Framework for Interaction Between Expert Users and Machine Learning Systems

Saveli Goldberg[1], Boris Galitsky[2], Ben Weisburd[3]

[1] Massachusetts General Hospital. Boston MA 02114, USA. savelig@gmail.com

[2] Oracle Corp. Redwood Shores CA 94065, USA. bgalitsky@hotmail.com

[3] Broad Institute. Cambridge MA 02142, USA. ben.weisburd@gmail.com

Abstract

We propose an approach to decision support systems (DSS) that starts with the user first making their own unassisted decision α_U and providing this as an input to the algorithm. Then, if the algorithm disagrees with the user's initial decision, it iteratively works with the user to converge on a common decision or at least make the user reconsider input values that are inconsistent with α_U. We also suggest a way in which the DSS can explain the machine learning (ML) algorithm's decision to the user. We provide a detailed description of this approach along with examples, and then discuss potential benefits and limitations.

INTRODUCTION

With rapid progress across a broad range of machine learning applications in recent years, some implications of these advances are also causing concern. One set of issues that may arise as people increasingly rely on these systems is that they diminish the users' sense of responsibility for decisions and outcomes, and that by reducing the need for human expertise, they gradually lead to a loss of human expertise in certain important areas. Current approaches to addressing these issues focus on improving the explainability of decisions generated by ML algorithms or by requiring that humans confirm or approve ML decisions (Goodman and Flaxman, 2017). These measures are indeed very helpful, but explainability remains challenging, particularly for complex ML models. Additionally, the better ML systems become, the more likely it will be that users will stop putting much effort into analyzing or critically evaluating the algorithms' decisions, even if automated explanations are also provided.

Here we propose an approach that introduces a meta-agent or decision support system (DSS) between the user and the ML algorithm. This DSS restructures the interaction between a user and the ML in order to mitigate the potential loss of expertise and restore a fuller sense of responsibility to users.

Central to this is that it requires the user to first make an unassisted decision and provide this as an input to the algorithm before the algorithm generates its own automated decision. The DSS is trying to find "weaknesses" in the decision of the user, which may be, in particular, a result of the user's "cognitive bias" (Scott, 1993). If the user's decision continues to differ from the decision of the ML, the DSS helps the user identify the reasons. Figure 1 shows the schematic diagram of the DSS.

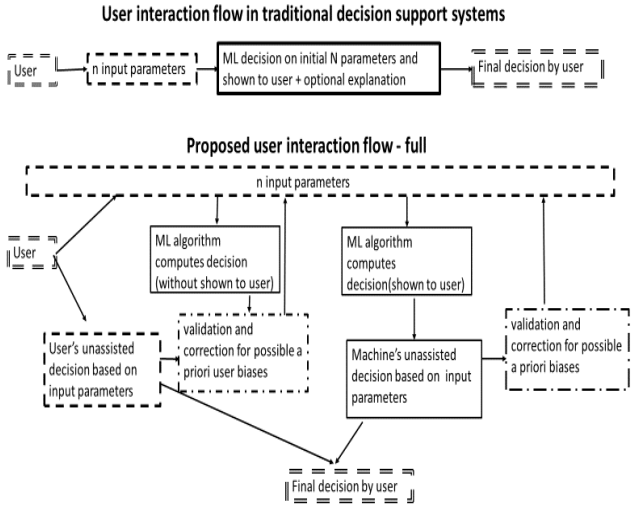

Figure 1. Traditional vs. proposed user interaction flow

EXAMPLE

As a simple illustrative example, let's say a hiker notices a wolf in the distance. She takes a picture of the animal or takes some notes, and later wants to double-check that the animal was indeed a wolf, rather than a dog or a coyote. Image recognition algorithms are unlikely to be helpful in this case since dogs, wolves and coyotes are similar, especially when seen from far away. Instead we propose a system where the user can enter key features such as size, fur color, posture, approximate tail length, etc. into a DSS and iteratively converge on the solution.

