2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)

Natural Language Explanations of Classifier Behavior

Rodrigo Monteiro de Aquino
Escola Politécnica - Universidade de São Paulo
São Paulo, Brazil
rodrigo-monteiro.aquino@usp.br

Fabio Gagliardi Cozman
Escola Politécnica - Universidade de São Paulo
São Paulo, Brazil
fgcozman@usp.br

***Abstract*—Tools that enhance interpretability of classifiers tend to focus on the knowledgeable data scientist. Here we propose techniques that generate textual explanations of the internal behavior of a given classifier, aiming at less technically proficient users of machine learning resources. Our approach has been positively evaluated by a group of users who received its textual output.**

***Index Terms*—machine learning, interpretability, transparency**

I. Introduction and Motivation

The machine learning community has dedicated significant effort to develop techniques that interpret black-box classifiers such as deep neural networks [1], [2]. As complex classifiers meet widespread application, it is important to make them understandable to a broader array of people. On top of that, recent legal issues have emerged that reinforce the need to explain decisions taken autonomously by complex classifiers. For instance, the General Data Protection Regulation [3], currently in force in the European Union (EU), and the law for personal data protection, recently implemented in Brazil, require automatic decisions to be explained if so requested.

Several research results have appeared on interpreting and explaining classifiers. Typically one builds an "interpreter" that examines the available data and the classifier, and outputs explanations geared to the user. Of course, the best way to explain a classifier depends on the end user: a data scientist may be very happy with a linear equation and a few graphs relating weights to outputs, while a less proficient user, say a member of a legal team or a final customer, may be very uncomfortable with a mathematical explanation. In particular, auditing bodies that oversee whether automatic classifiers can be properly interpreted may benefit from explanations at a lesser mathematical level of detail.

We present here a set of techniques that emphasize textual explanations; the goal is to generate a readable explanation for the behavior of a given (complex) classifier. Natural language explanations can be easily processed by a variety of users; in particular we aim at users with some mathematical sophistication but no serious knowledge of data science — the kind of user we anticipate to see in auditing and regulating bodies.

The work has been supported by Itaú Unibanco S.A. through Itaú Scholarship Program. The second author is partially supported by CNPq.

The paper is organized as follows. Section II discusses related work, and Section III presents our proposed approach to textual explanations. In section IV we evaluated the proposed technique on a fitted model, and an experimental validation study is described in Section V.

II. Related Work

Even though an accepted formal definition of interpretability is still to be found, there is wide interest in making automatic classification results more interpretable to the human end user [4].

A few popular models employed in machine learning are taken to be intrinsically interpretable: for instance, simple logistic regressions and shallow decision trees [5]. A simple analysis of weights or a simple visual inspection are enough to reveal the behavior of these models.

Non-interpretable models, often referred to as black box models, can be explained using a set of tools in the literature. Explanations may focus a "local" prediction (that is, they can explain why a particular output was generated by a particular input) or they may offer a global view of the model behavior [5].

Some interpretability techniques are only applicable to specific models, such as neural networks; they are said to be *model-specific*. Other techniques can in principle be applied to any model; they are said to be *model-agnostic*. Techniques such as Layer-wise Relevance Propagation (LRP) and deep Taylor decomposition [6] are examples of model-specific techniques. On the other hand, LIME [2] and Partial Dependence Plots (PDP) [7] are examples of model-agnostic techniques.

The approach followed by LIME is to create a surrogate (and easily interpretable) model that locally approximates a given complex model. As the interpretable model can be used to approximate any black box, the resulting method is agnostic. A freely distributed implementation of LIME can take data in tabular or textual form, or even imagens. Fig. 1 shows an example of explanation given by LIME for a particular prediction.

An important point about these previous techniques, and indeed many others in the literature [5], [8], [9], is that they aim their explanations at a data scientist. That is, they assume a rather sophisticated user.

978-1-7281-1488-0/19/$31.00 ©2019 IEEE
DOI 10.1109/AIKE.2019.00048

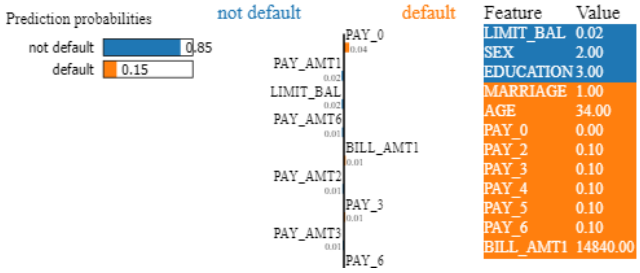

Fig. 1. Example of a LIME explanation.

