Supplementary Material
Towards Rigorous Interpretations: a Formalisation of Feature Attribution

Darius Afchar [1 2] **Romain Hennequin** [1] **Vincent Guigue** [2]

Supplementary Material

We list some general notations we use throughout this paper:

Symbol	Meaning
$\wedge$	Logical AND
$\mathcal{X}, \mathcal{Y}$	input and output space
X, Y	input and target random variable (r.v.)
x, y	input and target sample
$p_X, p_{Y\|X}$	distribution of X; Y conditional to X
$\mathcal{U}$	uniform distribution
$\mathcal{N}$	multivariate normal distribution
$\mathcal{B}$	Bernoulli distribution
$\vec{e}_i$	i-th canonical vector of $\mathbb{R}^n$
$[n]$	set of integer from 1 to n
I	subset of integer
f, f_i, ...	denotes a function
R, R_i, ...	denotes a binary relation (b.r.)
D	denotes a dataset (hence defines a b.r.)
X_I	r.v. X projected to the input features with indexes I
R_I	b.r. R with its domain projected to I
f_I	function with its domain projected to I
$\mathbb{E}_{Y\|X}$	mean equipped with the distribution of $Y \mid X = x$
δ_A	indicator function of set A

A. fANOVA instance-wise selection failure

We detail our claim that the fANOVA is indicative of a global feature dependence, but not an instance-wise one, with a simple example.

Let us take a binary bidimensional problem with input $X = (X_1, X_2) \in \{0,1\}^2$ following a uniform probability and we try to compute the subset dependence of a AND function f – *i.e.* $Y = f(X) = X_1 \wedge X_2$. The fANOVA gives the unique decomposition:

$$
\begin{aligned}
f_\emptyset = 1/4 = \mu &= \begin{bmatrix} \mu & \mu \\ \mu & \mu \end{bmatrix} \\
f_1 = \begin{bmatrix} -\mu & \mu \end{bmatrix} &= \begin{bmatrix} -\mu & \mu \\ -\mu & \mu \end{bmatrix} \\
f_2 = \begin{bmatrix} -\mu \\ \mu \end{bmatrix} &= \begin{bmatrix} -\mu & -\mu \\ \mu & \mu \end{bmatrix} \\
f_{1,2} = \begin{bmatrix} \mu & -\mu \\ -\mu & \mu \end{bmatrix}
\end{aligned}
$$

For convenience, we have represented the functions as matrices indicating their four possible values for X, where X_1 varies along the columns and X_2 along the rows. We check that the value given by f_1 (*resp.* f_2) only depends on X_1 (*resp.* X_2). Then we check the identifiability constraint that each function other than $f_\emptyset$ is zero-centered, and that we indeed obtain a decomposition of f:

$$
\begin{aligned}
f &= f_\emptyset + f_1 + f_2 + f_{1,2} \\
&= \begin{bmatrix} 0 & 0 \\ 0 & 4\mu \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \\
&= X_1 \wedge X_2
\end{aligned}
$$

Meanwhile, the selection solution is the following:

$$
\text{attr}(f) = \begin{bmatrix} \{\{1\},\{2\}\} & \{\{1\}\} \\ \{\{2\}\} & \{\{1,2\}\} \end{bmatrix}
$$

which can be found by testing all four possible restrictions. For instance, for $p_0 = (0,0)$, we have $p_0 \in A_1(f)$ and $p_0 \in A_2(f)$ as the value of f is constant and equal to zero in their respective complementary directions, and those two solutions are minimal – *i.e.* $p_0 \notin A_\emptyset(f)$ as we have a different 1 value on the complementary direction $\overrightarrow{(1,1)}$.

This solution is not translated from the strong symmetry of the ANOVA. For instance, the bivariate term only appear when $X = (1,1)$, but $f_{1,2}$ is non-null everywhere. The same discussion applies for $X = (0,0)$ for which two solutions co-exist, but this is not obvious only looking at the

[1]Deezer, Paris, France [2]LIPS, Paris, France. Correspondence to: Darius Afchar <research@deezer.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

ANOVA decomposition. As mentioned, this is due to the centering constraint that creates "artifacts" in all subfunctions – we say this from the standpoint of instance-wise attribution.