Step 1 (User enters input values):
Length = 110 cm, with error bounds [90 – 130]
Color = light grey, with error bounds [white … grey]
Height = 70 cm, with error bounds [55 – 85]
Speed = 60 km/h, with error bounds [45 – 75]
Tail.length = long, with error bounds [long or average]
Tail.direction = down, with error bounds [down or horizontal]

Step 2 (User enters their initial decision):
user thinks that it's a wolf

Step 3 (User enters what parameters were most important to their decision):
Length, Color, Height, Tail.direction

Step 4 (DSS uses grid search to perturb parameters within error bounds to test the stability of the user's decision):
DSS: If input values are passed to the **ML** without any changes, **ML** => wolf
DSS: If Length is changed from 110 cm → 100 cm and Height from 70 cm → 55 cm, **ML** => coyote
DSS: If Tail.direction is changed to horizontal,
ML => dog

Step 5 (DSS finds that a small perturbation of the input values within their error bounds causes the ML's decision to switch from "wolf" to "dog"):
DSS asks user if she's sure about the value of Tail.direction
User: Reconsiders Tail.direction and changes it to "horizontal", but also begins to doubt her initial value for Tail.length and decides to also change this to Tail.length = average. Despite this, the user maintains their initial conclusion that this is a wolf.

The user enters the modified input values into the DSS:
Tail.direction = horizontal, with error bounds [down, horizontal, or up]
Tail.length = average, with error bounds [short, average, or long]

The change in input values changes the ML decision:
ML => dog

Step 6 (DSS explains ML decision):
ML => dog. **DSS** identifies the following inputs as key distinguishing features in the pair-wise difference between dog and wolf:
Tail.direction = horizontal
Speed = 60 km/h

Step 7 (User evaluates the ML decision):
User: what if Tail.direction = down?
ML=> dog. **DSS** identifies the following inputs as key distinguishing features in the pair-wise difference between dog and wolf:
Speed = 60 km/h
Tail.length = average
User: What if both Tail.direction = down and speed = 45 km/h?
ML => wolf. Here the ML's decision matches the user's decision so it's unnecessary for the DSS to explain this decision.

Step 8 (Final User decision).
Now the user can make a final judgement.

METHODS

Let $\mathbf{x} = [x_1, x_2, .., x_n]$ be the n input parameters to the ML algorithm. x_i can be continuous (numerical) or categorical variables. Let $\mathbf{X}$ be a set of $\mathbf{x}$. Let $\mathbf{v} = [v_1, \ldots, v_n]$ be the particular input values entered by the user.

Let $D = \{\alpha_j\}$, $j=1,\ldots, k$ be the set of k possible decisions or output classes.

Let $\alpha_U \in D$ be the initial unassisted decision of the user.

Additionally we allow the user to mark a subset of input parameters $[v_1, \ldots, v_m]$ $m \leq n$ as being particularly important to their decision α_U

We define the ML algorithm's decision function f as a black box which maps an input vector $\mathbf{v}$ and a decision $\alpha \in D$ to interval [0, 1]:

$f(\alpha,\mathbf{x})$: $\alpha,\mathbf{x} \rightarrow [0, 1]$.

Let α_{ML} be the ML decision based on the user-provided input values v.

$f(\alpha_{ML},\mathbf{v}) = \max(f(\alpha,\mathbf{v}))$ for all $\alpha \in D$

For any parameter of $\mathbf{x}$, its value x_i may have bias or error so we define $\Omega(x_i)$ such that $\Omega(x_i) > \Omega^{lower}(x_i)$ & $\Omega(x_i) < \Omega^{upper}(x_i)$ as the set of values which are considered within the error bounds for x_i. Ω includes both measurement error as well as user bias - for example "confirmation bias" (the tendency to search for, interpret, focus on and remember information in a way that confirms one's preconceptions) .

We introduce a feature normalization $x_i^{norm}(\alpha)$ for each i-th dimension and class α, set based on the following five thresholds: $th_{0i}(\alpha)$, $th_{1i}(\alpha)$, $th_{2i}(\alpha)$, $th_{3i}(\alpha)$, $th_{4i}(\alpha)$ (Goldberg, 2007).