On the contrary, Biran, Join and Mckeown [10] focus on "human-centered" explanations that are based on natural language generation (NLG) using narrative roles. Even though their approach produces explanations that are similar to the ones we generate, their work focused on the selection of core messages to be presented; we instead focus on thorough explanations of model's behavior. That is, we wish to provide readable, even if long, explanations to a layperson, say a banking client or a lawyer busy with patent selection. To do so we use ideas behind the tools described in this section, but we use natural language to present analyses and conclusions.

III. Natural Language Explanations

In short, we wish to create an agnostic framework that generates textual explanations. To this end, the output of a given model must be analyzed as its inputs vary, and the relationship between input and output must be summarized in a text report. There are techniques in the literature that aim at capturing input-output relationships; in particular we resort here to Partial Dependence Plots (PDPs) [7]. Note however that we do not need to actually draw PDPs; we only use the main ideas and calculations behind them to generated explanations through Natural Language Generation (NLG).

A. Partial Dependence

A Partial Dependence (PD) function [7] shows the marginal effect on the model's predicted outcome when we vary the values of a subset of the input features. A PDP offers a graphical representation of such a function, exposing the causal relation of the variation of such features in the classifier's predictions; as a PDP is limited to three dimensions for presentation purposes, it is restricted to two input features at a time.

For a classifier model that produces probabilities of classes, the PD functions return the probability of a class to have a true value when the features of interest have certain values. Equation (1) defines a partial dependence function:

$$\bar{f_S}(X_S) = \frac{1}{N}\sum_{i=1}^{N} f(X_S, x_{iC}), \qquad (1)$$

where $\bar{f_S}$ stands for the marginal average of the function defined by the model, X_S is a subset of interest of the input features, N is the total number of training samples, x_{iC} assumes values of the complement of X_S in the the training data.

If we want to analyze the marginal influence of two features in the prediction of a model, for each combination of values of the features of interest we must calculate the mean result of the predictions obtained from the variation of the other features within a certain range. The result of this calculation is a global view of the model's behavior; the plots help understanding how the model behaves when the selected features varies within a certain range of values. Such range is usually defined by looking at the training data used to fit the model.

It is also possible to apply the PDP in categorical features by estimating its results setting all data points to the value of the interested category and getting the average result from the model. For example, with a feature named "MARITAL STATUS" there could be 3 possible values: "SINGLE", "MARRIED" and "DIVORCED". An estimate of the PDP for this feature with another one would be achieved assigning each possible value in this categorical feature to all data instances and averaging its results.

As our goal is to generate textual explanations instead of graphs, there is no limitation on the number of features analyzed simultaneously in our approach. The description of a model's behavior can be as complex as requested by the user. Thus one important aspect of our proposal is to allow the user to control the granularity of reports. Such granularity is tuned by a parameter, selected by the user, that indicates the maximum number of features that can vary in a particular analysis. Following the idea behind PDPs, the focus is on the surfaces of probability as features vary. These surfaces capture tendencies of a particular class label to be selected under various circumstances.

B. Gradient Analysis

Our approach is to compute the partial dependence for each possible label; from there, to generate descriptive texts about the model's behavior by describing the gradient throughout the probability surfaces. The complete process is illustrated in Fig. 2, which takes any fitted classifier and using the data from an analysis generates a report of its behavior. We resort to standard methods of gradient computation [11] as implemented in the Numpy library (in the Python programming language).

For each class, the gradient is calculated in each point of the defined surface. As it is unfeasible to describe a high-dimensional surface by detailing its behavior at every possible point, surfaces are divided in sectors that are then represented by the mean gradient of its internal points.

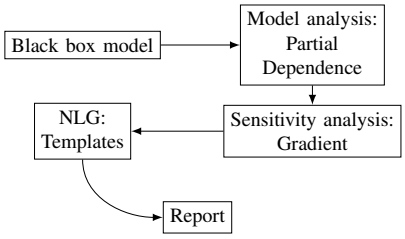

Fig. 2. Steps in report generation.

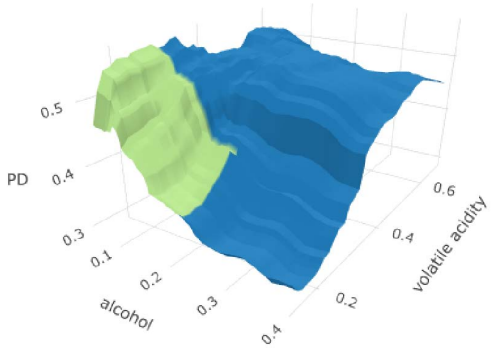

Fig. 3. Representation of sectors of a PDP.

As an example, the visualization of sectors in a three dimensional space (PDP with two input features) is depicted in Figure 3. In it, two features of interest have their values fixed and the PD function is calculated to generate the value on the vertical axis (that is, the average of the probability of the label of interest over all possible values of the non-fixed features).

