Label
for IEEE Conferences

Shashank Gupta[1,2] Antonio Robles-Kelly[2]
[1]School of Computer Science and Eng., Vellore Institute of Technology, Vellore, Tamil Nadu 632014, India
[2]Faculty of Sci., Eng. and Built Env., Deakin University, Waurn Ponds, VIC 3216, Australia
shashankgupta314@gmail.com *antonio.robles-kelly@deakin.edu.au*

***Abstract*—We present a flexible method of providing human intentions and domain knowledge during the supervised or unsupervised training of general types of neural networks on classification tasks based on feature extracting functions. We combine the declarative first-order logic rules which represents the human knowledge in logical structured format [1] with noisy labeling functions which acts as decision rules [2] to develop a modified version called feature-extracting functions. These functions are simple programming functions which represents the human expertise as a set of data-manipulative logical instructions and provides a cumulative set of modified distribution of the data. They are applied during the training process over a mini-batch directly. As these functions are just simple programming functions, they do not require any kind of special mathematical encoding and thus, have high expressive power for natural language representation which contributes to their flexibility and ease of use.**

I. Introduction

Deep Neural Networks have made a significant impact in the field of pattern recognition. They are able to provide high levels of performance in terms of both accuracy and efficiency on all kinds of supervised and unsupervised problems. Thus, there has been an explosion of interest in developing Deep-learning based systems across industry, government and academia.

But in order to achieve that state of the art performance, they require massive amounts of data being fed during their training procedure. This procedure is often purely data-driven as is there is no direct or indirect human intervention involved. As a result, we get a model whose decision making process is completely non-interpretable and just acts as a black-box mapping from input data to output predictions. Previous work has shown that supervision purely in the form of data can lead a model to learn some unwanted patterns and provide wrong decisions [3][4]. As critical processes such as medical diagnosis, planning, and control require a level of trust associated with the machine output, it is essentially important to understand how these models decide.

One of the ways to make their decision making process interpretable is to encode the intended rules or patterns derived from human domain knowledge in their trainable parameters. That is provide some sort of direct or indirect supervision in their training process to make them capture human intentions and desired patterns. This process can be termed as combining structured knowledge representing high-level cognition with Neural systems[5]. Logic rules provide a means to represent the human knowledge in a structured format but suffers from limited expressiveness and flexibility issues as they need to be translated from natural language to logical representation and also require a proper encoding format which is highly problem or task specific.

A recent paper[1] provides a general framework for encoding the human knowledge into the parameters of the model via a novel indirect supervision training method called Iterative-knowledge Distillation. It represents the desired rules and patterns as logical structured knowledge in the form of a set of declarative first-order logic rules which are encoded using soft-logic[6]. They define a parametric base neural network as student network which needs to be provided with logic rules knowledge and construct a non-parametric teacher network which is a projection of the base network over a regularized sub-space in training data constrained by logic rules as soft boundaries. This is achieved via adapting posterior-regularization[7] in a logic rules constraints setting. At each iteration, they calculate the initial conditional probability distribution of the student network which is essentially a softmax prediction vector, construct a set of valid distributions as rule-regularized sub-spaces in training data via posterior regularization, finds a distribution which is closest to student network distribution via KL-Divergence and finally, solves an optimization problem which helps the student network to imitate between the truth values and teacher network soft-predictions. The student network is trained to emulate between the teacher network output which represents rule-knowledge in the form of conditional probability distribution and truth values from the labeled data in the overall objective function and update the parameters of base network using a gradient-descent based technique.

In our study, we find that the proposed use of logic rules for representing human knowledge can be applied only for feature extracting purposes that is supervising the network on what features it should use while providing predictions given the presence of certain features. Moreover, the teacher network just represents soft-predictions calculated using rules knowledge which is transferred into the weights of the student network via a appropriate loss function. Thus, we propose the use of feature extracting functions instead of constructing the teacher network with logic rules. These functions are directly applied on the data and transfers the human knowledge into a

modified set of distribution of input data. We find that using this approach 1) Eliminates the need of finding initial posterior distributions of logic-rule features in train, validation and test data, 2) Avoids the task of conversion of human knowledge from natural language to logical format and subsequently 3) Requirement of any special encoding format for logic rules specific to a task. But most importantly it, 4) Eliminates the need of constructing a teacher network.