B. Selector-predictor degenerate selection solutions

We show with a constructive counter-example that selector-predictor methods have degenerate solutions that are optimal relative to their objective function, but with meaningless selected features. Our main point will be that, contrary to the intuition behind selector-predictor, splitting the model into a selector and a predictor does not enable to split the joint prediction and selection task between the two: predictions abilities may percolate to the selector and vice-versa. We show an extreme solution where the full prediction work is done by the selector.

We study the model *L2X* (Chen et al., 2018) for which the selector uses a reparametrisation trick to sample k selections of features. We do not fully detail the method since most of the difficult labor is related to finding a relevant relaxation allowing to train the selector model Sel with its discretised output through a gradient-descent. We skip training and directly study theoretical optimal parameters. The objective function of the model on each target sample (x, y) is the cross-entropy loss $l\left(y; \text{Pred}(x \odot v(x))\right)$, where v is a feature mask sampled for each x with a predefined and fixed number k of non-null values.

During training, sampled v are relaxed to $[0, 1]^n$ and have to explore different values for a given x around logit predictions of the selector $\text{Sel}(x)$. Again, we ignore these details as after the training phase Sel is trained and frozen, v is not longer sampled, it is binary and deterministically mapped from x as the top-k logit values of $\text{Sel}(x)$. For simplicity, we absorb the top-k filtering into Sel and directly denote with $\text{Sel}(x)$ the binary mask of the k selected features. For our selector and predictor model parametrised by θ, we will thus write the expected loss after training $\mathcal{L}(\theta) = \mathbb{E}[l(Y; \text{Pred}_\theta(X \odot \text{Sel}_\theta(X)))]$. We denote the minimal theoretical loss $\hat{\mathcal{L}} = \min_\theta \mathcal{L}(\theta)$.

We can compare this loss with the *unmasked* case where we would simply predict Y without restricting the number of usable input features in the predictor: $\hat{\mathcal{L}}_u = \min_\theta \mathbb{E}[l(Y; \text{Pred}_\theta(X))]$. In general, we have $\hat{\mathcal{L}} \geq \hat{\mathcal{L}}_u$ as the expressive power of the predictor is superior when having access to any order of interaction between variables.

Here, we assume that our task admits a unique instance-wise selection solution everywhere with exactly k features, *i.e.* the parameter k in the selector is well tuned and the problem well-posed. This means that given the ground-truth selection random variable S^*, we have that $\mathcal{L}^* = \min_\theta \mathbb{E}[l(Y; \text{Pred}_\theta(X \odot S^*))]$ will be equal to the optimal loss in the unmasked case $\hat{\mathcal{L}}_u$ as S^* only captures features that are relevant to predict Y. Now, if the selector could approximate S^* with a set of parameters θ^*, we would thus have $\mathcal{L}(\theta^*)$ tends to $\mathcal{L}^*$, and consequently $\mathcal{L}(\theta^*)$ tends to $\hat{\mathcal{L}}_u$. However, we do not have access to S^* and cannot evaluate how good the approximation is, we can only compare $\mathcal{L}(\theta)$ to its theoretical bound $\mathcal{L}^*$, that, when we assume that k is well chosen, is equal to the observable unmasked bound $\hat{\mathcal{L}}_u$. The question we should now ask ourselves is whether having found parameters $\hat{\theta}$ such that $\mathcal{L}(\hat{\theta}) = \hat{\mathcal{L}} = \hat{\mathcal{L}}_u = \mathcal{L}^*$ means that we have $\text{Sel}_{\hat{\theta}}(X) = S^*$?

This is critical as selector-predictor models are evaluated in comparison to their non-input-restricted counterpart as a proxy of their approximation of S^*: it is often shown that there is no significant drop in performances while selecting a minimal number of input features. We now show that there exists many equivalent and optimal solutions θ' such that $\mathcal{L}(\theta') = \mathcal{L}^*$ and $\text{Sel}_{\theta'}(X)$ verifies the constraint of selecting k features but while being nowhere close to S^* or to having any interpretation value.

For that we consider the case of a categorical task, and assume that Y is one-hot encoded as $g(Y) \in \{0, 1\}^C$ onto the C possible classes in $\mathcal{Y}$, where g denotes the one-hot-encoding function. Additionally, we introduce a random permutation σ of $[C]$, and its inverse permutation σ^{-1}. We assume we know the ground-truth value of k. Finally, we assume that the input dimension n is greater than $C + k - 1$, which is quite common, and we denote a padding operator $p : \mathbb{R}^C \mapsto \mathbb{R}^n$ that completes any vector of size C with $k - 1$ ones and $n - (k - 1)$ zeros to fit in $\mathbb{R}^n$.

