A Model-Agnostic Approach to Interpreting Sequence Predictions

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recurrent neural networks are a standard building block in numerous machine learning domains, from natural language processing to time-series classification. While their application has grown ubiquitous, our understanding of their inner workings is still lacking. In practice, the complex decision-making in these models is seen as a black-box, creating a tension between accuracy and interpretability. Moreover, the ability to understand the reasoning process of a model is important in order to debug it and, even more so, to build trust in its decisions. Although, in recent years, considerable research effort has been guided towards explaining black-box models, recurrent networks have received relatively little attention. KernelSHAP, a landmark model-agnostic explainer that proved to be the single optimal solution in a large family of explainers, is yet to be extended to recurrent architectures. Additionally, any method that aims to explain predictions/decisions from a sequence of events should assess, not only feature importance, but also event importance, an ability that is missing from previous explainers. In this paper, we contribute to filling these gaps by presenting TimeSHAP, a model-agnostic recurrent explainer that leverages KernelSHAP's sound theoretical footing and strong empirical results. As the input sequence may be arbitrarily long, we further propose a pruning method that is shown to dramatically improve its efficiency in practice.

1 Introduction

Recurrent neural networks (RNN) models, such as LSTMs [1] and GRUs [2], are state-of-the-art in solving numerous sequential decision tasks, from speech recognition [3] to language modelling [4] and time-series classification [5]. However, while the application of RNNs has grown ubiquitous and its performance steadily increased, our understanding of its inner workings is still lacking. In practice, the complex decision-making process in these models is seen as a black-box, creating a tension between accuracy and interpretability.

Understanding the decision-making process of a complex model may be crucial in order to detect and correct flawed reasoning, such as those stemming from spurious correlations in the training data. Models that rely on such spurious correlations, known as "Clever Hans" models[1] [7], may have strong test results but generalize poorly when deployed in the real-world. By explaining the reasoning in a given model, we simultaneously gather insight into how it may be improved and may advance our own human knowledge of the underlying task, as previously unknown patterns are uncovered by the explanation.

[1] Named after an early 20th century horse that was thought to be able to do arithmetic, but was later found to be picking up on behavioral cues from its owner when pointing to the correct answer [6].

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

Additionally, understanding the model's reasoning may be a requirement in certain real-world applications, as exemplified in GDPR's "right to explanation" [8] (although its reach is contested [9, 10]). Just as humans are biased and sometimes discriminatory towards certain groups of the population, so too can *deep learning* (DL) models be [11, 12, 13]. Regulators must be able to peek under the model's hood in order to audit for potential discriminatory reasoning. Although humans may be less accurate, and certainly less scalable than DL models, we can offer some form of after-the-fact reasoning for our decisions. For all the benefits and efficiencies DL has brought about, in order for the community to trust these models, it must be possible to explain their reasoning, at least to a level that humans can understand.

In recent years, numerous methods have been put forth for explaining DL models [14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]. However, RNNs pose a distinct challenge, as their predictions are not only a function of the immediate input sample, but also of the previous input samples and the context (hidden state) drawn thereafter. Blindly applying state-of-the-art DL explainers to RNNs often disregards the importance of the hidden state, distributing all feature importance solely through the features of the current input.

Recently, in landmark work, Lundberg and Lee [25] unified a large number of explanation methods into a single family of "additive feature attribution methods". The authors further proved that there is a unique solution to this task that fulfills three crucial properties of explanations, and dubbed this implementation KernelSHAP. However, no work yet extended this method to explaining timestep-wise RNN predictions. With this in mind, we put forth TimeSHAP, a model-agnostic recurrent explainer suited for tabular sequence learning that leverages KernelSHAP's strong theoretical foundations and empirical results.

TimeSHAP extends the framework put forth by Lundberg and Lee [25] to the recurrent model setting. By doing so, we enable explanation not only of which features are most important to a recurrent model, but also which previous events had the largest impact on a current prediction. As sequences are arbitrarily long, we further propose a pruning method that increases real-world efficiency considerably. We analyze local and global explanations of an RNN using our method, and find multiple instances in which these are crucial for debugging the underlying predictor.