Analogous to their approach, we derive a feature-extracting function from each logic rule which is essentially just a programming function and can be applied on a mini-batch during an iteration directly. Since we are applying the functions on the data, manipulating it's distributions for certain features and not transferring the rule knowledge into the weights of the network, we do not require to construct a teacher network thus removing a lot of complexity. Also, these functions can be modified at any time during the training process, thus provides a lot of flexibility in what proportion of data needs to be modified or manipulated.

We show that our approach can be used in every use-case possible for the original approach (Iterative-distillation) and find that simple programming functions have far more expressive capabilities to represent human intentions in natural language and works very well with every kind of variation in the training procedure. We provide a more direct nature of supervision which is applied on the input data during an iteration rather then on all the available data during preprocessing.

II. Methodology

The original approach (Iterative-distillation)[1] is used to encapsulate the human knowledge represented as logical structured knowledge into the trainable parameters of the neural network. This is achieved by making the network learn from soft-predictions of a teacher network explicitly representing rules knowledge which is also evolved during each iteration. Figure 1 shows an overview of the framework.

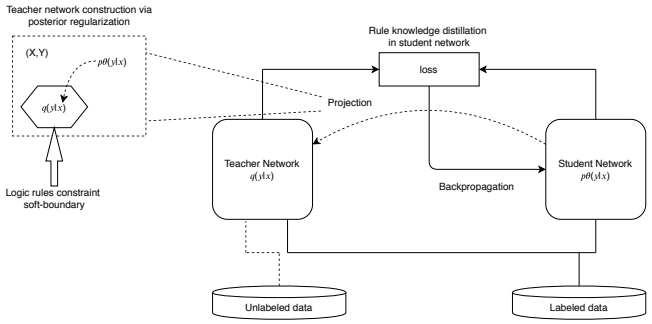

Figure 1: Iterative rule-knowledge distillation overview [1]. At each iteration, a teacher network $q(y|x)$ is constructed as a rule-regularized projection of student network $p_\theta(y|x)$; and the student network is trained to imitate both the truth labels from labeled training data and teacher network output.

Our proposed approach works in a slightly different manner while being applicable to every use-case of the original approach. Instead, we develop feature extracting functions from human knowledge which are basically programming functions and does the same job as what logic rules does in original method which is to take the input data and calculate the distribution of rule feature. But now, they feed the calculated distribution directly to the base network for training and eliminates the need for constructing a teacher network. Thus, we significantly reduce the complexity of the framework. As in previous method, these functions can be applied either during the training process in each iteration or during the preprocessing phase of the data. Figure 2 shows an overview of our proposed approach.

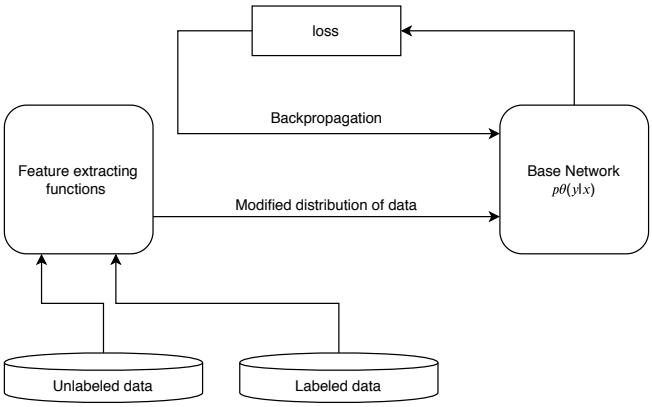

Figure 2: Proposed system overview. At each iteration, a batch of data is passed to the feature extracting functions which outputs modified distribution of data; and the base network is trained on the modified data.