$x_i < th_{0i}(\alpha)$: strong_deviation
$x_i^{new}(\alpha) = 0 + x_i / th_{0i}(\alpha)$

$th_{1i}(\alpha) < x_i < th_{2i}(\alpha)$: abnormal
$x_i^{new}(\alpha) = 1 + (x_i - th_{1i}(\alpha))/(th_{2i}(\alpha) - th_{1i}(\alpha))$

$th_{2i}(\alpha) < x_i < th_{3i}(\alpha)$: normal
$x_i^{new}(\alpha) = 2 + (x_i - th_{2i}(\alpha))/(th_{3i} - th_{2i}(\alpha))$

$th_{3i}(\alpha) < x_i < th_{4i}(\alpha)$: abnormal
$x_i^{new}(\alpha) = 3 + (x_i - th_{3i}(\alpha))/(th_{4i} - th_{3i}(\alpha))$

$th_{4i}(\alpha) < x_i$: strong_deviation
$x_i^{new}(\alpha) = 4 + x_i /(th_{4i}(\alpha))$

The normalization can be defined for categorical parameters also. For example, for wolves, normal colors are gray, light gray or dark gray = 2. White and black are considered abnormal so white = 1 and black = 3, and all other colors would

be considered strong deviation = 4. We expect that, when implementing a DSS based on this approach, the thresholds will be provided by domain experts using empirically established knowledge of what values of the input parameters are normal or abnormal for a given decision class α.

Based on this definition, we can define a mapping between the input parameters **X** and the normalized parameters $\mathbf{X}^{norm}$: $\mathbf{X} \rightarrow \mathbf{X}^{norm}$ and $\mathbf{X}^{norm} \rightarrow \mathbf{X}$.

We can then use the vector distance $||\mathbf{x} - \mathbf{y}||$ in the input parameter space or in the normalized space to measure similarity between input parameters.

Algorithm for Step 4: Stability assessment

In this step the DSS checks whether α_{ml} is stable when the input parameters are perturbed within the error bounds $\Omega^{lower}(v_i)$ $\Omega^{upper}(v_i)$. If, when entering the input values, the user also marked a subset of input parameters (v_1 .. v_m) as particularly important to their decision α_U, then the DSS only perturbs this subset because, given the user's focus on these parameters, they are the ones more likely to contain user bias.

As DSS uses grid search to perturb one or more of the input parameters, it passes these perturbed inputs **v'** to the ML and checks – if α_{ML}' differs from the original α_{ML}, then it uses **v'** in step 5. If more than one α_{ML}' is found, then the DSS uses the α_{ML}' for which **v'** differs from the original inputs **v** in the fewest dimensions (eg. where the smallest number of input parameters needed to be perturbed). If, on the other hand, the search does not identify any points **v'**, and α_{ml} remains stable, then step 5 is skipped.

Algorithm for Step 5: Discovering "suspicious" parameters

The DSS asks the user to reconsider the input values of the input parameters for which **v'** differs from **v**. The user may then realize that these input values imply a different α_U and change their initial α_U to a different α_U'. Alternatively, if input values have a subjective component or contain errors or bias, the user may adjust the input values. In either case, if changes are made, the DSS goes back to step 4 with the new values but does this no more than 3 times to avoid endless iteration.

Algorithm for Step 6: Explainability of ML

If at this point the user's decision still differs from the ML's decision, the DSS attempts to explain the difference between the ML decision α_{ml} and the user decision α_U in a way that is intuitive for a human user rather than a way that's based on the ML's internal representation. To do this, the DSS determines what input parameters were most important for the ML's decision. This can be done by finding the input vector **z** which is closest to the user's input values **v** and which leads the ML to change its decision from α_{ml} to α_U. A crucial part of this step is that the distance between points **v** and **z'** is computed in normalized parameter space ($\mathbf{X}^{norm}(\alpha_{ML})$). The DSS can use a grid search in normalized parameter space to find points on the boundary between α_{ML} and α_U (Figure 2). Once **z** is found, the parameters that have the largest one-dimensional distance between **z'** and **v** are taken as the parameters that are most important to explaining the difference between α_{ml} and α_U.