The contributions of TimeSHAP can be summarized as follows:

- explanations of both feature- and event-wise attributions in sequence predictions.
- a new perturbation function suited for the recurrent setting.
- a sequence pruning method that dramatically increases the method's efficiency in practice.

2 Related Work

Research on machine learning model explainers can generally be subdivided into two categories: model-agnostic explainers, and model-specific explainers.

Model specific explainers exploit characteristics of the model's inner workings or architecture to obtain more accurate explanations of its reasoning [26]. The task of explaining RNNs is often tackled by using attention mechanisms [27, 28, 29, 30]. However, whether attention can in fact explain a model's behavior is debatable and a known source of controversy in the ML community [31, 32, 33].

DL models, in which RNNs are included, can also be explained by gradient-based methods. These explainers attribute a weight w_i to each feature, representing the importance or saliency of feature i, based on the partial derivatives of the prediction function $f(x)$ with respect to the input x_i: $w(x_i) = \left|\frac{\partial f(x)}{\partial x_i}\right|$ [14, 15, 16]. Another family of DL explainers is that of layer-wise relevance propagation. In the network's backward pass, starting from the output neuron, the relevance (which initially corresponds to the predicted score) is propagated iteratively from higher layers to lower layers according to some set rules [18, 19, 20, 21]. However, when explaining sequential inputs, DL-specific methods focus on features instead of events, leaving event relevance as a largely unexplored research direction. Regarding RNN-specific explainers [22, 23, 24], these are often inflexible with regards to the model's architecture. For instance, if the RNN is a building block of a larger DL model, preceded/succeeded by other types of layers, it is not the direct input to the RNN we want to explain but the input to the model. Hence, a model agnostic explainer may be better suited to explain these architectures, as real-world models are seldom built only of recurrent layers.

Model agnostic explainers are substantially more flexible, and thus often preferred in real-world applications. These explainers generally rely on post-hoc access to a model's predictions under various settings, such as perturbations of its input [34]. A perturbation h_x of the input vector x, of length m, is the result of converting all values of a coalition of features $z \in \{0,1\}^m$ to the original input space, such that 1 means that a feature takes its original value, and 0 means that a feature i takes some uninformative background value b_i representing its removal. This background value may be, for instance, $b_i = 0$, such as in LIME [35], or the average value for that feature in the dataset, such as in KernelSHAP [25].

$$h_x(z) = \begin{cases} x_i & \text{if } z_i = 1 \\ b_i & \text{if } z_i = 0 \end{cases}, \; i \in \{1, ..., m\} \tag{1}$$

Lundberg and Lee [25] unified this and other explainers (both model-agnostic and model-specific) into a single family of "additive feature attribution methods". Moreover, the authors prove that there is a single solution to this family of methods that fulfills both local accuracy (the explanation model should match the complex model locally), missingness (features that are set to be missing should have no impact on the predictions), and consistency (if a feature's contribution increases then its attributed importance should not decrease).

The authors put forth KernelSHAP, a model-agnostic explainer that fulfills these three properties. KernelSHAP approximates the local behavior of a complex model f with a linear model of feature importance g, such that $g(z) \approx f(h_x(z))$. The task of learning the explanation model g is cast as a cooperative game where a reward ($f(x)$, the score of the original model), must be distributed fairly across the players ($i \in \{1, ..., m\}$, the features). The optimal reward distribution is given by the Shapley values formulation [36]. However, obtaining the true Shapley values for all features would imply generating all possible coalitions of the input, $z \in \{0,1\}^m$, which scales exponentially with m the number of features in the model, $\mathcal{O}(2^m)$. As this task is computationally intractable, KernelSHAP approximates the true values by randomly sampling feature coalitions [37]. The authors further show that a single coalition weighing kernel, $\pi_x(z)$, and a single loss metric, $L(f, g, \pi_x)$, lead to optimal approximations of the Shapley values:

$$\pi_x = \frac{(m-1)}{\binom{m}{|z|} |z| (m - |z|)} \tag{2}$$

$$L(f, g, \pi_x) = \sum_{z \in \{0,1\}^m} \left[f(h_x(z)) - g(z) \right]^2 \cdot \pi_x(z) \tag{3}$$

where $|z|$ is the number of non-zero elements of z, and L is the squared loss used for learning g.

