Augmenting Contextual Explanation Networks with Logic Rules

Angel C. Hernandez (angelh) [1] **Yao Chong Lim (yaochonl)** [1] **Ziyang Wang (ziyangw)** [1]
Wenchao Du (wenchaod) [1]

Abstract

We study the problem of augmenting contextual explanation networks (CENs) with logic rules. Compared to existing interpretable models, logic rules can potentially provide even more interpretable decisions, since they explicitly present the features used to make the decision. However, our experiments show that the effectiveness of logic rule explanations depend strongly on the informativeness of the interpretable features used to construct the logic rules, as efficiency concerns limit the size of the logic rules to be around 3. Nevertheless, qualitative analysis of our results suggest that our proposed method gives more interpretable results, making it useful to fields where white-box models are needed, such as medicine.

1. Introduction

Machine learning models have obtained better and better performance on many challenging tasks, usually at the cost of interpretability. This makes it difficult to analyze such models and determine their general robustness, reliability, and fairness. This also discourages the use of these models in real-world applications, where such qualities are desired in the predictive models used.

Contextual Explanation Networks (CEN) (Al-Shedivat et al., 2017) aimed to tackle this problem using a framework for models that learn to both predict and explain. In particular, the model learns to give an interpretable prediction using an interpretable model (traditionally linear models), with the help of a contextual model (typically a deep neural network). The current work on CEN only describes learning linear models for explanations on the interpretable attributes, but other prediction models could be used that still preserve interpretability.

As such, we will utilize logic rules (Okajima & Sadamasa, 2018) as our explanation model, where each prediction is accompanied with a set of logical explanations. Furthermore, different deep networks are explored as the contextual model and having these two models work together to learn highly accurate and interpretable predictions is the goal of the augmented CEN developed in this paper.

[1]Carnegie Mellon University, Pittsburgh, PA 15213, USA.

10708 Class Project, Spring, 2019.

2. Literature Review

In recent years great strides have been made in developing model interpretability methods and the paradigm breaks out into two approaches: *post-hoc* analysis and *embedded* interpretability.

2.1. Post-hoc Analysis

LIME (Ribeiro et al., 2016) is an algorithm that set out to cultivate trust between practitioners and machine learning models by explaining the predictions of *any* black box classifier or regressor through approximated locally interpretable models. After a black box model has finished training, LIME hones in on a single data instance, x, and attempts to explain the output of the black box model , $f(x)$, by using some explainable model g, e.g., linear classifier. LIME is able to learn the best simple linear model, using interpretable features, which can be used to understand the black box prediction $f(x)$. While LIME has showed great potential, Alvarez-Melis & Jaakkola (2018) showed some instability with LIME, e.g., two similar data instances yielded very different explanations.

2.2. Embedded Interpretability

Hu et al. (2016) aimed to incorporate logical rules into neural prediction. In this paper, they train a “teacher” network alongside a “student” network. The teacher network takes the same inputs as the student, but must additionally satisfy predefined first-order logic rules, that are encoded using soft logic (continuous instead of binary truth values). At each iteration, a new teacher network is created by projecting the student network. The student network is then optimized by both minimizing the loss and trying to imitate the output of the teacher network. In this way, the student network is regularized using the logic rules, but is not constrained to using the rules during test time.

Al-Shedivat et al. (2017) proposed a class of models that learned to predict by generating and leveraging intermediate explanations. CENs generate parameters for intermediate graphical models which are further used for both predictions and interpretations. Instead of doing post-hoc explanations, it learns to predict and explain jointly. CENs have close to state-of-the-art performance. While each prediction made by CEN comes with an explanation, the process of conditioning on the context is still uninterpretable. Moreover, the space of explanations considered in this work assumes the same graphical structure and parameterization for all explanations and uses a simple sparse dictionary constraint, which can be limiting.