Now, as in *L2X*, we assume that the predictor and selector are parametrised with two families of neural networks with comparable number of parameters. We denote f_θ a member of the selector neural network family and delete $n - C$ neurons in the output layer to obtain a C-dimensional output. Though f_θ belongs to the selector family, we use it to approximate Y: we denote by $\hat{\theta}_f$ some optimal parameter associated with the optimal loss $\hat{\mathcal{L}}_s = \min_\theta \mathbb{E}[l(g(Y); f_\theta(X))]$[1]. And we have,

$$\hat{\mathcal{L}}_s \simeq \hat{\mathcal{L}}_u$$

The $\simeq$ holds if the selector and predictor families have a comparable expressive power. We have an equality with the default implementation of *L2X*.

Then, we come back to the selector-predictor objective and

[1] We assume that f_θ outputs probability vectors, *e.g.* using a `softmax` in its last layer. Before this expression the one-hot encoding operations $g(Y)$ were eluded in the losses.

the trick is to study the solution given by

$$\text{Sel}_\sigma(x) = p(\sigma(f_{\hat{\theta}_f}(x)))$$
$$\text{Pred}_\sigma(x) = \sigma^{-1}\left(\begin{bmatrix}\delta_{|x_1|>0}\\ \vdots \\ \delta_{|x_C|>0}\end{bmatrix}\right)$$

We check that this solution is indeed part of the parametrised family for the predictor and selector. We have built f_θ to have the right selector architecture, except for $n-C$ missing neurons in its last layer; the composed permutation σ can be crafted by permuting the C output neurons of f_θ; composing by p is done by adding $n-C$ constant neurons in the output layer and we thus obtain the right architecture. As for the predictor, the only new element is the non-null indicator functions on the C first input feature. If we were to assume the activation were the step function H, this would be straight-forward to approximate with three neurons, *e.g.*

$$\delta_{|x_i|>\epsilon} = H(H(x_i \geq \epsilon) + H(-x_i \geq -\epsilon) \geq 2)$$

With *sigmoid* and *ReLU* activations, this can be done using big multiplicative coefficients. Overall, we need $\mathcal{O}(C)$ neurons on two layers to approximate the function Pred_σ. The other $n-C$ input features of the predictor are ignored using zero weights in the neurons parameters. It is reasonable to think that with neural network architectures used in practice, Pred_σ is indeed part of the predictor parametrised family, or can be closely approximated.

We denote the found parameters $\hat{\theta}_\sigma$. This particular solution enables us to have

$$\text{Pred}_{\hat{\theta}_\sigma}(x \odot \text{Sel}_{\hat{\theta}\sigma}(x)) = f_{\hat{\theta}_f}(x)$$
$$\mathcal{L}(\hat{\theta}_\sigma) \simeq \hat{\mathcal{L}}_u$$

In essence, **we are estimating Y in the selector and encoding this information in the selection mask we pass to the predictor**. We check that the found selector returns a binary mask with exactly k non-null components: one in the C first components that encodes the label, $k-1$ in the padding operator for the remaining features.

What about the found selections? We have $C!$ possible optimal set of parameters $\hat{\theta}_\sigma$, all of them maximal according to the *L2X* objective function, with them, all first C features of X can be made equally maximally important for selection, regardless of data. We conclude that the selector solution does not translate any truth about dependence between Y and X. A even more efficient label-passing degenerate case can be obtained by replacing g with a function that encodes labels with binary numbers, only requiring $n > \log_2(C) + k - 1$ instead of $n > C + k - 1$.

INVASE (Yoon et al., 2019) is similar to *L2X* with a Lagrangian penalty instead of constraining to have exactly k non-null selected features; it is similarly prone to degenerate selection solutions, but it goes even further. It must be noticed that we only require to output one non-null component in the selector to pass the true label and have an optimal prediction. This means that with a ground-truth selection cardinality $k > 1$, **our degenerate solutions yield an optimal prediction loss with a lower regularisation penalty than when using $\mathbf{S}^*$**, since S^* may have more than one required features for selection. We have observed such effect in practice: *INVASE* has good prediction performances and returns very sparse selection masks correlated with ground-truth labels and having nothing to do with ground-truth selection.