A. Input

This section explains about the input learning resources being given to the base network for training.

In Iterative-distillation[1], the student network is made to learn from both labeled instances and structured logical knowledge represented by a set of declarative first-order logic rules. These logic rules are encoded using soft-logic [6] for flexible usage in constructing soft-boundaries for calculating rule-regularized distributions. The learning resources constitute of training data $D = \{(x_n, y_n)\}_{n=1}^{N}$ as a set of N instantiations of (x,y) where x is a set of input independent variables and y is dependent target variable, and a set of logic rules as $R = \{(R_l, \lambda_l)\}_{l=1}^{L}$ where R_l is the lth rule constructed from human knowledge. To implement a rule R_l on data, it is represented as a set of groundings $\{(r_{lg}(D))\}_{g=1}^{G_l}$ on D.

For example, let's say we have a movie reviews dataset in which x represents a sentence as set of tokens or words and corresponding y represents the sentiment value which is 0 for negative and 1 for positive. Thus we can create a batch of N sentences as $D = \{(x_n, y_n)\}_{n=1}^{N}$. From the knowledge of English language, we know that in most of the cases, if a sentence has 'A-but-B' structure, then the sentiment of the whole sentence is consistent with the sentiment of 'B' part. Therefore, we can create a 'A-but-B' logic rule with it's $\lambda_l = 1$. To encode this formally, we define a Boolean random

variable $r_{lg}(x, y)$ = "Whether the sentence x has 'A-but-B' structure" calculate it's distribution in D.

Our method combines the input data and human knowledge to provide a modified output data which is used for training base neural network. We represent the input data as $D = \{(x_n, y_n)\}_{n=1}^{N}$ as a set of N instantiations of (x,y) where x is a set of input independent variables and y is dependent target variable. The human knowledge is represented as $F = \{(F_l(D)\}_{l=1}^{L}$ as a set of L feature extracting functions which are applied on D. These are simple programming functions thus do not require any special encoding and can be applied on input data directly. In the previous example, instead of creating a random variable feature for implementing 'A-but-B' rule, we write a function as $F_l = A-but-B(x, y)$ which outputs (x*,y) where x* has only 'B' features.

B. Feature extracting functions

This section gives a brief explanation about the Iterative-distillation method[1] with construction of a teacher network and transfer of knowledge from teacher to student network. Then, we show how our approach can be used to replace the teacher element using programmable functions and provide a more simple and flexible solution for transferring knowledge.

At each iteration of the training process, the initial conditional probability distribution of student network $p_\theta(y|x)$ and the initial distributions of every rule-feature denoted by random variables $\{(r_{lg}(D))\}_{g=1}^{G_l}$ is calculated on input data D. In order to find a posterior probability distribution $q(y|x)$ which fits the rules and at the same is close to $p_\theta(y|x)$, we adapt the posterior regularization technique[7] to find a set of valid or allowed distributions denoted as Q. Thus we calculate the posteriors of every rule-feature distribution denoted as $p_\theta(y|rule - features)$ and apply the rule constraints via an Expectation operator to find set Q. Formally, it can be denoted as $Q = \{q(y|x) : E_q[r_{lg}(D)] = 1\}$. Every $q(y|x)$ in Q defines a valid distribution or a rule-regularized subspace in D. Now to find a $q(y|x)$ in Q which is closest to $p_\theta(y|x)$, we use KL-Divergence between them and wish to minimize it. Thus, the resultant $q(y|x)$ is our desired teacher network output which represents the rules-knowledge in a conditional probability distribution. Further, this knowledge is distilled into the parameters of the student network by introducing an additional loss term in which it is forced to imitate the soft-predictions $q(y|x)$ and accordingly update it's parameters. The following equations[1] summarizes the Iterative-distillation procedure.