Very similar to decision trees, Bayesian Rule Lists (Letham et al., 2015) generate an *if-then* decision list that is used to classify a given example x. The decision list is a set of *antecedents* where a single antecedent may look something like: *if* age > 30 *and* isMale. Maximum likelihood estimation is used to learn the parameters where priors are used to ensure the learned decision list is not long in both total number of antecedents and number of logicals per antecedents. BRLs proved to perform just as good, in some cases better, than traditional machine learning models, while providing highly interpretable results. Their inception was motivated by applications within the medical field where doctors could use field expertise and machine learning to inform decision making.

3. Relevant Work

3.1. Contextual Explanation Networks

We define a dataset as $\mathcal{D} = \{\boldsymbol{c}^{(i)}, \boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$ where $\boldsymbol{c}^{(i)}$ are context features, $\boldsymbol{x}^{(i)}$ are interpretable features, $y^{(i)}$ is the target and N is the number of training examples. The difference between $\boldsymbol{c}$ and $\boldsymbol{x}$ is that $\boldsymbol{c}$ are features that tends to come with low interpretability (image data, video data, etc.), while $\boldsymbol{x}$ are highly interpretable features (categorical data, tabular data, super pixels etc.). The goal of CENs is to learn a model $\mathbb{P}_{\mathbf{w}}(\mathbf{y} \mid \mathbf{x}, \mathbf{c})$ parametrized by $\boldsymbol{w}$ that can predict $\boldsymbol{y}$ from $\boldsymbol{x}$ and $\boldsymbol{c}$

$$\boldsymbol{y} \sim \mathbb{P}(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}),\ \boldsymbol{\theta} \sim \mathbb{P}_{\mathbf{w}}(\boldsymbol{\theta}|\boldsymbol{c})$$
$$\mathbb{P}_{\mathbf{w}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{c}) = \int \mathbb{P}(\boldsymbol{y} \mid \mathrm{x}, \boldsymbol{\theta}) \mathbb{P}_{\mathbf{w}}(\boldsymbol{\theta} \mid \boldsymbol{c})\, d\boldsymbol{\theta}$$

where $\boldsymbol{\theta}$ can be viewed as some encoding of the context $\boldsymbol{c}$. The $\mathbb{P}_{\mathbf{w}}(\boldsymbol{\theta}|\boldsymbol{c})$ is typically modeled with a delta function and in practice is a deep network with a soft attention mechanism. Furthermore, $\mathbb{P}(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})$ tends to be a linear model (like logistics regression or CRF). The model is trained end-to-end and learns to use both, $\boldsymbol{c}$ and $\boldsymbol{x}$ to generate a prediction, while explanations lie in the linear model and are evaluated on an example-by-example basis.

In the coming sections we build on this framework and incorporate work done by (Okajima & Sadamasa, 2018) with hope of producing more interpretable explanations, while maintaining CENs' current state of the performance.

4. Method

Our final approach was slightly different from what was submitted during the Midway Report. In the Midway Report, our proposed approach involved using a BRL as the explanation model of the CEN (see Appendix A for details). However, further experimentation showed that our approach was intractable. Hence, we decided to simplify the model, and instead use decision rules as our explanation model.

Here, we define decision rules as a single antecedent. For example, "age > 30 *and* isMale" is a possible decision rule explanation that our model can output. Decision rules are made over a limited number of interpretable features. For example, the previous decision rule is made over the features "age" and "gender" (isMale).

To obtain the set of decision rules $\mathcal{A}$, for each datapoint, we first obtain a set of interpretable features for each datapoint (see Section 5.1 for more details). Then, we use the FP-Growth algorithm (Borgelt, 2005) to obtain sets of features that frequently occur together, and are satisfied by a significant proportion of the training data. This gives us $\mathcal{A}$, the set of antecedents from which we can choose from. The exact decision rules generated depend on the dataset in question, and is described in Section 5.1 in further detail.

4.1. CENs with decision rule explanations

We consider the multiclass classification setting.

We first obtain a contextual encoding $\phi_1 = g_{\mathbf{w}_1}(c) \in \mathbb{R}^h$. The structure of $g_{\boldsymbol{w}}$ depends on c. For example, if c is a sequence, then g might be parameterized by a RNN.