One way to avoid label-passing issues is to verify the properties 1 and 2 we propose.

C. Tasks generation

In this section we explain in detail how the centroids $(c_1, ...c_m)$, their label y_j and ground-truth selection s_j^* are chosen. The unifying condition of these latter variables is that $s_j^* \subset [n]$ should be the unique subset of minimal cardinality verifying $c_j \in A_{s_j^*}(f)$, with $f = p(y=1|x)$.

C.1. Binary Hypercube

We first propose to study the case of centroids forming the vertices of an hypercube. For that, we choose a subset of indexes $J^k \subset [n]$ and study a set Q^k that contains the vertices coordinates of an hypercube of dimension $|J^k|$ placed in $\mathcal{X}$ with its edges aligned with the canonical vectors $\{\vec{e_i} \mid i \in J^k\}$. Since hypercube graphs are bipartite (Foldes, 1977), we can assign a binary label to each vertex with the nice property that for a given point $x \in Q^k$ and associated label y, each single coordinate change to x to find another point $x' \in Q^k$ will yield a neighbor associated to an opposite label $\bar{y}$ (*i.e.* we can color the graph with labels y and $\bar{y}$). This is illustrated in figure 1, we display a generated distribution $p_{X,Y}$ with centroids defined using two hypercubes: one of dimension 2 (also known as the *XOR* problem) and another of dimension 1 oriented along $\vec{e_1}$. We have added dotted lines to highlight the edges of the considered hypercubes. Therefore, for all $x \in Q^k$, for all $i \in J^k$ we have $x \in B_i(f)$: all points of Q^k have neighbors with *contradicting labels* along the dimensions indexed in J^k. Using the property 2 on hierarchy, for $H \subset [n]$ such that $H \cap J^k \neq \emptyset$, *i.e.* if H contains at least one index of J^k, we have $x \in B_H(f)$. By defining $H' \subsetneq J^k$ and choosing $H = [n] \setminus H'$, we check that $H \cap J^k = J^k \setminus H' \neq \emptyset$, and we obtain $x \in B_H(f)$ thus $x \notin A_{H'}(f)$ for all $H' \subsetneq J^k$. Since the hypercube is defined on the dimensions of J^k, we have $x \in A_{J^k}(f)$ and know that it is no use selecting dimensions outside of J^k. We conclude that J^k is the unique minimal subset verifying the functionality property,

and in general that **any set of centroids forming an hypercube defined on the dimensions indexes J and with labels corresponding to the coloring of the graph will have the unique selection solution J for all its vertices**.

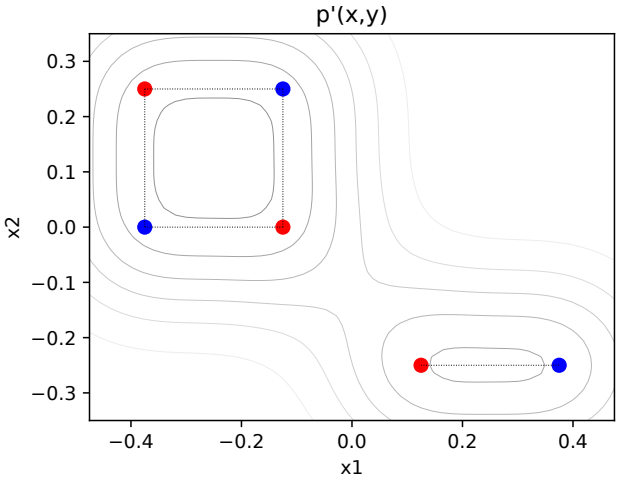

Figure 1. **Distribution of task `2_14`** The centroid c_j are indicated with colored dots – red for $y_j = 1$ and blue for $y_j = 0$. Dotted lines connect centroids with opposite labels and that differ by only one coordinate, *i.e.* that will be superposed if projected on the other coordinate. The corresponding Gaussian mixture for the distribution with imperfect dependence is hinted in black contours.