Despite being widely adopted by the ML community, KernelSHAP cannot be applied out-of-the-box to recurrent models. Doing so disregards an important part of the model: the hidden state, which carries information on the previous predictions. Hence, the current KernelSHAP framework cannot be applied to recurrent models due to (1) assuming independence between subsequent model predictions, and (2) disregarding the importance of the hidden state $\mathfrak{h}$ as a feature. There have been few approaches to extending this method to a recurrent setting, but with debatable implementations. Ho et al. [38] use KernelSHAP to explain RNN predictions on an ICU mortality dataset [39]. However, their implementation perturbs only the time-step t being explained (violation (1)), and distributes the model's score $f(x^{(t)}, \mathfrak{h}^{(t-1)})$ solely through the input features $x^{(t)}$ (violation (2)).

3 Methodology

Firstly, our goal is to enable explaining sequence models while preserving three desirable properties of importance attribution stemming from the Shapley values: local accuracy, missingness, and consistency [40]. Secondly, as sequence models handle one extra dimension (spanning the sequence of input events), we aim to explain both feature importance and event importance. Finally, our method should be resource-efficient, as calculating the true Shapley values is trivial but computationally intractable. Hence, we put forth TimeSHAP, a model-agnostic recurrent explainer with sound theoretical footing and strong empirical results.

3.1 RNN Preliminaries

Although our method can be used to explain any sequence model, throughout this paper we will use the example of a canonical RNN, as it is both simple and widely used. Other types of recurrent models that could be used include the Long Short-Term Memory [1], the Gated Recurrent Unit [2], or the Conditional Random Field [41]. The predictions f of a sequence model at a given time-step t are a function not only of the current input $x^{(t)}$ but also of its previous inputs $x^{(t-1)}, x^{(t-2)}, ..., x^{(0)}$. For the RNN, this recurrence is achieved indirectly through a hidden state $\mathfrak{h}$ that aims to encode all relevant information from previous time-steps. As such, an RNN's prediction, $\hat{y}^{(t)} = f(x^{(t)}, \mathfrak{h}^{(t-1)})$, is given as follows [42]:

$$a^{(t)} = b + W\mathfrak{h}^{(t-1)} + Ux^{(t)} \tag{4}$$

$$\mathfrak{h}^{(t)} = \sigma(a^{(t)}) \tag{5}$$

$$o^{(t)} = c + V\mathfrak{h}^{(t)} \tag{6}$$

$$\hat{y}^{(t)} = \text{softmax}(o^{(t)}) \tag{7}$$

where b, c are learnable bias vectors, U, V, W are learnable weight matrices, and σ is a nonlinearity (often chosen to be the hyperbolic tangent). Moreover, $a^{(t)}$ is known as the activation, $\mathfrak{h}^{(t)}$ is the hidden state, $o^{(t)}$ is the output, and $\hat{y}$ is a vector of probabilities over the output. If the RNN is the last layer of the model, then $\hat{y}^{(t)}$ may be used directly as the prediction. Otherwise, $o^{(t)}$ is passed to the following layers, and the prediction is given by the last layer in the forward-pass.

3.2 TimeSHAP

TimeSHAP builds on top of KernelSHAP [25], a state-of-the-art model agnostic explainer, and extends it to work on sequential data. Hence, our method attributes importance to each feature in the input, reflecting the degree to which each feature affected the final prediction. Alternatively, TimeSHAP can explain the sequence (or temporal) dimension, attributing importance to each event (current or past). In order to explain a sequential input, $X \in \mathbb{R}^{l \times d}$ with d events and l features per event, our method fits a linear explainer g that approximates the local behavior of a complex explainer f by minimizing the loss L given by Equation 3. As events are simply features in a temporal dimension, and the algorithm for explaining features $x \in \mathbb{R}^{1 \times d}$ and events $x \in \mathbb{R}^{l \times 1}$ is conceptually equal, we will hence use the word feature to mean both rows and columns of $X \in \mathbb{R}^{l \times d}$. Thus, the formula for g is as follows:

$$f(h_X(z)) = g(z) = w_0 + \sum_{i=1}^{m} w_i \cdot z_i \tag{8}$$

where the bias term $w_0 = f(h_X(\mathbf{0}))$ corresponds to the model's output with all features toggled off (dubbed *base score*), the weights $w_i, i \in \{1, ..., m\}$ correspond to the importance of each feature, and either $m = l$ or $m = d$ depending on which dimension g is explaining. The perturbation function h_X maps a coalition $z \in \{0, 1\}^m$ to the original input space following Equation 1. A feature's background value b_i is given by the its average value on the train dataset. Crucially, the sum of all feature importances corresponds to the difference between the model's score $f(X)$ and the base score of an uninformative input $f(h_X(\mathbf{0}))$.

Input perturbations are generated differently depending on which dimension is being explained. For a given input $X \in \mathbb{R}^{l \times d}$, when perturbing a feature i with its background value b_i, we set all cells $x_{i,j} = b_i, j \in \{1, ..., d\}$. Thus, the feature i is essentially toggled off for all events of the sequence. Note that current KernelSHAP implementations perturb only the current event's features, clearly ignoring the recurrency in Equation 4. On the other hand, when perturbing an event j, we set all cells $x_{i,j} = b_i, i \in \{1, ..., l\}$, toggling off all features of that single event. This change in the perturbation function h_X is the sole implementation difference between explaining events and features. Moreover, the perturbation of X according to a null-vector coalition $z = \mathbf{0}$ is the same regardless of which dimension is being perturbed. In practise, the perturbation of a null-vector coalition $h_X(\mathbf{0})$ is equal to

a matrix $X' \in \mathbb{R}^{l \times d}$ in which all cells are set to the average value of that feature on the train dataset, $x_{i,j} = b_i, i \in \{1, .., l\}, j \in \{1, .., d\}$.

3.2.1 Pruning

Algorithm 1 Sequence Pruning

Input: sequence to prune X,
model to explain f,
tolerance η,

$true_score \leftarrow f(X)$ ▷ Compute true sequence score
$error_tol \leftarrow true_score * \eta$ ▷ Compute error tolerance
for $i \in \{seq_len, seq_len - 1, ..., 1\}$ **do** ▷ Starting from the end of the sequence.
  $truncated_X \leftarrow X[i : seq_len]$ ▷ Truncate to most recent events
  $truncated_score \leftarrow f(truncated_X)$ ▷ Compute truncated sequence score
  $score_delta \leftarrow truncated_score - true_score$ ▷ Distance from target score
  if $|score_delta| < error_tol$ **then** ▷ If heuristic is fulfilled
    return $truncated_X$
return X

One glaring issue with both TimeSHAP and the original KernelSHAP is that the number of coalitions scales exponentially with the length of the sequence being explained. Moreover, in a recurrent setting, the input sequence can be arbitrarily large. To address this we propose a sequence pruning algorithm.

It is common for real-world events to be preceded by a long history of past events (e.g., the whole transaction history of a client), only a few of which are relevant to the current prediction. Additionally, recurrent models are known to seldom encode information from events in the distant past (e.g., the vanishing gradient problem [42]). Based on this insight, we prune a sequence to its most recent events such that the predicted score is approximately equal to the predicted score of the unpruned sequence. In case this smaller sequence is found, it will serve as proxy for the original sequence on the explanation model.

The pruning method, defined in Algorithm 1, receives three parameters: the original sequence X, the model being explained f, and a user defined tolerance η. Hence, the method prunes a sequence X to the smallest sequence $X' \subseteq X$ whose score fulfills $|f(X) - f(X')| < \eta \cdot f(X)$. Therefore, if an event in the distant past is critical to the current prediction, than it will be included in the explanations model; otherwise, we can use only recent events to explain the current prediction. We will empirically show in Section 4.1 that distant events seldom affect the model's score.