Using this encoding, we compute $p(\theta \mid c)$ using dot-product attention with ϕ_1. To do this, we need to encode each logic rule using a vector.

4.1.1. Vector representation of decision rules

The intuitive idea behind our representation of decision rules is that rules that are satisfied by similar datapoints should have similar representations.

Hence, given inputs $\boldsymbol{x} = \{x_i\}_{i=1}^{N}$ and antecedents $\mathcal{A}$, define

$$S_{ij} = \mathbf{1}[x_i \text{ satisfies } a_j]$$

So $S \in \{0,1\}^{N \times L}$ is a binary matrix that indicates whether some input x_j satisfies some antecedent a_j. Then, to compute the representation ψ_j for antecedent a_j, we average the representations ϕ_2 of the inputs that satisfy it. Hence, define:

$$T_{ij} = \frac{1}{\sum_{k=1}^{n} S_{kj}} S_{ij}$$
$$\psi_j = T_{ij}^{\intercal} \Phi_2$$

where $\Phi_2 = g_{\boldsymbol{w}_2}(\boldsymbol{c}) \in \mathbb{R}^{N \times h}$ is a separate encoding of the context. While we could set $\Phi_2 = \Phi_1$ to speed up training, we found that doing so worsened the model's performance significantly.

Finally, to compute $P(a_j \mid c)$, we use dot-product attention between ψ_j and $\phi_1(c)$:

$$P_{\boldsymbol{w}_1, \boldsymbol{w}_2}(a_j \mid x, c) = [T^{\intercal} \Phi_2 \phi_1(c)]_j$$

For a given antecedent a, we use the MLE estimate of $p(y \mid a)$, which is simply the ratio of training examples that satisfy a and have the corresponding label, with some smoothing. Concretely, let $N_{j,\ell}$ be the number of datapoints have label ℓ and satisfy a_j. Then,

$$p(y = \ell \mid a_j) = \frac{N_{j,\ell} + \alpha}{\sum_{\ell' \in Y} N_{j,\ell'} + \alpha}$$

Therefore, given the sections above, for each datapoint (x_i, y_i), we can calculate

$$\mathbb{P}_{\boldsymbol{w}}(y_i \mid x_i, c_i) = \sum_{a \in \mathcal{A}} p(y_i \mid a) p_{\boldsymbol{w}}(a \mid x_i, c_i)$$
$$\mathbb{P}_{\boldsymbol{w}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{c}) = \prod_{i=1}^{n} \mathbb{P}_{\boldsymbol{w}}(y_i \mid x_i, c_i)$$

During training, we minimize this negative log-likelihood by optimizing $\boldsymbol{w}$ using gradient descent.

5. Results

We test our model on a wide variety of tasks, and investigate the performance and interpretability of our model, compared to other baselines.

5.1. Datasets and preprocessing

We experimented with three different datasets: SUPPORT2, MNIST, and IMDB.

5.1.1. SUPPORT2

The SUPPORT2 dataset contains information about intensive care unit patients. Each patient was represented by a set of categorical (income, race, gender etc.) and continuous features (age, body temperature etc.). The frequent itemset mining algorithm we use, FP-Growth, only works with binary/categorical features, so the data was preprocessed to convert the continuous features to categorical features by splitting up the values into quartiles.

Typically, survival analysis models output T, the predicted survival time, which makes it a regression task. Here, we adopt a similar (but slightly different) approach as in Al-Shedivat et al. (2017) and frame survival analysis as a multiclass classification task.

The range of possible times is split into intervals represented by binary variables $(y_1, y_2, y_3, ..., y_m)$, which indicate whether the event (death, in this case) occurs in that particular interval. The final variable y_m indicates whether the event occurred after of the range. So exactly one y_i is true for any input, making it a multiclass classification problem.