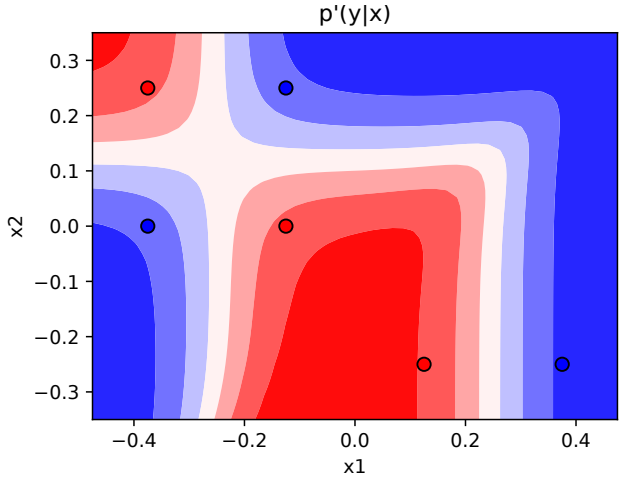

Figure 2. **Optimal mapping for task `2_14`** We plot the conditional probability corresponding to figure 1 with variance $\sigma/2$.

C.2. Hypercubes superposition

We have found a way to create global unique selection solution using hypercubes. We can then superpose several different hypercubes in $\mathcal{X}$. We avoid interactions between hypercubes by storing the coordinates occupied by each hypercube k on each dimensions (*e.g.* in figure 1, the bidimensional hypercubes *occupies* the coordinates $\{-0.375, -0.125\}$ on the axis x_1 and $\{0, 0.25\}$ on axis x_2), and ensuring that others allocate different coordinates (*e.g.* in figure 1, the univariate hypercube *occupies* the coordinates $\{0.125, 0.375\}$ on x_1 and $\{-0.25\}$ on x_2, which does not collide with the other hypercube).

C.3. Centroids minimum relative distance

To create the distribution with imperfect dependencies, we also ensure that all occupied coordinates are equally spaced with a minimum distance σ. Thus, when defining the Gaussian mixture $p'_{X,Y}$ using the centroids as means, we are able to choose the global standard-deviation as a multiple of σ (typically $\sigma/2$) to control the superposition ratio of the Gaussian distributions. With our previous example, the obtained optimal mapping f is shown in figure 2.

C.4. Hypercube erosion

Lastly, we wanted to create more diversity and break the global ground-truth selection within each hypercube. For that, we randomly erase some centroids in each hypercube with a fixed probability P_e. For each centroids c_j we denote the set of its remaining hypercube neighbors $\mathcal{N}_j$. Note that for all $c \in \mathcal{N}_j$, c differs from c_j by only one coordinate and has an opposite label. Within its hypercube k, c_j has its neighbors located on the dimensions $\mathcal{J}_j = \{i \mid \exists c \in \mathcal{N}_j, P_i c_j \neq P_i c\}$. By construction, $\mathcal{J}_j \subset J^k$. Then, by the same reasoning as before, we know that for all $i \in \mathcal{J}_j$, $c_j \in B_i(f)$ and that $c_j \in A_{\mathcal{J}_j}(f)$; and thus deduce that $\mathcal{J}_j$ is the minimal dependence subset for c_j and hence its selection solution s^*_j. From this last result, **a simple principle emerges to visually deduce instance-wise selection for centroids**: we only have to find the set of their hypercube neighbors to deduce the set dimensions indexes containing contradicting labels, then $\mathcal{J}_j = s^*_j$. We have conveniently drawn all neighbor relations in our figures with dotted lines. This last principle is only valid while working with hypercubes.

We found this "erosion" procedure that deletes random points from an hypercube to be quite interesting as from an initial global selection solution J^k we create many diverse solutions $s^*_j \subset J^k$. An example is given in figure 3 where one point was erased from a bidimensional hypercube (*i.e.* a *XOR*). Instead of having a global selection $S^* = \{1, 2\}$, we end up with $s^*_{-0.125,0.125} = \{1\}$, $s^*_{0.125,0.125} = \{1, 2\}$ and $s^*_{0.125,-0.125} = \{2\}$.

