Molecular Substructure-selecting Linemar Gramression

Bill McDowell
Computer Science Department
Stanford University
450 Serra Mall
Stanford, CA 94305
`wmcdowel@stanford.edu`

Stephan Eismann
Applied Physics Department
Stanford University
450 Serra Mall
Stanford, CA 94305
`seismann@stanford.edu`

Abstract

Many machine learning tasks require models to determine the relationships between logically complex features of an object. The space of such features may be too large to explore exhaustively, and so models must use an efficient method to navigate the space in search of features relevant to the task. For example, tasks in computational chemistry require models to learn the relationships between a molecule's atomic structure and its quantum mechanical properties. The structure of a molecule determines an enormous space of logically complex, substructural features for a model to consider in predicting molecular properties. In this paper, we consider models that learn to perform such tasks over large feature spaces, defined inductively through "feature grammars". A "feature grammar" consists of a set of rules for constructing increasingly complex features from simpler features, and it implicitly defines a discrete space of features where two features are neighbors if one can be constructed from another by the application of some deterministic feature transformation rules. Our learning models use such rules to navigate a feature space in search of relevant features under the assumption that the grammatical relationship between two features in the space encodes some information about their relevance for a task. We apply the models to predict a molecule's properties from substructural features of its molecular graph[1]. Our results suggest that the models are able to improve prediction accuracy while discovering relevant, interpretable, and physically intuitive features.

1 Introduction

In supervised machine learning tasks, we are given a set of training data $D = \{(\mathbf{x}^1, l^1), (\mathbf{x}^2, l^2), \dots, (\mathbf{x}^n, l^n)\}$ where each data instance $\mathbf{x}^d \in X$ has a target label $l^d \in L$, and we wish to learn a function $c : X \to L$ [8]. Many learning algorithms assume that data instances can be represented by some finite feature vector in an m-dimensional space given by a function $\mathbf{f} : X \to \mathbb{R}^m$. The particular choice of featurization $\mathbf{f}$ is task specific, and is usually chosen according to prior knowledge of the relevant domain. Unfortunately, for many tasks, our prior knowledge of the domain is insufficient for precisely specifying the relevant set of features to include in $\mathbf{f}$, and so feature selection algorithms might be necessary to determine a relevant set of features automatically [2]. These algorithms typically assume that we begin with some large finite set containing both relevant and irrelevant features, and they select a smaller subset from these according their usefulness within a learning model.

[1]Our implementation for this project can be found at `https://github.com/forkunited/grampy`

In this paper, we propose "feature grammar" models that weaken the requirement of feature selection algorithms that we must start with a finite set of features from which to select those that are relevant. Rather than starting with a finite feature set from which to select, these models will assume that we have access to a large–possibly infinite–set of features describable by some formal language. Furthermore, these models will assume that the features describable in this language can be generated by some grammar, and that we have some heuristic rules for navigating the space of feature derivations by this grammar in search of relevant features. Intuitively, these rules should encode assumptions that if a feature of some form is relevant, then a feature of another grammatically related form is also likely to be relevant. This assumption will allow a feature selection model to traverse the grammatical feature space in search of relevant features.