As the BRL algorithm performed poorly on classification when the number of classes were large, we tried using larger intervals for the death event: 7 days (SUPPORT2 1-week), 28 days (1-month), 84 days (3-month), and whether the death event occurred at all (binary). We only considered death events in the first 1092 days, like in Al-Shedivat et al. (2017).

5.1.2. MNIST

In (Al-Shedivat et al., 2017), the interpretable features used are either superpixels of the original 28×28 image, or histogram of oriented gradients (HOG) features. For better interpretability, we work with only superpixels. As the efficiency of our model's training depends on the total number of antecedents, we use the values of 7×7 superpixels (each of size 4×4), converted to categorical features, as our rules. However, for the context, we use the original 28×28 image.

5.1.3. IMDB

We also test our model on binary sentiment prediction with the IMDB movie review dataset (Maas et al., 2011). Following the procedure adopted by (Al-Shedivat et al., 2017), we use bag-of-words features as the interpretable features from which we obtain decision rules. We convert the bag-of-words into decision rules by having a feature when a word is present in the review. For the context, we use the original review, truncated for efficiency.

5.2. Preliminary Results

As a baseline for our CEN+BRL model, we first adapted the original BRL implementation, which only considers binary classification, for survival analysis.

To learn the rule lists, the Metropolis-Hastings algorithm

Dataset	Acc	Ratio	# classes
SUPPORT2 (binary)	0.755	0.063	2
SUPPORT2 (3-month)	0.453	0.035	14
SUPPORT2 (1-month)	0.353	0.014	40
SUPPORT2 (1-week)	0.17	0	157

Table 1. Test accuracy of the final BRL chosen for the SUPPORT2 dataset, and the acceptance ratio during Metropolis-Hastings sampling. We report the results for different interval sizes chosen for the survival time for SUPPORT2.

Model	Acc@25	Acc@50	Acc@75	RAE
MLP	51.3	62.6	64.4	0.91
CEN+CRF	91.2	90.9	80.8	0.751
CEN+RCN^3	83.8	98.7	82.8	0.997

Table 2. Performance of models on the SUPPORT2 dataset. CEN+RCN^n indicates that the set of decision rules $\mathcal{A}$ contains rules that cover at most n interpretable features.

Model	Accuracy (%)
CEN+MLP	98.7
CEN+RCN^2	52.9
CEN+RCN^3	60.4

Table 3. Performance of models on the MNIST dataset. CEN+RCN^n indicates that the set of decision rules $\mathcal{A}$ contains rules that cover at most n interpretable features.

Model	Accuracy (%)
CEN+MLP	81.2
CEN+RCN^2	57.8

Table 4. Performance of models on the IMDB dataset. CEN+RCN^n indicates that the set of decision rules $\mathcal{A}$ contains rules that cover at most n interpretable features.

was used. Table 1 shows the acceptance ratio during sampling and classification accuracy of the final decision list on the test set.

As the table shows, the BRL algorithm performs poorly in multiclass classification. Furthermore, the acceptance ratio during sampling is low. Analyzing the log-posteriors of the generated rule lists during sampling, the proportion of sampled rule lists with higher log-posteriors than the current sample greatly decreased as the number of possible antecedents increased, which is probably the cause of the low acceptance ratio.

The time taken for learning also increased greatly as many more sampling iterations were required for convergence as the number of antecedents increased, with models for the 1-month and 1-week datasets taking several hours. The BRL algorithm failed to converge on the 1-week dataset even after 1 million sampling iterations.

5.3. Further results

5.3.1. SUPPORT2

Table 2 shows the results of our model and baseline models on the SUPPORT2 dataset. We report the accuracy of predicting the survival of the patients at each quartile of survival time (Acc@25, Acc@50, Acc@75). Based on the accuracy metrics, our model performs competitively with the baseline CEN+CRF model. The total number of mined antecedents was low compared to the other datasets, partially due to the smaller number of features in the data. When mining for antecedents that cover at most 3 features, we have about 10K antecedents to choose from. Furthermore, qualitative analysis suggests that our model potentially gives more interpretable results than the baseline CRF model (see Section 6).

