select entire rings, when it does not need to, acts as a regularizer for the set of possible selections, thereby penalizing its performance.

The sequential RL model performs on par with the concurrent model, though slightly better when a higher percent of the atoms are selected, and worse when a smaller fraction of atoms are selected for each molecule. The RL model is a much more powerful model by nature, and is more likely to overfit to extremes when not many atoms are selected.

We can see example rationales for the four models in Figure 5-2 and Figure 5-3 for the solubility and Ames datasets respectively. Although using nodes might decrease the performance of the models slightly, the rationales selected with the models using nodes generally look much nicer as rationales. However, as there are no golden labels for any of the chemical datasets, it is difficult to empirically validate the quality of these rationales. Nevertheless, we see that in Figure 5-2, the model picks up different hydrophobic and hydrophillic groups, prominent contributors to the solubility of a molecule. In Figure 5-3, The models pick up the aromatic nitro group, which according to [14], is highly indicative of toxic molecules.

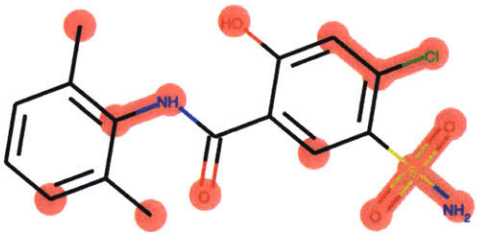


(a) Plot of performance vs percent of atoms selected using atoms and nodes for the solubility dataset.

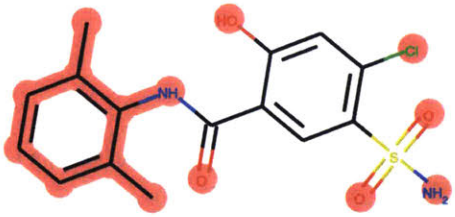


(b) Plots of performance vs percent of atoms selected using atoms and nodes for the Ames dataset

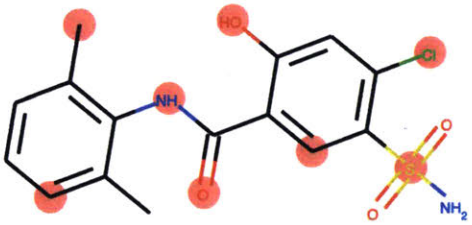
Figure 5-1: Plots of the performance of the concurrent and sequential rationale models for the solubility and Ames datasets. The expected trend reveals that as the fraction of atoms increases, the performance of the models goes up.



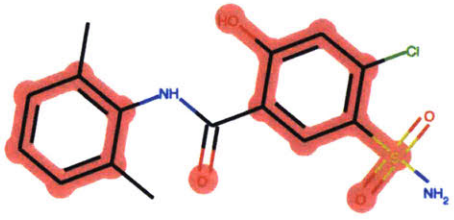
(a) Concurrent model with atoms.



(b) Concurrent model with nodes.