As a test of the new models, we consider their application to tasks from computational chemistry–predicting the quantum mechanical properties from molecular structure. For these tasks, the models navigate a feature space consisting of features that count tree substructures within graphs over molecules' atoms and bonds. For example, one feature in this space counts the number of times a $\mathrm{C} - \mathrm{C} = \mathrm{O}$ chain occurs within a molecule (two carbons single bonded, with one double bonded to an oxygen). Figure 2 shows some additional examples of tree substructure features in the feature space. Our feature grammar models represent these tree substructures as first-order logic existential conjunctive formulas over unary predicates representing element types (e.g. carbon), and binary predicates representing bond types (e.g. double). The models navigate the space of tree structures using a heuristic rule that joins two smaller tree structure formulas on a single, unary, chemical element predicate. Specifically, for example, this rule allows for the feature $\mathrm{C} - \mathrm{C} = \mathrm{O}$ to be constructed from the two substructures $\mathrm{C} = \mathrm{O}$ and $\mathrm{C} - \mathrm{C}$. If the learning model discovers that $\mathrm{C} = \mathrm{O}$ and $\mathrm{C} - \mathrm{C}$ are both relevant to predicting a molecular property, then it might also consider selecting $\mathrm{C} - \mathrm{C} = \mathrm{O}$. The model considers $\mathrm{C} - \mathrm{C} = \mathrm{O}$ under the assumption that the grammatical relationship between $\mathrm{C} - \mathrm{C} = \mathrm{O}$ and the other two features encodes a relationship between their relevances to the prediction task. Using the grammatical heuristic, our models are able to efficiently explore the large space of molecular (tree) substructures in search of relevant features.

The models considered in this paper were built on previous work on constructing a "feature grammar" version of logistic regression. In the next section, we describe how we have adapted models from this work to apply to the task of predicting quantum mechanical properties from molecular structure. After we have described the models and feature space, we give a brief review of related learning models and past work in computational chemistry. Following the review, we describe the results of applying our model to learn to predict molecular heat capacities and electronic spatial extents. Lastly, we conclude with some general discussion of the results and possible directions for future work.

2 A Model and a Features Space

2.1 Linemar Gramression: A Linear Regression Feature Grammar Model

In past work, we developed feature-selecting "logistmar gramression"—a variation on logistic regression that learns to perform a classification task given an initial seed feature vocabulary and a set of heuristic rules for expanding this vocabulary with other grammatically related features.[2]For computational chemistry tasks discussed in this paper, we modify the "logistmar gramression" classifier learning algorithm to construct a linear regression variation–"linemar gramression". Linemar gramression applies the same gradient descent weight updates as l_1-regularized linear regression, but simultaneously uses heuristic rules to navigate a feature space in search of relevant features. More specifically, as linemar gramression updates its feature weights during gradient descent, it applies production rules to features whose weight magnitudes go above a certain threshold. The rules construct new features that are grammatically related to the input features, and these new features are added to the feature vocabulary as the gradient descent updates continue. If the heuristics guide the model to all relevant parts of the space prior to irrelevant parts of the space, then linemar gramression will produce (roughly) the same classifier as l_1-regularized linear regression over some finite feature vocabulary.[3]

[2]See `https://github.com/forkunited/grampy/blob/master/src/main/resources/papers/feature-selection.pdf` for details.

[3]This is at least intuitively approximately true, but there are some caveats, and we haven't proven it yet.

To construct the linemar gramression algorithm, we modify a feature-selecting version of traditional linear regression. In traditional linear regression [11], we are given labeled data instances $(\mathbf{x}, l) \in X \times \mathbb{R}$ with input vectors given by a feature function $\mathbf{f} : X \to \mathbb{R}^m$. Furthermore, we assume a label l is normally distributed, conditioned on input $\mathbf{x}$, for some parameters $\mathbf{w}, \sigma$ according to $(l \mid x; \mathbf{w}, \sigma, \mathbf{f}) \sim \mathcal{N}(\mathbf{w}^\top \mathbf{f}(\mathbf{x}), \sigma)$. Given these assumptions, linear regression can find the maximum-likelihood estimate (MLE) by running gradient descent to choose the $\mathbf{w}$ that maximizes the likelihood of the training data. For the present work, we consider variations on linear regression that approximate the MLE using *stochastic* gradient descent (SGD)—which only computes gradients with respect to single or small batches of training examples rather than the full data set from the log-likelihood objective. Furthermore, to construct linemar gramression, we modify the feature-selecting version of linear regression that has an l_1-regularization term added to the MLE objective [12]. However, using SGD with the l_1-regularizer is complicated by the fact that the l_1-regularization term in the objective function is non-differentiable, and naively applying SGD subgradient weight updates will not yield the sparse parameter estimates we desire for feature selection. To deal with this issue, we take the approach given in [18] of smoothing out the effects of fluctuating gradient estimates by considering the cumulative l_1 penalty over multiple updates, and clipping the resulting weight updates to zero when a weight changes sign.