5.3.2. MNIST

As shown in Table 3, our model performs very poorly compared to the baseline CEN model. Intuitively, this makes sense as due to space limitations, we could only use decision rules containing at most 3 superpixels, which will perform worse than even linear models, which can still use all (super)pixels of the image to make a decision.

This suggests that in order to maximize the performance of the decision rule explanation, significant effort has to be made to ensure that the interpretable features that the decision rules contain must be as informative as possible.

5.3.3. IMDB

Table 4 shows the performance of our model and the baseline CEN+MLP model on the IMDB movie review dataset. Like for MNIST, our model performs significantly worse compared to the baseline, due to the large number of antecedents even when they cover only a few features (for example, after mining for possible antecedents that cover at most 3 features, we had almost 100K possible antecedents to choose from).

Like for MNIST, it seems that smarter feature selection would help our model to perform better.

6. Interpretability Analysis

While our proposed model does not outperform the CEN-CRF model with respect to accuracy, our team argues our model delivers more interpretable explanations. We begin by comparing the results across both models on the Support2 dataset. Figure 1 highlights how well both models were able to predict when a patient would become deceased across the test examples. We notice at time period eight is when both models were able to obtain their best accuracy. We use this time period to compare explanations, where we identify the examples where both models were able to correctly classify the deceased flag at this time period. Of the 1,000 test examples, there were 854 examples where both models were able to correctly classify. In the coming section we will randomly select from these 854 examples and qualitatively analysis each model's explanations.

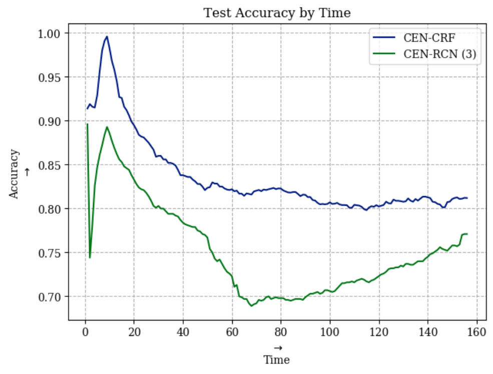

Figure 1. Accuracy by time across CEN-CRF and RCN

6.1. CEN-CRF Interpretability

Because the explanation layer of CEN-CRF is a conditional random field, we can use the weights generated across all 854 examples in the CRF layer as feature magnitude explanations. Feature weights that are the most positive and negative (furthest away from 0) can be interpreted as the features that had the most impact on an example's prediction at this given time period. As a result, we can visualize a heat-map across the 51 weights and some randoms examples to see which features contributed the most to a given prediction.

6.2. CEN-RCN (3) Interpretability

In the CEN-RCN (3) model each prediction is accompanied with 3 logicals, where all logicals are satisfied by a given example's interpretable features. Table 5 highlights a single test example and the explanations that were generated for its prediction at time period 8.

Test Example	Antecedent	Prob Deceased
37	slos $<$ 0.009 0.301 $<$ pafi $<$ 0.366 incomeUnder$11k	0.08

Table 5. Examples of CEN-CRF (3) explanations at time period 8

We can see in Table 5 that **slos, pafi** and **income** were were the explanation features and the range of values that were taken on by this test example. So, we can also generate a heat map for the CEN-RCN predictions where each feature simply receives a count of 1, if it was used in an example's prediction.

6.3. Heat-map Comparison

We randomly selected 20 test examples that were both correctly classified by both models and generated respective heat maps as described in the previous sections. Figure 2 and Figure 3 highlight the respective heat maps. In Figure 2 we notice the magnitude of the weights over the features is extremely sparse. For these 20 examples it is clear almost all features contributed in the final prediction. In the case of Figure 3 we notice each test example has three contributing features (which is the intention of the model) with magnitude being absent. Since CEN-CRF (3) was able to correctly classify these test examples, this proves explanations can take on a much simplistic form while being accurate.