$$\begin{aligned} \min_{q,\xi\geq 0} KL(q(y|x)||p_\theta(y|x)) + C\Sigma_{l,g_l}\xi_{l,g_l} \\ where\ \lambda_l(1 - E_q(r_{l,g_l}(D))] \leq \xi_{l,g_l} \\ g_l = 1, ..., G_l, l = 1, ..., L \end{aligned} \tag{1}$$

$$\begin{aligned} \theta^{t+1} = \arg\min_{\theta\in\Theta} \frac{1}{N}\Sigma_{n=1}^{N}(1-\pi)l(y_n, p_\theta(y|x)) \\ + \pi l(q(y|x), p_\theta(y|x)) \end{aligned} \tag{2}$$

At each iteration, solving Eq.(1) in which $\xi_{l,g_l} \geq 0$ represents slack variable for respective rule constraint and C is the regularization parameter will give the desired output of a rule-regularized projection of $p_\theta(y|x)$ which is $q(y|x)$. Plugging that $q(y|x)$ in the Eq.(2) which is the objective function for student network will distill the rule-knowledge into the trainable parameters while updating them using any Gradient Descent technique like AdaDelta[8]. In Eq.(2), l denotes the loss function selected according to specific applications (e.g., the cross entropy loss for classification), t denotes the iteration and π denotes the imitation parameter calibrating the relative importance of the two objectives.

As first-order logic can only represent static relationships between objects, they cannot accommodate the dynamic nature of rules and patterns in the real world data. Since the process of transferring rule-knowledge into the parameters is permanent and irreversible, the trained model can fail to provide correct output predictions in a real world scenario. Thus we justify the use of feature extracting functions which will apply only on input data, modify it's distribution and feed it to the neural network.

Inspired by the labeling functions used in a data-programming framework called Snorkel[2] which takes the input of an x and provides an output class-label based on the analysis of it's features, we construct feature extraction functions which also takes the input x but instead provide an output x* which has modified values of features based on the analysis of x. Since a neural network is parametric based model which calculates output as weighted sum of features, by passing the modified values of features from the function, we get an output which represents the desired intended prediction according to human knowledge and thus is interpretable. Following provides a skeletal format for the feature extracting functions in both mathematical and programmatic encodings while taking the example of A-but-B rule.

$$\begin{aligned} F_l : x \rightarrow x* \\ s.t.\ F_l(x) = A - but - B(x) = x* \end{aligned}$$

Writing a programmatic function

```
def A-but-B(x):

    if A-but-B feature in x:
        x* = 'B' features of x
    else:
        x* = x

    return x*
```

We can see that the x* explicitly represents the combination of input data and the human knowledge which can be directly fed to the data loss term of the student network in Eq.(2). Now, we can rewrite the Eq.(2) by eliminating the loss term for teacher network as:

$$\theta^{t+1} = \arg\min_{\theta \in \Theta} \frac{1}{N} \Sigma_{n=1}^{N} (1-\pi) l(y_n, p_\theta(y|x*)) \qquad (3)$$

where x* is the cumulative output of all the feature extracting functions and $p_\theta(y|x*)$ is the conditional probability distribution on x*. Since the information is purely present in a modified feature-set form, we are not encoding it into the parameters of the model. Thus the feature extracting functions becomes an ad-hoc initialization of input data but with the exception that they are applied during the training process and thus can be used at test time to help model predict output using the human knowledge.

III. Implementation

This section explains the training and testing process of our method summarized in Algorithm 1 and 2 respectively.

Algorithm 1: Training

Input: The training batch $D = \{(x_n, y_n)\}_{n=1}^{N}$,
The functions set $F = \{(F_l(D)\}_{l=1}^{L}$
Initialize the neural network parameters θ
while *Iteration* **do**
1: Calculate $D^* = \{(x_n^*, y_n)\}_{n=1}^{N}$
2: Calculate probability distribution $p_\theta(y|x*)$
3: Update the parameters θ using objective function in Eq.(3)
end
Output: Trained neural network p_θ