Algorithm 1 shows the linemar gramression model—a "feature grammar" version of l_1-regularized linear regression that relies on the modified SGD adapted from [18]. The algorithm takes as input the following parameters:

- D: A labeled training data set
- $\mathbf{f}_0$: An initial seed vocabulary of feature functions
- H: A set of heuristic rules for navigating the feature space to extend $\mathbf{f}_0$
- t: A weight (relevance) threshold above which to apply rules in H to features to construct new features
- K: Number of training iterations
- C: Weight of the l_1 regularization term in the objective
- η_0: Initial learning rate
- α: Learning rate exponential decay parameter

As mentioned above, the algorithm uses SGD to optimize the l_1-regularized linear regression objective over data D, while applying rules from H to features whose weights go above threshold t. The heuristic rules in H could be applied after every iteration, but for the sake of efficiency, they are only applied by "ApplyRules" once after each pass over the data. The resulting features are appended to $\mathbf{f}$ (where $\mathbf{f}$ initially only contains the seed features $\mathbf{f}_0$). The weight vector $\mathbf{w}$ and cumulative l_1 penalty vector $\mathbf{q}$ are extended to contain dimensions corresponding to the new features.

As in [18], the algorithm exponentially decays the learning rate η according to parmeters η_0 and α, and applies stochastic gradient weight updates estimated by single examples from D using the "UpdateWeights" function. Over successive updates, the u variable keeps track of the total l_1 penalty that any weight could have received so far, and $\mathbf{q}$ keeps track of the amount of l_1 penalty that weights have actually received. To ensure that the stochastic updates correctly optimize the objective, D is shuffled after each pass.

2.2 A Molecular Substructure Feature Space

The linemar gramression model described above requires a task-specific feature space, seed features $\mathbf{f}_0$, and set of heuristics H for navigating the space. In this paper, we are concerned with computational chemistry tasks of predicting the properties of molecules from their bond structures. Prior work in this domain has focused on the task of predicting molecule atomization energies based on Coulomb matrices computed from atomic distances and charges [16]. Instead of using Coulomb matrix features, we consider graph-based (tree) structure features based on bond types, as they allow us to more easily define a language of human-interpretable features for our feature grammar models to navigate. They also allow for the definition of a simple heuristic rule that takes two tree-structure features, and joins them at a single atom node to form a larger tree structure.

More formally, for these molecular property prediction tasks, each data instance $\mathbf{x} = (A_\mathbf{x}, B_\mathbf{x}, P_\mathbf{x})$ is a molecule consisting of atoms $A_\mathbf{x}$, bonds $B_\mathbf{x}$, and properties $P_\mathbf{x}$. Each feature $f^{c_1,\dots,c_k} : X \to \mathbb{R}$