Also, it should be noted that (Al-Shedivat et al., 2017) ran a similar heat-map analysis, but they selected to analyze a finite number of weights and mapped the magnitude overtime across a single training example, while our analysis selects to view all weights at a given time point.

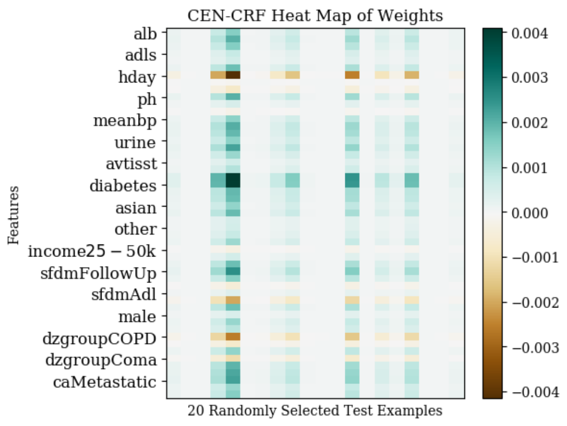

Figure 2. Heat map of weights over 20 randomly selected test examples

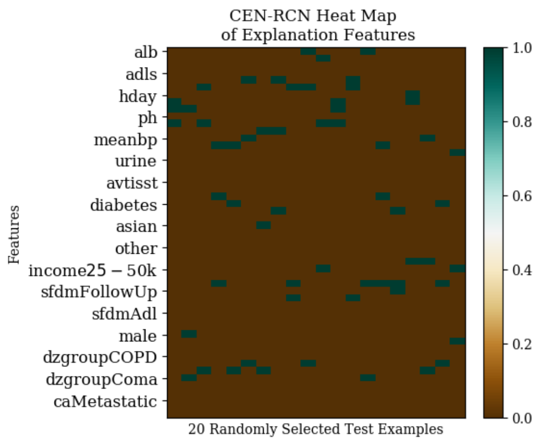

Figure 3. Features used as explanations over 20 randomly selected test examples

7. Conclusion

We used decision rules over interpretable features as the explanations for the CEN framework, and tested it on three separate classification tasks: survival analysis on SUPPORT2, image classification on MNIST, and sentiment prediction on IMDB movie reviews.

Our results suggest that a key requirement for good performance of the model when using decision rules as the explanations is that the interpretable features should be as informative as possible, as the decision rules cannot contain too many features for space and efficiency reasons. This suggests that expert knowledge in crafting the interpretable features would be useful if trying to use decision rule explanations.

While our results were in general worse than the original CEN models, based on our qualitative analysis, we believe that using decision rules as explanations potentially gives much more interpretable decisions, even compared to linear models, especially when there are a large number of interpretable features.

References

Al-Shedivat, M., Dubey, K. A., and Xing, E. P. Contextual Explanation Networks. *CoRR*, abs/1705.10301, 2017.

Alvarez-Melis, D. and Jaakkola, T. S. On the robustness of interpretability methods. *CoRR*, abs/1806.08049, 2018. URL http://arxiv.org/abs/1806.08049.

Borgelt, C. An Implementation of the FP-growth Algorithm. In *Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations*, OSDM '05, pp. 1–5, New York, NY, USA, 2005. ACM. ISBN 1-59593-210-0. doi: 10.1145/1133905.1133907. URL http://doi.acm.org/10.1145/1133905.1133907.

Hu, Z., Ma, X., Liu, Z., Hovy, E., and Xing, E. Harnessing Deep Neural Networks with Logic Rules. Association for Computational Linguistics, 2016. doi: 10.18653/v1/p16-1228.

Letham, B., Rudin, C., McCormick, T. H., and Madigan, D. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *Annals of Applied Statistics*, 9(3):1350–1371, 2015.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P11-1015.

Okajima, Y. and Sadamasa, K. Deep neural networks constrained by decision rules. 2018.