Algorithm 2: Testing

Input: The testing batch $D = \{(x_n, y_n)\}_{n=1}^{N}$,
The functions set $F = \{(F_l(D)\}_{l=1}^{L}$
1: Calculate $D^* = \{(x_n^*, y_n)\}_{n=1}^{N}$
2: Calculate probability distribution $p_\theta(y|x*)$
3: Predict the class-label using $\arg\max p_\theta(y|x*)$
Output: Neural network prediction inline with human intention

During each iteration, we calculate the modified distribution as $D^* = \{(x_n^*, y_n)\}_{n=1}^{N}$ which is a cumulative output of every function $F_l \in F$ applied on input batch $D = \{(x_n, y_n)\}_{n=1}^{N}$. We pass on this distribution to the neural network and calculate the conditional probability distribution as $p_\theta(y|x*)$ where each $(x^*, y) \in D^*$. There maybe a possible situation where two or more than two rules are conflicting in nature. That's where the flexibility of our functions come in handy since we can choose which functions to apply on D and change the set F anytime during training.

IV. Experiments

We test our approach by applying it on a Sentiment-classification task using base network as Convolutional Neural Network. We perform sentence-level sentiment analysis and classify each sentence into positive or negative category. For CNN, we use the state-of-the art architecture proposed in [9] which has achieved compelling performance on various sentiment classification benchmarks. We use it's "non-static" version with the exact same configurations. Specifically, word vectors are initialized using word2vec[10] and fine-tuned throughout training, and the neural parameters are trained using SGD with the AdaDelta update rule[8].

Since contrasive senses are hard to capture for a model, we define a linguistically motivated rule called 'A-but-B' rule[1] which states that if a sentence has a 'A-but-B' structure, the sentiment of whole sentence will be consistent with the sentiment of it's 'B' part. Accordingly, we define a function for this rule as 'has-A-but-B-structure(S)' which outputs features of 'B' part.

So far, we have evaluated our method on three public datasets which are as follows:-
1) **SST2:** Stanford Sentiment Treebank[11] which contains 2 classes (negative and positive), and 6920/872/1821 sentences in the train/dev/test sets respectively. Following (Kim, 2014) we train models on both sentences and phrases since all labels are provided.
2) **MR:**[12], a set of 10,662 one-sentence movie reviews with negative or positive sentiment.
3) **CR:**[13], customer reviews of various products, containing 2 classes and 3,775 instances.

For MR and CR, we use 10-fold cross validation as in previous work. In each of the three datasets, around 15 percent sentences contains the 'A-but-B' structure.

V. Results

In this section, we present a comparative study of the results of Plain CNN non-static version[9], Iterative-distillation method[1] and our method on all the three datasets.

Model	SST2	MR	CR
CNN	87.2	81.3±0.1	84.3±0.2
CNN-rule	88.8	81.6±0.1	85.0±0.3
CNN-F	89.1	81.8±0.4	84.8±0.1

Table 1: Accuracy percentages of sentiment classification task. Row 1, CNN is the base network corresponding to the CNN-non-static model in [9]. Row 2 is the enhanced CNN model (student network $p_\theta(y|x)$) in Iterative-distillation[1] whose parameters have the logic rules knowledge encoded in them. Row 3 is the our method $p_\theta(y|x*)$ performance on all 3 datasets. For MR and CR, we report the average accuracy ± one standard deviation using 10-fold cross validation.

Model	SST2		
	Precision	Recall	F1-score
CNN	0.89	0.85	0.87
CNN-rule	0.90	0.87	0.88
CNN-F	0.91	0.87	0.89
Model	**MR**		
	Precision	Recall	F1-score
CNN	0.80±0.016	0.83±0.018	0.82±0.010
CNN-rule	0.81±0.014	0.82±0.020	0.82±0.010
CNN-F	0.81±0.014	0.83±0.012	0.82±0.010
Model	**CR**		
	Precision	Recall	F1-score
CNN	0.78±0.038	0.78±0.054	0.78±0.031
CNN-rule	0.77±0.044	0.79±0.046	0.78±0.027
CNN-F	0.76±0.045	0.78±0.047	0.77±0.026

Table 2: Precision, recall and f1-scores sentiment classification task. Row 1, CNN is the base network corresponding to the CNN-non-static model in [9]. Row 2 is the enhanced CNN model (student network $p_\theta(y|x)$) in Iterative-distillation[1] whose parameters have the logic rules knowledge encoded in them. Row 3 is the our method $p_\theta(y|x*)$ performance on all 3 datasets. For MR and CR, we report the average values of precision, recall and f1-scores $\pm$ one standard deviation using 10-fold cross validation.