Algorithm 1 l_1-regularized Linemar Gramression SGD

```
1: function TRAIN-LG-$l_1$($D$, $\mathbf{f}_0$, $H$, $t$, $K$, $C$, $\eta_0$, $\alpha$)
2:     $u, \mathbf{w}, \mathbf{q} \leftarrow 0$
3:     $\mathbf{f} \leftarrow \mathbf{f}_0$
4:     for $k = 0$ to $K$ do
5:         $j \leftarrow k \mod |D|$
6:         if $j = 0$ then
7:             Shuffle $D$
8:             $\mathbf{f}, \mathbf{w}, \mathbf{q} \leftarrow$ APPLYRULES($D, \mathbf{f}, H, t, \mathbf{w}, \mathbf{q}$)
9:         $\eta \leftarrow \eta_0 \alpha^{k/N}$
10:        $u \leftarrow u + \eta \frac{C}{|D|}$
11:        $\mathbf{w}, \mathbf{q} \leftarrow$ UPDATEWEIGHTS($D, j, \mathbf{f}, \eta, u, \mathbf{w}, \mathbf{q}$)
12: return $w, F$
13:
14: function APPLYRULES($D$, $\mathbf{f}$, $H$, $t$, $\mathbf{w}$, $\mathbf{q}$)
15:     $\mathbf{f}_{>t} \leftarrow \{f_i \mid |w_i| > t\}$
16:     $\mathbf{f}_h \leftarrow \bigcup_{h \in H} h(\mathbf{f}_{>t,i})$
17:     $\mathbf{f} \leftarrow (\mathbf{f}, \mathbf{f}_h)$
18:     $\mathbf{w} \leftarrow \mathbf{w}$ extended with $|\mathbf{f}_h|$ zeros
19:     $\mathbf{q} \leftarrow \mathbf{q}$ extended with $|\mathbf{f}_h|$ zeros
20: return $\mathbf{f}$,$\mathbf{w}$, $\mathbf{q}$
21:
22: function UPDATEWEIGHTS($D$, $j$, $\mathbf{f}$, $\eta$, $u$, $\mathbf{w}$, $\mathbf{q}$)
23:     $(\mathbf{x}^j, l^j) \leftarrow D[j]$
24:     $\mathbf{g} \leftarrow \mathbf{x}^j(l^j - \mathbf{w}^\top \mathbf{f}(\mathbf{x}^j))$
25:     for $i = 1$ to $|\mathbf{f}|$ do
26:         $w_i \leftarrow w_i + \eta g_i$
27:         $w_i, q_i \leftarrow$ APPLYPENALTY($w_i, q_i, u$)
28: return $\mathbf{w}$,$\mathbf{q}$
29:
30: function APPLYPENALTY($w_i$, $q_i$, $u$)
31:     $z \leftarrow w_i$
32:     if $w_i > 0$ then
33:         $w_i \leftarrow \max(0, w_i - (u + q_i))$
34:     else if $w_i < 0$ then
35:         $w_i \leftarrow \min(0, w_i + (u - q_i))$
36:     $q_i \leftarrow q_i + (w_i - z)$
37: return $w_i$,$q_i$
```

in our tree-structural feature space is computed as a weighted existential conjunction (in first-order logic) that counts the number of times some tree structure (defined by the conjuncts) occurs within a molecule, and returns a value proportional to that count. Specifically:

$$f^{c_1,\ldots,c_k}(\mathbf{x}) = \frac{\mid \mathbf{Sat}(\bigwedge_{i=1}^{k} c_i, \mathbf{x}) \mid - \mu_{f^{c_1,\ldots,c_k}}}{\sigma_{f^{c_1,\ldots,c_k}}}$$

Each conjunct $c_i \in E \cup B$ where $E = \{\mathbf{C}(a), \mathbf{H}(a), \mathbf{N}(a), \mathbf{O}(a), \mathbf{F}(a) \mid a \text{ is a variable}\}$ is the set of atomic element predicates, and $B = \{\mathbf{SINGLE}(a_1, a_2), \mathbf{DOUBLE}(a_1, a_2), \ldots \mid a_1, a_2 \text{ are variables}\}$ is the set of bond-type predicates. $\mathbf{Sat}(\bigwedge_{i=1}^{k} c_i, \mathbf{x})$ computes the set of satisfying assignments of atoms from molecules in $\mathbf{x}$ to variables in $\bigwedge_{i=1}^{k} c_i$ where each atom is assigned to at most one variable. $f^{c_1,\ldots,c_k}$ takes the size of this set, subtracts the mean $\mu_{f^{c_1,\ldots,c_k}}$ (of this count computed across all training examples), and divides by the standard deviation $\sigma_{f^{c_1,\ldots,c_k}}$ to compute a normalized count of tree structures within $\mathbf{x}$ of the type defined by $\bigwedge_{i=1}^{k} c_i$. Concretely, for example, $f^{\mathbf{C}(a_1),\mathbf{DOUBLE}(a_1,a_2),\mathbf{O}(a_2)}$ computes a normalized count of the number of times the structure C$=$O occurs within a given molecule.