Ribeiro, M. T., Singh, S., and Guestrin, C. Why Should I Trust You? Explaining the Predictions of Any Classifier. *CoRR*, 2016.

In the appendix, we discuss some ideas that have been developed but discarded due to unsuccessful attempts.

A. Neural Bayesian-Rule-List Based on Attention

As in the Bayesian Rule Lists paper, we consider the multiclass classification setting.

Assume the contextual encoder gives us some encoding $\boldsymbol{\phi}$. One way we could adapt Bayesian Rule Lists (BRL) to CEN is to use the encoding to output a decision list d that maximizes the probability of the training data:

$$\begin{aligned}\hat{d} = f_{\boldsymbol{\psi}}(\boldsymbol{\phi}, \mathbf{x}, \mathcal{A}) &= \arg\max_d p(\mathbf{y} \mid d) \\ &= \arg\max_d \prod_{i=1}^{N} p(y_i \mid d)\end{aligned}$$

where $\boldsymbol{\psi}$ are the parameters of f, and

$$\begin{aligned}&\mathbf{x} = \{x_i\}_{i=1}^N \\ &\mathbf{y} = \{y_i\}_{i=1}^N \\ &\mathcal{A} = \{a_1, ..., a_L\} \quad \text{(mined antecedents using FPGrowth)} \\ &d = (a^{(1)}, a^{(2)}, ..., a^{(m)}) \qquad \text{(ordered subset of } \mathcal{A})\end{aligned}$$

We propose using a recurrent neural network (RNN) as $f_{\boldsymbol{\psi}}$. RNNs (and similar models like LSTMs) take in a sequence of inputs one at a time, modifying an internal hidden state, which is used to output a sequence one at a time. For now, we propose simply using $\boldsymbol{\phi}$ at each timestep. We could also use the previously generated antecedent as the input for the next timestep instead.

At each timestep, the RNN outputs a vector $\boldsymbol{e}_t$, which we use to select the next antecedent to output. This requires an encoding for each antecedent. Given $\mathbf{x}$ and $\mathcal{A}$, define

$$S_{ij} = \mathbf{1}[x_i \text{ satisfies } a_j]$$

So $S \in \{0,1\}^{N \times L}$ is a binary matrix that indicates whether some input x_i satisfies some antecedent a_j. Then each column $S_{\cdot j} \in \{0,1\}^L$ could be an encoding for each antecedent.

Given $\boldsymbol{S}$ and $\boldsymbol{e}_t$, we compute a score vector $\boldsymbol{u} \in \mathbb{R}^L$ over all antecedents using some function h (that might have learnable parameters). For example, for a linear regression model,

$$\boldsymbol{u} = h(\boldsymbol{S}, \boldsymbol{e}_t) = \boldsymbol{W}_h \left[\begin{array}{c|c} & -\,\boldsymbol{e}_t\,- \\ \boldsymbol{S} & \vdots \\ & -\,\boldsymbol{e}_t\,- \end{array}\right] + \boldsymbol{b}_h$$

Then we select the best antecedent $a^{(t)} = \arg\max_a u_a$.

Given the output list of antecedents d, we can compute $\log p(\mathbf{y} \mid d)$, which we then optimize using SGD.

In summary:

1. Obtain encoded context $\boldsymbol{\phi} = g_{\mathbf{w}}(\mathbf{c})$.
2. Obtain antecedent set $\mathcal{A}$. We use the FP-Growth frequent itemset mining algorithm (Borgelt, 2005).
3. Compute $\boldsymbol{S}$ from $\mathbf{x}$ and $\mathcal{A}$.
4. Use an RNN-type network and $\boldsymbol{S}$ to output decision list $d = f_{\boldsymbol{\psi}}(\boldsymbol{\phi}, S) \quad (= f_{\boldsymbol{\psi}}(\boldsymbol{\phi}, \mathbf{x}, \mathcal{A}))$.
5. Repeat step 4, optimizing $\log p(\mathbf{y} \mid d)$.