3.2.2 Algorithm

Algorithm 2 TimeSHAP

Input: sequence to explain X,
model to explain f,
pruning tolerance η,
background instance X',
number of coalitions to sample $n_samples$,

$trunc_X \leftarrow sequence_pruning(X, f, \eta)$ ▷ Apply pruning to the sequence
$event_exps \leftarrow Seq_KernelSHAP(trunc_X^{\mathrm{T}}, f, X', n_samples)$ ▷ Event explanations
$feature_exps \leftarrow Seq_KernelSHAP(trunc_X, f, X', n_samples)$ ▷ Feature explanations
return $event_exps, feature_exps$

TimeSHAP's objective is to answer two questions: (1) *Which events contributed the most to the current prediction?* and (2) *Which features contributed the most to the current prediction?* Hence, our method first (1) prunes the original sequence (described in Section 3.2.1), secondly (2) uses our adapted KernelSHAP to explain feature-wise importance, and finally (3) uses our adapted KernelSHAP to explain event-wise importance. TimeSHAP is described in Algorithm 2. Notedly, the main differences between event- and feature-wise explanations can be ascribed to transposing

Table 1: Pruning method analysis

	original	$\eta = 0.1$	$\eta = 0.05$	$\eta = 0.01$	$\eta = 0$
Average Seq. Length	178.59	13.734	25.04	65.46	178.59
Median Seq. Length	134.50	4	10.00	36.50	134.50
Max Seq. Length	1745.00	387.00	456.00	1009.00	1745.00

the input sequence X^T, as instead of explaining along the input's rows we aim to explain along its columns.

With both event level explanations $A^{1\times d}$, and feature level explanations $B^{l\times 1}$, we can essentially create an explanation grid $W^{l\times d}$ such that $w_{i,j} = a_j \cdot b_i, i \in \{1, ..., l\}, j \in \{1, ..., d\}$. Both A, B, and W can be used by data scientists and data analysts alike to explain a prediction and debug the underlying model. Domain experts can compare these explanations with their knowledge to assess whether a given prediction is expected, and whether the model's reasoning is sound.

4 Experiments and Results

To validate our method we train a recurrent deep learning model on a large-scale real-world banking dataset. The model is composed of an embedding layer for categorical variables, followed by an RNN layer, and subsequently followed by a Feed Forward layer. The dataset is composed of approximately 20M events, aggregated by bank accounts. Our objective is to predict account takeover fraud. The data is tabular, and is comprised of each client's history of events, from geographical and demographical data, to information on the client's interactions with the bank The sequences can be composed of three types of events: transactions, logins, and enrollments.

We run TimeSHAP on 1K randomly chosen sequences that were predicted positive by the model. We choose the following hyperparameters: $n_samples = 32K$, $\eta = 0.05$. Our method's execution scales linearly with $n_samples$, $\mathcal{O}(n_samples)$. While η does not impact performance directly, it does impact accuracy of results, as longer sequences (lower η) would need a higher number of coalitions to accurately approximate their Shapley values. We empirically found this choice of values to be a good compromise between resource-efficiency and accuracy of results.

4.1 Pruning method results

To evaluate and benchmark the pruning algorithm, we run TimeSHAP on 1K positively-predicted instances, pruned with varying error tolerance levels. Table 1 shows information on the pruned sequences and the original sequences. We note that the median length of the unpruned sequences is 134.5, leading to a median number of coalitions of $2^1 34.5$. Estimating these Shapley values with $32K$ sampled coalitions would be highly inaccurate, as it represents approximately $1^{-34}\%$ of the universe of coalitions. Hence, pruning is not only resource-efficient but also a necessary step in order to achieve accurate results. When using $\eta = 0.01$, the median sequence length is 36.5, and thus $32K$ coalitions represent on the order of $1^{-5}\%$ of the median universe of coalitions. When using $\eta = 0.05$, the median sequence length is 10, thus TimeSHAP can exhaustively explore all coalitions of the median sequence. Hence, we choose $\eta = 0.05$ as our pruning error tolerance.