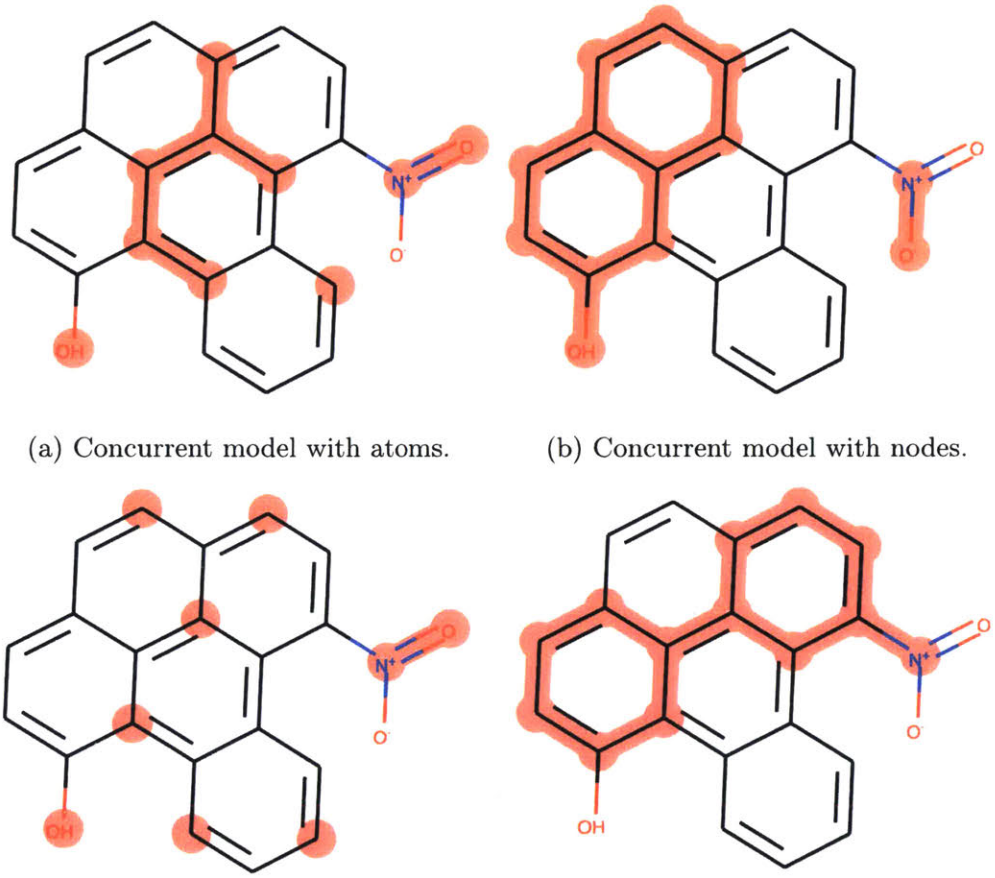


(c) RL model with atoms.



(d) RL model with nodes.

Figure 5-2: Example rationale selections for a test molecule in the solubility dataset.



(a) Concurrent model with atoms.

(b) Concurrent model with nodes.

(c) RL model with atoms.

(d) RL model with nodes.

Figure 5-3: Example rationale selections for a test molecule in the Ames dataset.

Chapter 6

Conclusion

We present a model that treats the problem of selecting rationales from molecules as a reinforcement learning problem. By creating an auxiliary prediction network, we use a learned reward structure to facilitate the selection of atoms in the molecule that are irrelevant to the prediction task, thereby highlighting the important substructures of the molecules. This is done without substantial loss in predictive accuracy. In this work, we explore the applicability of rationales in the chemistry domain. Through various experiments on the solubility and Ames, we demonstrate that our model successfully learns to select reasonable substructures in an unsupervised manner, requiring the same data as an end-to-end prediction task, which is relevant to many applications including drug design and discovery. Molecules are far more complicated to reason about as compared to images or text due to complex chemical theories and a lack of definitive ground truth rationale labels. As deep learning algorithms continue to permeate the chemistry domain, it will be ever more important to consider the interpretability of such models.

6.1 Future Directions

While this work presents a good foray into rationale exploration for the chemical space, there are improvements and future directions that can be taken.

- **Abstractive Rationale** In this work, we used extractive rationale, meaning

that the rationale came exactly from the input atoms themselves. However, often this may not be the most useful kind of explanation for the human. It could be more helpful and interesting to create rationales for chemical problems beyond simple highlighting of atoms in the molecule.

- **Reward Shaping** The rewards used in this work is still relatively simple, only comparing each new prediction with the previous and optimal prediction. There could be better ways to formulate the reward to propel the agent to select more smartly.

Bibliography

[1] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

[2] Robert E. Babine and Steven L. Bender. Molecular recognition of proteinâĹŠligand complexes:âĂĽ applications to drug design. *Chemical Reviews*, 97(5): 1359–1472, 1997. doi: 10.1021/cr960370z. URL `http://dx.doi.org/10.1021/cr960370z`. PMID: 11851455.