To prevent repeated antecedents in d, we apply a mask to $\boldsymbol{u}$ to zero-out the scores for antecedents that have already been selected.

Given d, we obtain the associated multinomial parameters $(\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_m)$ for each $a^{(m)}$ by following the same procedure as for BRLs: Let $B_{ij} = \mathbf{1}[x_i \text{ satisfies } a^{(j)} \text{ and not } a^{(1)}, \ldots, a^{(j-1)}]$. So $B \in \{0,1\}^{N \times m}$, and each row of B has exactly one value of 1. Then, $\boldsymbol{\theta}_j \sim \text{Dirichlet}(\boldsymbol{\alpha} + \sum_{i=1}^N B_{ij})$, where $\boldsymbol{\alpha}$ is a prior (Letham et al., 2015).

Then, we can derive $\log p(\mathbf{y} \mid d)$:

$$\begin{aligned}p(y_i \mid d, x_i) &= \prod_{j=1}^{m} [p(y_i \mid \boldsymbol{\theta}_j)]^{B_{ij}} \\ &= \prod_{j=1}^{m} \left[\prod_{\ell=1}^{k} (\theta_{j\ell})^{\mathbf{1}[y_i=\ell]}\right]^{B_{ij}} \\ \log p(y_i \mid d, x_i) &= \sum_{j=1}^{m} B_{ij} \sum_{i=1}^{k} \mathbf{1}[y_i = \ell] \log \theta_{j\ell} \\ &= \sum_{j=1}^{m} B_{ij} \log \theta_{j(y_i)} \\ \log p(\mathbf{y} \mid d) &= \sum_{i=1}^{N} \sum_{j=1}^{m} B_{ij} \log \theta_{j(y_i)}\end{aligned}$$

B. Variational Contextual Bayesian-Rule-List Autoencoder

The joint probability distribution of BRL-CEN is

$$p(\boldsymbol{y}, \boldsymbol{\theta}, d \mid \boldsymbol{x}, \boldsymbol{c}, \alpha) = p(\boldsymbol{y} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{x}, d, \alpha) p(d \mid \boldsymbol{c})$$

Propose variational posterior

$$q(\boldsymbol{\theta}, d) = q(\boldsymbol{\theta} \mid \gamma) q(d \mid \delta)$$

The ELBO is given by

$$\begin{aligned}&E_q[\log p(\boldsymbol{y}, \boldsymbol{\theta}, d)] - E_q[\log q(\boldsymbol{\theta}, d)] \\ =&E_q[\log p(\boldsymbol{y} \mid \boldsymbol{\theta}, \boldsymbol{x}, d)] + E_q[\log p(\boldsymbol{\theta} \mid \alpha)] \\ &+ E_q[\log p(d \mid \boldsymbol{c})] - E_q[\log q(\boldsymbol{\theta})] - E_q[\log q(d)]\end{aligned}$$

Calculating the expected log-likelihood involves enumerating all possible lists, which is computationally expensive. Hence, we propose to approximate the expectation term by randomly sampling a subset of the lists at each update step. Moreover, we propose to use an RNN for posterior inference of d. For simplicity, we choose uniform prior on d, although a poisson prior on the length of d is also desirable.

The inference steps are given below:

- At each update step, randomly draw rule lists from antecedents, $\mathcal{D}$
- compute posterior likelihood $q(d)$ for each d in $\mathcal{D}$
- $\gamma \longleftarrow \arg\max E_q[\log p(\boldsymbol{\theta}|\alpha)] - E_q[\log q(\boldsymbol{\theta})] + \sum_{d \in \mathcal{D}} \log p(\boldsymbol{y} \mid \boldsymbol{\theta}, \boldsymbol{x}, d) q(d)$
- $\delta \longleftarrow \arg\max \sum_{d \in \mathcal{D}} \log p(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{x}, d) q(d;\delta) + \log p(d|\boldsymbol{c}) q(d;\delta) - \log q(d;\delta) q(d;\delta)$

Note that δ is the parameters of an RNN.