Given this feature space, our linemar gramression models are seeded with the vocabulary of features representing single edge trees $\mathbf{f}_0 = \{f^{e_1,b,e_2} \mid e_1, e_2 \in E, b \in B\}$. The models are given a single heuristic rule in H for navigating the space. This rule takes two features $f^{e_1^f,\ldots,e_k^f,b_1^f\ldots,b_k^f}$ and $g^{e_1^g,\ldots,e_k^g,b_1^g\ldots,b_k^g}$ with $e_i^f, e_i^g \in E$ and $b_i^f, b_i^g \in B$, and constructs new features that join them at atom predicates e_p^f and e_q^g that represent a common element. The resulting feature represents a larger tree structure with the two input tree structures joined at a single atom. For example, the heuristic would take features representing $\mathrm{C{=}O}$ and $\mathrm{C{-}O}$, and construct $\mathrm{C{=}O{-}C}$ and $\mathrm{O{=}C{-}O}$.

3 Related Work

Several existing machine learning paradigms seem intuitively related to the idea of grammatically constructing and selecting features and compositions of features within a supervised learning model. First, deep neural networks models that solve supervised learning tasks can be thought of as constructing complex features from simpler features in the input vector [7]. This is similar to the feature construction carried out by a feature grammar model. Linemar gramression can use the heuristics in H to create logically complex features based on simpler features in $\mathbf{f}_0$, and these features might reflect the features learned by a deep neural network. Second, feature grammar models also bear resemblances to other learning paradigms like decision trees [14] and inductive logic programming (ILP) [10]. These paradigms impose some logical structure over the input features to approximate a function which computes the output class or label. Most notably, "structural logistic regression" uses ILP methods to construct logically complex features for selection within step-wise logistic regression model [13]. This method is very similar to ours. However, our method differs in that we do not require a separate ILP algorithm for the feature candidate generation, and it is applicable to feature spaces that are not defined through first-order logic.

Some of the traditional machine learning models mentioned above have been applied to computational chemistry tasks of predicting the quantum mechanical properties from molecular structure. Most notably, prior work has applied neural networks and other traditional machine learning methods to predicting atomization energies [16, 9, 3, 17]. In general, this work computes "Coulomb matrices" based on atomic distances and charges within molecules, and transforms the matrices into feature vectors as inputs to the learning models. The main progress in these papers has been in finding the appropriate methods for transforming the Coulomb matrices into vectors that are useful for learning models in predicting atomization energies. In our experiments described below, we compare our proposed models to the neural network architecture from [9] that uses a "sorted Coulomb matrix" as a feature vector. The sorted Coulomb matrix is not the state-of-the-art approach, but we chose it as a simple baseline model for feasible comparison under our time constraints.

4 Experiments

We experiment with models that learn to predict heat capacity (C_v) and electronic spatial extent (R^2) on a subset of the data set of 6,095 constitutional isomers of $C_7H_{10}O_2$ from [15]. We decided to use a sample from the 6,095 constitutional isomers rather than from the larger, more diverse set of 194,000 molecules for convenience under our time constraints. We diverge from previous work by predicting R^2 and C_v rather than atomization energies as atomization energies are expected to differ little across constitutional isomers. In accordance, none of our models were able to predict them with lower error than a simple mean baseline predictor. For these initial experiments, we took a small training sample of 1,000 training molecules and 500 test molecules. We did not have time to run hyper-parameter searches for our models, and so we did not require a separate development set.