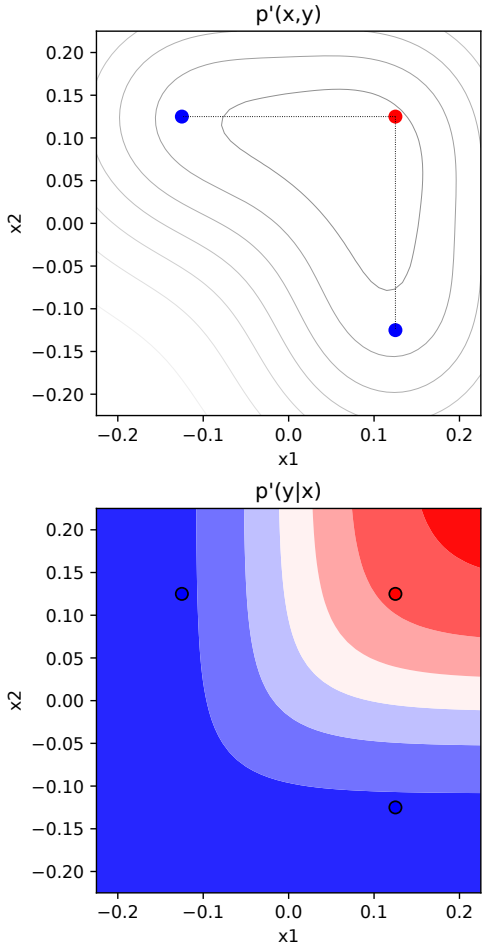

Figure 3. **Task 2_6** The legend is similar to figure 1 and 2.

C.5. Full generative process

To create a collection of tasks, we sample a number of hypercubes to create, generate each one by sampling J^k that gives its orientation along the dimensions, and its occupied coordinates, and finally, we randomly erase some points of the hypercube and update S^* accordingly. The corresponding algorithm is provided in Python in the code repository. More generated examples are given in figures 4 and 5, as well as examples for $\mathcal{X} \subset \mathbb{R}^3$ in figure 6 and 8.

D. Experimental details

We list more implementation details, configurations and tuned parameters for the methods we evaluate.

LIME (Ribeiro et al., 2016)

We set the sampling number to **1000** and the ridge regression parameter to **1**, as in the official implementation. Everything else is similar to the official repository.

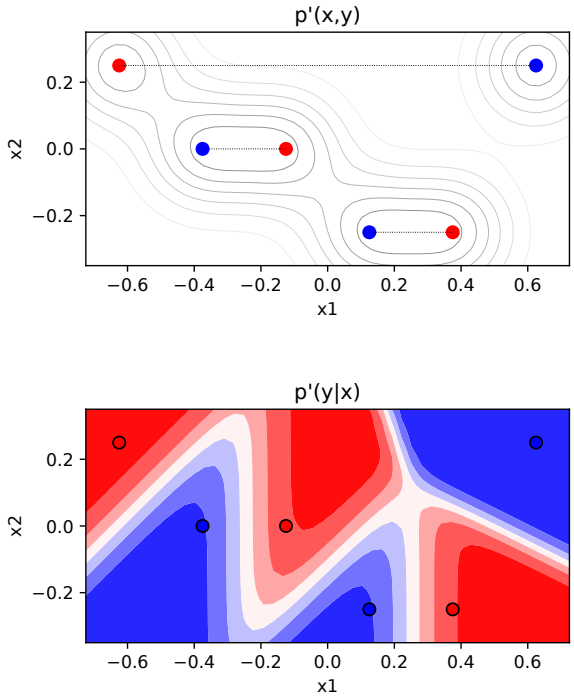

Figure 4. **Task 2_23** The legend is similar to figures 1 and 2.

Shapley Sampling

We set the maximum sampling number to **128**, meaning that up to dimension 7, we compute exact shapley values, and sample permutations otherwise. We tried augmenting this number to **512**, which did yield a non-significative **0.2** accuracy improvement that we chose not to report due to the important trade-off in computation time. We wanted to keep the table clear with a computation budget comparable to $GA^\infty M$ that achieved similar performances.

GAM, ... GA$^\infty$M

No hyper-parameter. These methods directly use the conditional probabilities $p(y = 1 \mid x_I)$, for which we have access to an analytical form, to estimate each restricted expert f_I. Then we use the categorical attribution measure (3), which translates in our case as $\text{attr}_I(x) = \max(f_I(x), 1 - f_I(x))$.

Grad, Grad $\times$ Input (Simonyan et al., 2014)

No hyper-parameter. We use noise-free analytical expressions for f and ∇f.