Algorithm 1 details how the sectors in the PD function surface are determined, and how the mean gradient is calculated in each one of them. Steps in this algorithm are repeated for each label, limited only by a *sensitivity* value that limits the effect of less relevant labels, and controls the size of the sectors. The computed gradient is averaged in each sector to determine a trend; this information feeds the natural language generation step.

Algorithm 1 Gradient Analysis

Input: *pd_data* (PD data for one particular class)
Output: list of $(bound\_index, mean\_gradient)$
  slices_perc_sizes ← percentual size for each feature in PD
  dim_stride ← empty list
  $num\_features$ ← length($pd\_data.shape$)
  for $i = 0$ to $num\_features - 1$ **do**
    shape_size ← pd_data.shape[i]
    perc_size ← slices_perc_sizes[i]
    dim_stride.append($\lfloor shape\_size * perc\_size \rfloor$)
  end for
  do_loop ← True
  slices ← empty list
  indexes ← list with size $num\_features$, filled with zeroes
  while $do\_loop = True$ **do**
    bound_index ← empty tuple
    for $i = 0$ to $num\_features - 1$ **do**
      $last\_index \leftarrow indexes[i] + dim\_stride[i]$
      if $last\_index \geq pd\_data.shape[i]$ **then**
        $last\_index \leftarrow pd\_data.shape[i] - 1$
      end if
      $bound\_ind \leftarrow bound\_ind + (indexes[i], last\_index)$
    end for
    $mean\_grad = mean(grad(pdp\_data[bound\_ind]))$
    $slices.append((bound\_ind, mean\_grad))$
    $indexes[0] = indexes[0] + dim\_stride[0]$
    for $i = 1$ to $num\_features - 1$ **do**
      if $indexes[i-1] \geq pdp\_data.shape[i-1]$ **then**
        $indexes[i] = indexes[i] + dim\_stride[i]$
        $indexes[i-1] = 0$
      end if
    end for
    if $indexes[-1] \geq pdp\_data.shape[-1]$ **then**
      $do\_loop = False$
    end if
  end while
  return $slices$

C. Natural Language Generation

Usually, the description of a chart focuses on increasing and decreasing steps of the function of interest. In words, one might say, "There is an increase in the probability of label y when feature X varies from a_1 to a_n and feature Z varies from b_1 to b_n". As we have gradients over sectors of probability surfaces, we can generate datapoints by dividing sectors of increase/decrease (as measured by mean gradient) of probabilities.

Dynamic templates [12], [13] are the best fit to the format of the resulting data, and can be used to generate text snippets with respect to each sector. Depending on the data from the analysis there is a corresponding snippet of text to be filled with it, also the selected sensitivity value influence how the information may grouped within similar clusters of near sectors.

IV. Experiments

The evaluation of our approach was based on the generation of a behavior explanation for a Random Forest model with 300 estimators and a maximum depth of 50. We focused on random forests as wanted to emphasize that our approach is not solely geared towards "complicated" deep neural networks (for which model-specific methods exist); other common classifiers can be quite complex and difficult to understand. The data we used was the Wine Quality Data Set [14], a data set containing physicochemical properties related to red and white variants of a Portuguese wine, with a label determining the class of each sample, from 0 (very bad) to 10 (excellent).

Figure 4 shows a part of the resulting output, which describes how the probability of class 3 wines (the most relevant one) changes when features **alcohol** and **volatile acidity** vary within the range of the training data set. The text comments on the probability of each class of wine given the training data set.

Note that this approach is not focused on showing whether the training is balanced, nor whether the model is properly encoding the data. The output represents the model's behavior in the space of possible values of the features.

Even though the sensitivity variable was set to 0.4 on average (meaning that the first sector will have 60% of the total size of the feature space) the generated text is quite extensive if we consider all labels. The larger the sensitivity the smaller the sectors, causing the text to be more detailed.

Overall, class 3 has a considerable decrease in chance to occur when features alcohol and volatile acidity increases. For this same class, it is detailed above 4 ranges of values where the input features vary, showing how the output may change: There is a considerable decrease of chance for this class to occur when:
- feature alcohol varies from value 0.14 to 0.46 and
- feature volatile acidity varies from value 0.05 to 0.17.

There is a major decrease of chance for this class to occur when:
- feature alcohol varies from value 0.46 to 0.68 and
- feature volatile acidity varies from value 0.05 to 0.39.

There is a major increase of chance for this class to occur when:
- feature alcohol varies from value 0.14 to 0.46 and
- feature volatile acidity varies from value 0.17 to 0.39.