We compare our linemar gramression feature grammar model ($\mathbf{LG}$) to the 2-layer neural network architecture ($\mathbf{NN}_{Mont}$) used to predict atomization energies in [9], standard l_1 regularized linear regression ($\mathbf{LR}$), and a mean baseline predictor for both the C_v and R^2 prediction tasks. We use TensorFlow [1] to reproduce $\mathbf{NN}_{Mont}$ with a first layer of 400 ReLUs and a second layer of 100 ReLUs. We use Xavier weight initialization, and train the network with the Adam optimizer (step-size $\alpha = 0.01$) [5]. Our $\mathbf{LR}$ implementation uses Algorithm 1 for training, but without applying the heuristic rules given to $\mathbf{LG}$. Both $\mathbf{LG}$ and $\mathbf{LR}$ use $C = 0.01$ for their l_1 regularization term, an initial learning rate $\eta_0 = 0.001$, and a learning rate decay $\alpha = 0.8$. In $\mathbf{LG}$, we use a heuristic rule threshold of $t = 0.15$ to predict C_v and $t = 20$ to predict R^2.[4]

[4]The values for t were chosen so that $\mathbf{LG}$ would generate a few hundred new features.

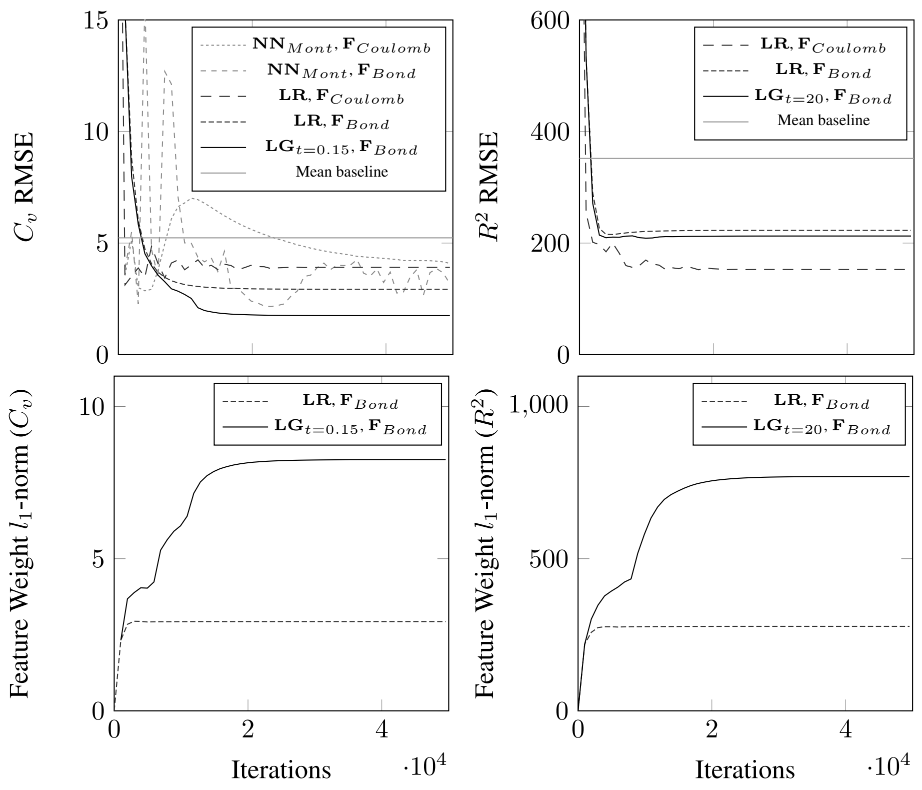

Figure 1: Results of training models for 50,000 iterations on the constitutional isomer test to predict heat capacities (C_v) and electronic spatial extents (R^2). The top panels show the test set RMSE for each prediction task over successive iterations, and the bottom panels show the corresponding l_1-norms of the feature weights in linear regression and linemar gramression.