[3] Kan Chen, Jiang Wang, Liang-Chieh Chen, Haoyuan Gao, Wei Xu, and Ram Nevatia. Abc-cnn: An attention based convolutional neural network for visual question answering. *arXiv preprint arXiv:1511.05960*, 2015.

[4] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

[5] John S. Delaney. Esol:âĂĽ estimating aqueous solubility directly from molecular structure. *Journal of Chemical Information and Computer Sciences*, 44(3): 1000–1005, 2004. doi: 10.1021/ci034243x. URL `https://doi.org/10.1021/ci034243x`. PMID: 15154768.

[6] Joseph L. Durant, Burton A. Leland, Douglas R. Henry, and James G. Nourse. Reoptimization of mdl keys for use in drug discovery. *Journal of Chemical Information and Computer Sciences*, 42(6):1273–1280, 2002. doi: 10.1021/ci010132r. URL `http://dx.doi.org/10.1021/ci010132r`. PMID: 12444722.

[7] David K. Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. *CoRR*, abs/1509.09292, 2015. URL `http://arxiv.org/abs/1509.09292`.

[8] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *CoRR*, abs/1704.01212, 2017. URL `http://arxiv.org/abs/1704.01212`.

[9] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *CoRR*, abs/1702.08165, 2017. URL `http://arxiv.org/abs/1702.08165`.

[10] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[11] Wengong Jin, Connor W Coley, Regina Barzilay, and Tommi Jaakkola. Predicting organic reaction outcomes with weisfeiler-lehman network. *arXiv preprint arXiv:1709.04555*, 2017.

[12] Andrej Karpathy, Justin Johnson, and Fei-Fei Li. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078, 2015. URL `http://arxiv.org/abs/1506.02078`.

[13] Theodora Katsila, Georgios A Spyroulias, George P Patrinos, and Minos-Timotheos Matsoukas. Computational approaches in target identification and drug discovery. *Computational and structural biotechnology journal*, 14:177–184, 2016.

[14] Jeroen Kazius, Ross McGuire, and Roberta Bursi. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of medicinal chemistry*, 48(1): 312–320, 2005.

[15] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

[16] Been Kim, Julie A Shah, and Finale Doshi-Velez. Mind the gap: A generative approach to interpretable feature selection and extraction. In *Advances in Neural Information Processing Systems*, pages 2260–2268, 2015.

[17] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730*, 2017.

[18] Greg Landrum. Rdkit: Open-source cheminformatics. URL `http://www.rdkit.org`.

[19] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 253–256, 2010.

[20] Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. Rationalizing neural predictions. *CoRR*, abs/1606.04155, 2016. URL `http://arxiv.org/abs/1606.04155`.

[21] Benjamin Letham, Cynthia Rudin, Tyler H McCormick, David Madigan, et al. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3):1350–1371, 2015.

[22] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*, 2015.

[23] Yvonne C Martin, Mark G Bures, Elizabeth A Danaher, Jerry DeLazzer, Isabella Lico, and Patricia A Pavlik. A fast new approach to pharmacophore mapping and its application to dopaminergic and benzodiazepine agonists. *Journal of computer-aided molecular design*, 7(1):83–102, 1993.

[24] Andre Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pages 1614–1623, 2016.

[25] Andreas Mayr, GÃijnter Klambauer, Thomas Unterthiner, and Sepp Hochreiter. Deeptox: Toxicity prediction using deep learning. *Frontiers in Environmental Science*, 3:80, 2016. ISSN 2296-665X. doi: 10.3389/fenvs.2015.00080. URL `https://www.frontiersin.org/article/10.3389/fenvs.2015.00080`.

[26] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016.

[27] Huijuan Xu and Kate Saenko. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *European Conference on Computer Vision*, pages 451–466. Springer, 2016.

[28] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 21–29, 2016.

[29] Chun Wei Yap. Padel-descriptor: An open source software to calculate molecular descriptors and fingerprints. *Journal of Computational Chemistry*, 32(7):1466–1474, 2011. ISSN 1096-987X. doi: 10.1002/jcc.21707. URL `http://dx.doi.org/10.1002/jcc.21707`.