4.2 Local explanations

We analyze TimeSHAP's local explanations on two predicted-positive randomly chosen sequences, hence labeled A and B. Sequence A has a model score of $f(A) = 0.97$, and a total length of 97 events. After applying Algorithm 1, this sequence is pruned to 8 events, A', leading to a model score of $f(A') = 0.94$. Sequence B has a model score of $f(B) = 0.71$, and a total length of 85 events. This sequence is pruned to the last 5 events, B', leading to a model score of $f(B') = 0.68$.

Figures 1 and 2 show, respectively, event and feature explanations for Sequence A. When analysing the explanations for sequence A, we conclude that the most relevant events are 89, 90, and 91, followed by 5 events with small contributions. Importantly, for this sequence *the most relevant event is not the current input*. We find that the most relevant event is an enrollment, which in our setting

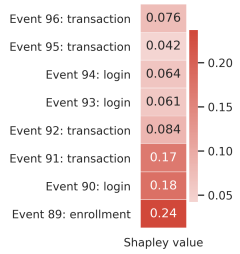

Figure 1: Event explanations for sequence A.

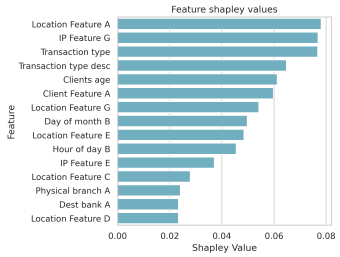

Figure 2: Feature explanations for sequence A.

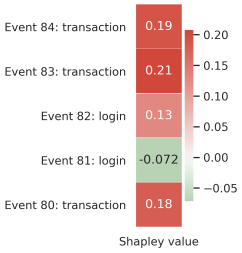

Figure 3: Event explanations for sequence B.

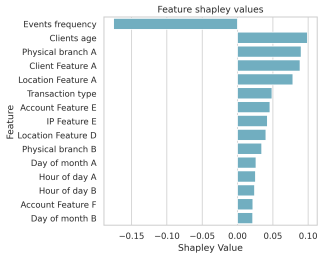

Figure 4: Feature explanations for sequence B.

represents an event associated with account setting behaviour, e.g., changing passwords or logging into a new device. This event is followed in importance by the two subsequent events, which are a login and a transaction. This sequence of events is notoriously suspicious to domain experts, correctly triggering an alert from the model. Regarding the most relevant features, these are both related to location (both geographical and inferred from IP). When analyzing the raw data seen by the model, we can see that an enrollment took place at a new location, correctly alerting the model to clearly suspicious behavior.

Figures 3 and 4 show, respectively, event and feature explanations for Sequence B. TimeSHAP indicates that the most relevant events are 80, 83, and 84, which are all transactions; followed by event 82 which is a login. Regarding feature explanations, the most relevant feature (event frequency) has a negative contribution, followed by strong positive contributions from the client's age and which physical bank branch the account was opened. In this setting, a high frequency of events is strongly linked to fraudulent activity. When inspecting the raw feature data, we see that frequency in Sequence B is exceedingly low, justifying its strong negative contribution to the model's prediction. When inspecting the next three relevant features, these are features that do not vary throughout the sequence which might indicate that this account might inherently have a high fraud risk. Upon inspecting the raw data, we see that events 80 and 81 have similar location data, but different from events 82 and 83, prompting the model to conclude that a location change took place. As seen from the event importance attribution, events 82 and 83 have a strong positive contribution to the predicted score.

4.3 Global explanations

Supplying a data scientist with global explanations can enable an overview of the model's decision process, revealing which features, or events in the case of TimeSHAP, are relevant and which aren't. These explanations are obtained by explaining N sequences, which on these results $N = 1000$, and drawing conclusions from aggregations and visualizations of the individual Shapley values.

Figure 5 shows the distribution of event Shapley values, represented on the y-axis, over the sequence index, represented in the x-axis. Index 0 corresponds to the explained event and negative indices correspond past events of the sequence. This data is the output of running TimeSHAP on 1000 randomly sampled predicted positive sequences.