For input feature sets, we train $\mathbf{LR}$ and the neural network using the initial seed vocabulary of bond features ($\mathbf{F}_{bond}$) for $\mathbf{LG}$, described in Section 2.2. We also train both of these baseline models using the sorted-Coulomb matrix features ($\mathbf{F}_{coulomb}$) from [9].

4.1 Results

We train each model for 50,000 iterations, and inspect the test set RMSE.[5] As shown in the top-left plot of Figure 1, $\mathbf{NN}_{Mont}$ tends to perform no better than our $\mathbf{LR}$ baseline in predicting C_v. Also, the neural training failed entirely on R^2 (with greater RMSE than the mean baseline predictor), and so we left the neural net performance out of the R^2 RMSE plot. We hypothesize that the relatively poor neural network performance is due to our small sample size and improper hyper-parameter settings.

The top plots of Figure 1 also show that the Coulomb matrix features ($\mathbf{F}_{Coulomb}$) lead to better performance for $\mathbf{LR}$ than the bond-type features ($\mathbf{F}_{Bond}$) in predicting R^2, but $\mathbf{F}_{Bond}$ features tend to perform better in predicting C_v. We speculate that this may be due to the encoding of inter-atom distances in $\mathbf{F}_{Coulomb}$ that is left out of $\mathbf{F}_{Bond}$, which might be especially relevant to predicting R^2.

Most interestingly, $\mathbf{LG}$ with the $\mathbf{F}_{Bond}$ seed features leads to lower RMSE than $\mathbf{LR}$ with $\mathbf{F}_{Bond}$ in predicting both R^2 and C_v—especially C_v. This suggests that $\mathbf{LG}$ was able to use the heuristic rules to discover features in the tree-structure feature space that were relevant to both tasks. Further evidence for this is shown in the bottom plots of Figure 1, which suggest that steps in l_1-norm of the feature weights for $\mathbf{LG}$ roughly correspond to dips in the $\mathbf{LG}$ RMSE plots. These steps in the graph correspond to iterations where the model added new features to the vocabulary.

[5]The results in this section are preliminary, and shouldn't be taken too seriously. We did not have enough time to run the appropriate hyper-parameter searches, and the training sample is rather small.

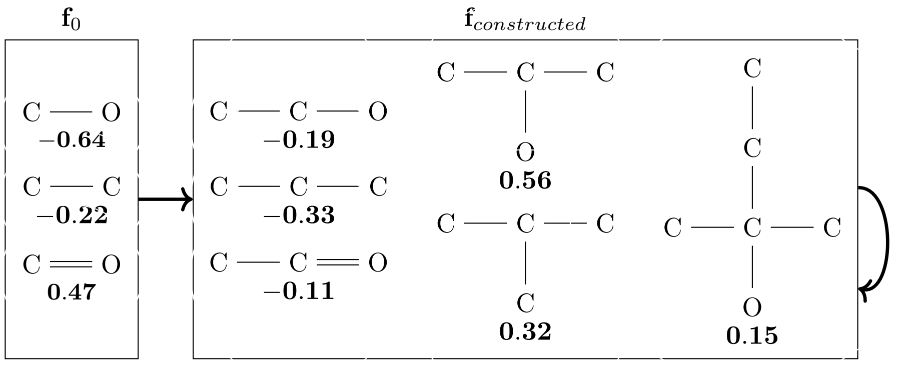

Figure 2: Some examples of features that received high-magnitude weights in the linemar gramression model trained to predict heat capacity (C_v). The learned weights are shown below each feature structure. Features in the left box were in the seed vocabulary $\mathbf{f}_0$, and features in the right box were constructed through successive applications of the heuristic rules.

When training to predict C_v, **LG** assigns non-zero weights to features representing tree structures with up to 7 atoms. Some of the most highly weighted features that the model discovers are shown in Figure 2.