Fig. 4. Text result for the Wine Quality Data Set.

TABLE I
EVALUATION RESULTS

Technique	Quick understanding of information			Seemingly reliable result			Chance of usage		
	Negative	Neutral	Positive	Negative	Neutral	Positive	Negative	Neutral	Positive
PDP	1	36	11	12	28	8	1	30	17
LIME	2	29	17	2	22	24	2	22	24
Proposed Technique	3	16	29	5	30	13	4	26	18

It is to be expected that with a more complex problem (with more features and labels) the textual explanation for the model's behavior will be significantly longer. A longer report is not a problem: the granularity of the report can be controlled, so the end user can choose anything from a few paragraphs to a book-length description.

V. VALIDATION

To evaluate the effective application of our approach, an experiment was conducted on a machine learning class of 48 students from a Master Engineering degree. These students heard first a 30 minute explanation about interpretability and several tools related to the field, focusing on LIME, PDP and the proposed framework, then they were asked to participate on the validation experiment.

This experiment consisted of showing the results obtained by two state of the art interpretability model-agnostic techniques: PDP and LIME, an then showing the results obtained by our approach, with the objective of compare techniques. For each technique presented, students were asked to score with respect to three criteria:

1) **Quick understanding of information:** How much effort the student spent to understand the underlying information represented by the technique.
2) **Seemingly reliable result:** Whether the result given by the technique appeared correct and reliable.
3) **Chance of usage:** Whether the student would use the technique in a real project.

These criteria were related to "satisfaction", with possible values: **Positive**, **Neutral** and **Negative**. For instance, a positive satisfaction for the first criterion means that little effort was spent to understand the information provided by the method.

The evaluation results presented on Table I indicate that our approach outperformed other approaches with respect to the first criterion in positive gradings, while it was equally as satisfactory with respect to the other criteria (assuming non negative responses) — suggesting that textual explanations led to more satisfactory understanding. This is particularly interesting in our setting as students were knowledgeable about mathematical expressions and graphs; even then text was the preferred medium.

VI. CONCLUSION AND FUTURE WORK

This work presented a framework that generates a textual explanation of a classifier behavior with respect to the variation of some of its input features. We employed state of the art model-agnostic interpretability techniques and developed new algorithms to analyze a classifier model, generating a text representing the model behavior with a series of dynamically generated template-based sentences.

This framework can be improved with additional interpretability techniques such as Individual Conditional Expectation [15] and Accumulated Local Effects [16]. The first enables a more local explanation and the second is an alternative to the PDP. The dynamic generation of text should also be improved by implementing more sophisticated NLG techniques, enabling a text more fluid and less "machine-like".

REFERENCES

[1] DARPA-BAA-16-53, "Explainable Artificial Intelligence (XAI)," *Defense Advanced Research Projects Agency*, p. 1.

[2] M. T. Ribeiro, S. Singh, and C. 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*, ser. KDD '16, 2016, pp. 1135–1144.

[3] European Union, "Regulation 2016/679 of the European Parliament and the Council of the European Union," *Official Journal of the European Communities*, pp. 1–88, 2016.

[4] F. Doshi-Velez and B. Kim, "Towards A Rigorous Science of Interpretable Machine Learning," 2017. [Online]. Available: http://arxiv.org/abs/1702.08608

[5] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 93:1–93:42, Aug. 2018.

[6] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, "Explaining nonlinear classification decisions with deep Taylor decomposition," *Pattern Recognition*, pp. 211–222, may.

[7] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[8] Y. Hechtlinger, "Interpretation of Prediction Models Using the Input Gradient," 2016. [Online]. Available: http://arxiv.org/abs/1611.07634

[9] N. Puri, P. Gupta, P. Agarwal, S. Verma, and B. Krishnamurthy, "MAGIX: Model Agnostic Globally Interpretable Explanations."

[10] O. Biran and K. Mckeown, "Human-centric justification of machine learning predictions," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, ser. IJCAI'17, 2017, pp. 1461–1467.

[11] B. Fornberg, "Generation of finite difference formulas on arbitrarily spaced grids," *Mathematics of Computation*, 1988.

[12] A. Gatt and E. Krahmer, "Survey of the state of the art in natural language generation: Core tasks, applications and evaluation," *Journal of Artificial Intelligence Research*, 2018.

[13] E. Reiter and R. Dale, *Building Natural Language Generation Systems*. Cambridge University Press, 2000.

[14] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Systems*, no. 4, pp. 547–553, nov.

[15] A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin, "Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation," *Journal of Computational and Graphical Statistics*, no. 1, pp. 44–65, sep.

[16] D. W. Apley, "Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models," dec 2016. [Online]. Available: http://arxiv.org/abs/1612.08468