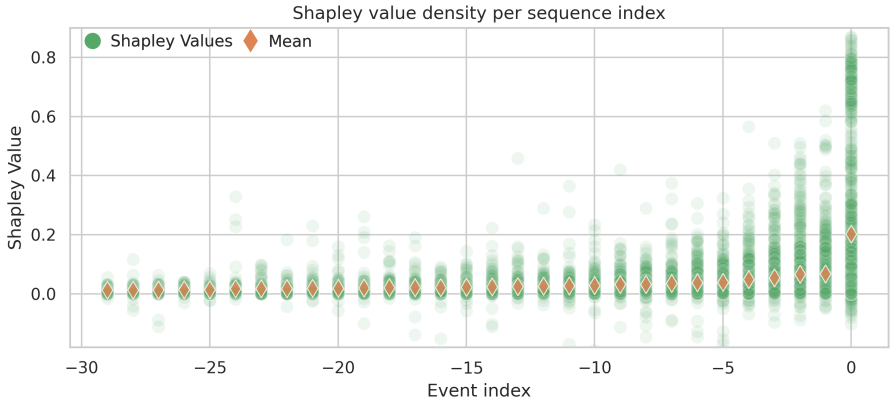

Figure 5: Event Shapley values density over the sequence index

When analysing the distribution of Shapley values over the sequence, we conclude that recent events have a markedly higher importance. At the same time, events at index 0 have significantly higher average attribution values than events at lower indices.

5 Conclusion

While considerable effort has been guided towards explaining deep learning models, recurrent models have received comparatively little attention. With this in mind, we present TimeSHAP, a model-agnostic recurrent explainer that leverages KernelSHAP's sound theoretical footing and strong empirical results. TimeSHAP carries three main contributions: (1) a new perturbation function suited for the recurrent setting, (2) explanations of both feature- and event-wise importance, and (3) a sequence pruning method that dramatically increases the method's efficiency in practise.

Hence, we enable data scientists to see which features were most important for a model's prediction, but also which past events impact the current prediction the most. We apply our method on a large-scale real-world dataset on account takeover fraud. We notice potentially discriminatory reasoning based on the client's age, as it is shown to increase the model's fraud score on several occasions. Overall, we find multiple instances whose explanations are crucial for debugging the underlying predictor.

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[2] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Stroudsburg, PA, USA, 2014. Association for Computational Linguistics.

[3] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, Jie Chen, Jingdong Chen, Zhijie Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Ke Ding, Niandong Du, Erich Elsen, Jesse Engel, Weiwei Fang, Linxi Fan, Christopher Fougner, Liang Gao, Caixia Gong, Awni Hannun, Tony Han, Lappi Johannes, Bing Jiang, Cai Ju, Billy Jun, Patrick LeGresley, Libby Lin, Junjie Liu, Yang Liu, Weigao Li, Xiangang Li, Dongpeng Ma, Sharan Narang, Andrew Ng, Sherjil Ozair, Yiping Peng, Ryan Prenger, Sheng Qian, Zongfeng Quan, Jonathan Raiman, Vinay Rao, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Kavya Srinet, Anuroop Sriram, Haiyuan Tang, Liliang Tang, Chong Wang, Jidong Wang, Kaifu Wang, Yi Wang, Zhijian Wang, Zhiqian Wang, Shuang Wu, Likai Wei, Bo Xiao, Wen Xie, Yan Xie, Dani Yogatama, Bin Yuan, Jun Zhan, and Zhenyao Zhu. Deep speech 2 : End-to-end speech

recognition in english and mandarin. volume 48 of *Proceedings of Machine Learning Research*, pages 173–182, New York, New York, USA, 20–22 Jun 2016. PMLR.

[4] Gábor Melis, Tomáš Kočiský, and Phil Blunsom. Mogrifier lstm. In *International Conference on Learning Representations*, 2020.