5 Discussion and Future Work

The preliminary results described above suggest that linemar gramression is able to discover useful structural features for predicting molecule heat capacities and electronic spatial extents. Furthermore, both the seed vocabulary and discovered features are human-interpretable, and they might align in theory with group-contribution methods developed in organic chemistry over the last century [6]. Group-contribution methods rely on the *additive principle* which means that a compound can be divided into fragments (e.g. bonds, atoms or groups of atoms) with each having a partial value. The molecular property value is then calculated as the sum of these partial values. Group-contribution methods for heat capacity have e.g. been developed by Joback and Reid [4] with the individual contributions determined empirically based on experimental data. Unfortunately, we cannot currently compare the feature weights in Figure 2 to these empirically estimated coefficients from group contribution methods because our features were constructed to ignore hydrogen atoms for efficiency, and the group-contribution values are computed with hydrogen atoms included. In the future, we hope to improve the efficiency of our feature computation, so that the hydrogen atoms can be included in the structural features, and we can compare the constructed features' weights to their corresponding values from group-contribution methods.

Although the initial results are promising, we still hope to improve their robustness by training on a larger set of data, and running the appropriate hyper-parameter searches. In addition, the current implementation of linemar gramression is limited by the efficiency of computing the tree-structural features, and so we hope to improve the runtime using dynamic programming techniques to reuse the tree-structure counts across grammatically related features. We also hope to experiment with additional feature grammars (that include information from the Coulomb matrices), and train the models to predict atomization energies a more divers data set, following prior work.

Acknowledgments

Tom Mitchell at CMU suggested the "feature grammar" idea to the first author when he was working for him a few years ago, and so he is responsible for inspiring the overarching theme of this work. We also thank Amarish Kulkarni for his advice about working with chemistry data, and Leon Bergen for his suggestions for future work on feature grammar models.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[3] Katja Hansen, Grégoire Montavon, Franziska Biegler, Siamac Fazli, Matthias Rupp, Matthias Scheffler, O Anatole Von Lilienfeld, Alexandre Tkatchenko, and Klaus-Robert Müller. Assessment and validation of machine learning methods for predicting molecular atomization energies. *Journal of Chemical Theory and Computation*, 9(8):3404–3419, 2013.

[4] KG Joback and RC Reid. Estimation of pure-component properties from group-contributions. *Chemical Engineering Communications*, 57, 1987.

[5] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] Alena Randova Kolska, Zdenka and Milan Zabransky. Group contribution methods for estimation of selected physico-chemical properties of organic compounds. 2012.

[7] Yann LeCun and M Ranzato. Deep learning tutorial. In *Tutorials in International Conference on Machine Learning (ICML'13)*. Citeseer, 2013.

[8] Tom M Mitchell. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45(37):870–877, 1997.

[9] Grégoire Montavon, Katja Hansen, Siamac Fazli, Matthias Rupp, Franziska Biegler, Andreas Ziehe, Alexandre Tkatchenko, Anatole V Lilienfeld, and Klaus-Robert Müller. Learning invariant representations of molecules for atomization energy prediction. In *Advances in Neural Information Processing Systems*, pages 440–448, 2012.

[10] Stephen Muggleton, Ramon Otero, and Alireza Tamaddoni-Nezhad. *Inductive logic programming*, volume 38. Springer, 1992.

[11] Andrew Ng. Cs229 lecture notes. *CS229 Lecture notes*, 1(1):1–3, 2000.

[12] Andrew Y Ng. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM, 2004.

[13] Alexandrin Popescul, Lyle H Ungar, Steve Lawrence, and David M Pennock. Towards structural logistic regression: Combining relational and statistical learning. 2002.

[14] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[15] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1, 2014.

[16] Matthias Rupp. Machine learning for quantum mechanics in a nutshell. *International Journal of Quantum Chemistry*, 115(16):1058–1073, 2015.

[17] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O Anatole Von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical review letters*, 108(5):058301, 2012.

[18] Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 477–485. Association for Computational Linguistics, 2009.