From the experimental results, we show that our method is a promising avenue for further exploration. The boost on accuracy is not very significant as only one rule of A-but-B feature is constructed and compared. If there many more rules representing complex relationships between features, we think that our method has the potential to provide a huge boost in performance. Also, experimenting with varying sizes of unlabeled and labeled sets in training data can uncover some interesting information about performance and applications.

VI. Conclusion

We have presented an approach to provide an interpretable machine output via representing human knowledge in programmable feature extracting functions which directly controls what features or what values of input data should the model consider for providing output given the presence of a certain property in data. We have shown that using feature extracting functions, we can create a model whose posterior output can be influenced by model parameters as well as direct supervision from functions. Thus eliminating the need of transferring that supervision knowledge into the parameters.

To our understanding, we can only see one drawback of this approach that our model will not be considered as an enhanced model since it does not have the knowledge encoded in it's parameters and cannot make a decision on it's own. But if we take our approach in-terms of applicability, it can be used in each and every possible use-case of the rule-enhanced model.

VII. Future work

We would like to further expand on the use of feature-extracting functions in constructing a rule-enhanced model and see what new interesting information we can discover in this field. Also, we would like to evaluate our method on a range of other classification tasks as well as sequence learning tasks using the base neural network as a recurrent neural network.

References

[1] Z. Hu, X. Ma, Z. Liu, E. H. Hovy, and E. P. Xing, "Harnessing deep neural networks with logic rules," *CoRR*, vol. abs/1603.06318, 2016. [Online]. Available: http://arxiv.org/abs/1603.06318

[2] A. Ratner, S. H. Bach, H. R. Ehrenberg, J. A. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," *CoRR*, vol. abs/1711.10160, 2017. [Online]. Available: http://arxiv.org/abs/1711.10160

[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014. [Online]. Available: http://arxiv.org/abs/1312.6199

[4] A. M. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," *CoRR*, vol. abs/1412.1897, 2014. [Online]. Available: http://arxiv.org/abs/1412.1897

[5] A. Gabbay, A. Garcez, K. Broda, D. Gabbay, and P. Gabbay, *Neural-Symbolic Learning Systems: Foundations and Applications*, ser. Perspectives in Neural Computing. Springer London, 2002. [Online]. Available: https://books.google.com.au/books?id=6NZYVSOyD-UC

[6] S. H. Bach, M. Broecheler, B. Huang, and L. Getoor, "Hinge-loss markov random fields and probabilistic soft logic," *CoRR*, vol. abs/1505.04406, 2015. [Online]. Available: http://arxiv.org/abs/1505.04406

[7] K. Ganchev, J. Graa, J. Gillenwater, and B. Taskar, "Posterior regularization for structured latent variable models," 01 2010.

[8] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012. [Online]. Available: http://arxiv.org/abs/1212.5701

[9] Y. Kim, "Convolutional neural networks for sentence classification," *CoRR*, vol. abs/1408.5882, 2014. [Online]. Available: http://arxiv.org/abs/1408.5882

[10] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," *CoRR*, vol. abs/1405.4053, 2014. [Online]. Available: http://arxiv.org/abs/1405.4053

[11] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. [Online]. Available: https://www.aclweb.org/anthology/D14-1162

[12] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Ann Arbor, Michigan: Association for Computational Linguistics, Jun. 2005, pp. 115–124. [Online]. Available: https://www.aclweb.org/anthology/P05-1015

[13] M. Hu and B. Liu, "Mining and summarizing customer reviews," 08 2004, pp. 168–177.