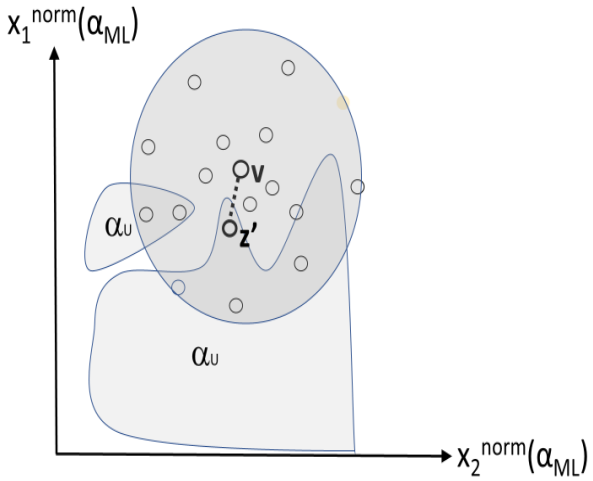

Figure 2. Finding **z'**

Here is the user interaction flow:

1st Step:
User input1: $\mathbf{v} = [v_1, \ldots, v_n] \in \mathbf{X}$

2nd Step:
Initial unassisted decision α_U of the user

3nd Step:
User indicates m out of n input values as being particularly important to his decision α_U $[v_{1,\ldots} v_m]$ m<<n

4th Step:
Now ML verified the decisions of user α_U without sharing α_{ML}

In order to determine how stable α_U is relatively to perturbations of **v** within error bounds Ω, we compute α_{ML} by means of Stability Assessment Algorithm.

If α_{ML} does not match α_U, go to Step 5.

If α_{ML} matches α_U, then α_U is selected as a preliminary solution, and we proceed to Step 6.

5th Step:

Since $\alpha_U \neq \alpha_{ML}$ iteratively work with the user to see if we can converge on a stable decision. We apply Discovering suspicious parameters Algorithm.

We could, at this point, just show α_{ML} to the user, but we specifically avoid doing this in order to prevent the user from unthinkingly changing their decision to α_{ML}. Instead we use a more nuanced, indirect approach where we try to find the parameter whose value v_i, from the ones indicated by the user to be in the set proving α_U, v_i, is such that its possible deviation affects α_U in the highest degree.

After finding this parameter, we report to the user that the value they provided for this parameter is to some degree inconsistent with α_U. We then give the user the option to change their initial α_U

If the user maintains the same decision α_U, α_U is set as a preliminary decision and we proceed go to Step 6

If user changes their decision, go to Step 2 (unless this point is reached a third time, in which case go to Step 6 to avoid an overly long interaction loop).

6th Step:

Compute decision α_{ml} based on unchanged input values **v**. α_{ml} is set as a decision of ML and is shown to the human expert along with the set of key features which has yielded α_{ML} instead of α_U. Explainability of ML algorithm is in use here.

7th Step:

The human expert can modify *v* and observe respective decisions of ML. ML can in turn change its decision and provide an updated explanation. Once the human expert obtained ML decision for all cases of interest, she obtains the final decision.

Hence in the third step the human explains its decision, and in the sixth step the ML explains its decision. In the fifth step ML assesses the stability of human experts' decision with respect to selected features. In the seventh step the human expert does the same with ML decisions. So, the sixth step is inverse to the third and the seventh is inverse to the fifth.

APPLICATION

Some features of this proposed approach were implemented in the decision support system "Dinar-2" which assisted physicians in establishing the pathology and severity of cases when triaging emergency calls at the Center for Child Air-Ambulance Services in Yekaterinburg, Russia (Goldberg et al. 1991) One of the goals of this Center was to provide remote consultation to regional medical centers and doctors involved in treating seriously ill children, and thereby reduce the need to airlift children to larger or more specialized hospitals.

The Center was responsible for a large geographic area, which meant that air-ambulance services, when dispatched, could still take a long time to reach their destination. Given the volume and complexity of requests for consultation and air-ambulance services, a computerized decision support system was key to the efficient functioning of the Center. Dinar-2 was developed to fill this need. This system provides assistance in diagnosing the type of pathology (8 distinct classes of pathology), and in determining its severity (between 3 and 5 levels of severity - depending on the class). It also assists in selecting the best course of action, and in selecting the healthcare center that's best suited for treating a given patient.

The Dinar-2 decision support algorithm consists of 3 stages:

1. Identification of informative patterns and groups of symptoms
2. Determination of the likely pathologies based on 1.
3. Determination of severity

These steps were implemented using rule-based machine learning algorithms.

Besides objective measurements and test results, the system had to take into account a significant amount of subjective information about the patient's' condition. This made the decision support task more complicated because the subjective information was susceptible to conscious and subconscious biases on the part of the reporting physicians. Specifically, these biases tended to skew the provided information toward making a patient's condition appear either more or less severe than it actually was.