[5] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. Multivariate LSTM-FCNs for time series classification. *Neural Networks*, 116:237–245, aug 2019.

[6] Oskar Pfungst and Carl Leo Rahn. *Clever Hans (the horse of Mr. Von Osten) a contribution to experimental animal and human psychology,*. H. Holt and company, New York,, 1911.

[7] Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Unmasking Clever Hans predictors and assessing what machines really learn. *Nature Communications*, 10(1):1096, dec 2019.

[8] General Data Protection Regulation. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46. *Official Journal of the European Union (OJ)*, 59(1-88):294, 2016.

[9] Bryce Goodman and Seth Flaxman. European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation". *AI Magazine*, 38(3):50–57, oct 2017.

[10] Andrew Selbst and Julia Powles. "meaningful information" and the right to explanation. In *Conference on Fairness, Accountability and Transparency*, pages 48–48. PMLR, 2018.

[11] Julia Angwin, Jeff Larson, Lauren Kirchner, and Surya Mattu. Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. `https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing`, May 2016. Accessed: 2020-01-09.

[12] Robert Bartlett, Adair Morse, Richard Stanton, and Nancy Wallace. Consumer-Lending Discrimination in the FinTech Era. Technical report, National Bureau of Economic Research, 2019.

[13] Joy Buolamwini and Timnit Gebru. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In Sorelle A Friedler and Christo Wilson, editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 77–91, New York, NY, USA, 2018. PMLR.

[14] K Simonyan, A Vedaldi, and A Zisserman. Deep inside convolutional networks: visualising image classification models and saliency maps. pages 1–8. ICLR, 2014.

[15] Misha Denil, Alban Demiraj, and Nando De Freitas. Extraction of salient sentences from labelled documents. *arXiv preprint arXiv:1412.6815*, 2014.

[16] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, California, June 2016. Association for Computational Linguistics.

[17] Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*, 2016.

[18] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211 – 222, 2017.

[19] S. Lapuschkin, A. Binder, G. Montavon, K. Müller, and W. Samek. Analyzing classifiers: Fisher vectors and deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2912–2920, 2016.

[20] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):1–46, 07 2015.

[21] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153, 2017.

[22] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):667–676, 2018.

[23] W. James Murdoch and Arthur Szlam. Automatic rule extraction from long short term memory networks, 2017.

[24] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks, 2015.

[25] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[26] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-Agnostic Interpretability of Machine Learning. *ICML Workshop on Human Interpretability in Machine Learning (WHI 2016)*, jun 2016.

[27] Aya Abdelsalam Ismail, Mohamed Gunady, Luiz Pessoa, Hector Corrada Bravo, and Soheil Feizi. Input-cell attention reduces vanishing saliency of recurrent neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 10814–10824. Curran Associates, Inc., 2019.

[28] Jinghe Zhang, Kamran Kowsari, James H. Harrison, Jennifer M. Lobo, and Laura E. Barnes. Patient2vec: A personalized interpretable deep representation of the longitudinal electronic health record. *IEEE Access*, 6:65333–65346, 2018.

[29] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3504–3512. Curran Associates, Inc., 2016.

[30] Ying Sha and May D. Wang. Interpretable predictions of clinical outcomes with an attention-based recurrent neural network. In *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology,and Health Informatics*, ACM-BCB '17, page 233–240, New York, NY, USA, 2017. Association for Computing Machinery.

[31] Sofia Serrano and Noah A. Smith. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy, July 2019. Association for Computational Linguistics.

[32] Sarthak Jain and Byron C. Wallace. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[33] Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China, November 2019. Association for Computational Linguistics.

[34] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5), August 2018.

[35] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery.

[36] H P Young. Monotonic solutions of cooperative games. *International Journal of Game Theory*, 14(2):65–72, jun 1985.

[37] Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41(3):647–665, dec 2014.

[38] Long V. Ho, Melissa D. Aczon, David Ledbetter, and Randall Wetzel. Interpreting a recurrent neural network's predictions of icu mortality risk, 2020.

[39] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

[40] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[41] John D Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann Publishers Inc., 2001.

[42] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.