Due to this, the Dinar-2 decision support system assigned an a-priori confidence interval to every input parameter that was based on subjective information. Then, the system perturbed the inputs within the bounds of these confidence intervals, and checked whether it's computed diagnosis was consistent with the diagnosis initially proposed by the user (in this case a physician at the Center, in consultation with the regional doctor).If, under these perturbations, Dinar-2's diagnosis of the pathology or severity did not match that of the user, Dinar-2 would follow the proposed interaction flow (described in section II above) to clarify the diagnosis.

After its initial deployment in 1989, Dinar-2 was soon adopted by 39 air-ambulance centers across Russia, Kazakhstan, and Belarus, and has been in continuous use since then. For example, available statistics for the Yekaterinburg region for 2017 indicate that during that year, the system assisted in evaluating 537 cases. In 131 of these cases (24%), effective remote diagnosis and consultation proved sufficient for resolving the patient's crisis, and the need to dispatch an air-ambulance was avoided (Report Neonatology Department of Sverdlovsk State Children Hospital, 2018)

DISCUSSION

There are a number of benefits and opportunities afforded by the proposed approach. Requiring the user to first reach their own decision serves to counteract the loss of users' expertise and sense of responsibility that often occurs when users delegate decisions to a DSS. It prevents the user from becoming complacent and motivates them to give more thought to their initial decision. It provides continued opportunity for user to revisit and refresh their domain knowledge. When the user and the algorithm don't agree, it forces the user to reconsider their decision in light of parameters highlighted by the algorithm. In the end, it makes it more likely that the user will critically evaluate the machine's decision. In applications where the algorithm is more accurate than human users, this even allows the user to challenge themselves to anticipate the algorithm's answer – either on their own, or explicitly, by adding game-playing elements to the interaction.

Additionally, the DSS elements presented here may be used separately. Approach to explaining the ML decision and the algorithm for evaluating the users initial decision α_U can be used independently from each other. However, for applications where the ML algorithm must discriminate between multiple classes ($k > 2$) and input parameters contain uncertainty, we consider both elements to be necessary.

Explaining an ML classifier's decision while treating the classifier as a black box has been proposed before, for example (Baehrens at al. 2010). However, our approach serves to provide explanations that are more intuitive to the user because it uses the normalization of input parameters to map them into a space that's based on the way experts naturally think about these parameters.

This approach is also relevant in light of the European Union's new General Data Protection Regulation which controls the applicability of machine learning (https://eugdpr.org/). These regulations restrict automated individual decision-making (that is, algorithms that make decisions based on user-level predictors) which "significantly affect" users. The law effectively creates a *right to explanation*, whereby a human user can request an explanation of an algorithmic decision that was made about them.

Our approach has several limitations. The user's interaction with the DSS requires time which may be unavailable, or example in a system that assists with time-sensitive tasks such as operating machinery or driving a car. Additionally, in applications where the ML algorithm is significantly better than the user at accurate classification, the approach becomes unnecessary.

CONCLUSION

In conclusion, there are a number of benefits to structuring decision support systems in a way that makes the user provide their own unassisted decision to a decision support system as a first step. We expect this approach will increase the accuracy of the final decision and will serve to maintain and possibly even improve the expert users' domain knowledge.

.

References

Baehrens D.; Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, Klaus-Robert Müller How to Explain Individual Classification Decisions.; 11(Jun):1803−1831, 2010.

Goldberg S.I.; Lomovskikh V.E.; Makhanek A.O.; Sklyar M.S. 1991, Expert system "DINAR-2".-methodological basis for the pediatric emergency aid organization in a large region. In: *Medical Informatics Europe 1991,* Vienna, Austria, 270-274

Goldberg S.; Nikita Shklovskiy-Kordi.; Boris Zingerman. 2007. Time-oriented multi-image case history - way to the "disease image" analysis. *VISAPP (Special Sessions*) 200-203.

Goodman B.; Flaxman S. 2017, European Union regulations on algorithmic decision-making and a "right to explanation" *AI Mag Magazine*, Vol 38, No 3,

Report Neonatology Department of Sverdlovsk State Children Hospital, Russia, 2018, 43-47

Scott, 1993, *The Psychology of Judgment and Decision Making*