Integrated Gradient (Sundararajan et al., 2017)

We set the sampling number to **50** for the integral estimation. The baseline point is chosen as the mean of the task centroids.

Expected Gradient (Erion et al., 2019)

We set the sampling number to **500** for tuples of α interpolation coefficients and background points taken among the task centroids.

SHAP (Lundberg and Lee, 2017)

We implement SHAP similarly to Shapley Sampling, the only difference is in the choice of the baseline value. With the original paper notation, $f_S(x) = f(x_S, \mathbb{E}_{\bar{S}}[x_{\bar{S}}])$. Then we directly use $\text{attr}_I(x) = \phi_I(x)$.

Archipelago (Tsang et al., 2020)

No hyper-parameter.

InterpretableNN (Afchar and Hennequin, 2020)

With the original paper notation, we choose $g^i_\theta(x) = 4(F^i_\theta(x) - 0.5)^2$, with F our model output in $[0, 1]$.

L2X (Chen et al., 2018)

We instantiate two three-layer neural network identical in architecture with `selu` activations, and **100** neurons in their hidden layers. For the concrete sampler, we chose $\tau = 0.1$, as in the official implementation. We train the predictor with a cross-entropy loss. The whole model is trained for a maximum of **500** epochs of **100** steps with a batch-size **512**. We add an early stopping after **200** epochs with patience **10** on the selection solution for the task centroids.

INVASE (Yoon et al., 2019)

We instantiate two three-layer neural network with `selu` activations, and **100** neurons in their hidden layers. Following the official implementation, we add batch-normalisation layers in the predictor model. For the selector – referred to as "actor" in the original paper, we grid-searched the regularisation parameter in $[0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1]$ and found the value $\lambda^* = 0.01$. The whole model is trained with the same epoch configuration as *L2X*.

Tuning λ is quite difficult as performances between tasks tend to vary widely with *INVASE*. As illustrated in figure 9, there is no clear significantly better parameter choice. With small λ values, the predictor performances increase as the selector tend to select almost all input features; with higher values the selector sparsity constraint dominates and the predictor quickly collapses to a random 50% accuracy. Most of the time though, *INVASE* does not return the correct selection solution, no matter the value of λ, we suspect that the method falls into the label-passing trap we covered in section B. As we had decided to only grid-search one λ value for all tasks – which worked reasonably well with other methods – we did not try to further tune this method.

Attribution threshold

For all methods returning **feature** attributions *i.e.* n values for the n input features, given values for each point, we select all features with an attribution value higher than μ times the maximum attribution on this point. We tune this multiplicative coefficient by evaluating the methods on 100 tasks, generated and used only for tuning, on a range $[0.1, 0.95]$ with step 0.01. Except for selector-predictors, we have observed rather convex curves of performances and clear maximums for each method, as displayed in figure 10. The obtained parameters are given in table 1. We must underline that the variety in the found coefficients support the specificity and task-dependence we mentioned for each method in the definition of their attribution relative values: some yield sparse attributions values, others do not.

Method	μ^*
LIME (Cat.)	0.23
LIME (Cont.)	0.61
GAM	0.18
Shapley ($\mathbb{E}(f)$)	0.73
SHAP ($f(\mathbb{E})$)	0.28
Gradient	0.67
Gradient x Input	0.73
Integrated Gradient	0.52
Expected Gradient	0.56
L2X	0.82
INVASE	0.55

Table 1. Tuned multiplicative coefficient μ to estimate subset selection from feature attribution.

The remaining methods return **subset** attribution values – *i.e.* 2^n values. They provide estimations of many conditional means $\mathbb{E}(Y \mid X_I = x_I)$, up to a fixed cardinality for I – two for GA^{2}M, three for GA^{3}M, etc. In our case, this last quantity is directly equal to $\mathbb{P}(Y = 1 \mid X_I = x_I)$, and we thus obtain the simple attribution measure (3) we derived by applying the function $g(p) = \max(p, 1 - p)$ on each subfunction output. Then, we use a threshold parameter η and find the subset I with lowest cardinality such that $\text{attr}_I(x) > \eta$. As we have a binary problem, η is bounded in $[1/2, 1]$, we tune η in this range with 0.01 steps. For *InterpretableNN*, a custom function is applied over the probability and yield a method-specific attribution value in $[0, 1]$, we tune η in this range with steps 0.01. The results are given in table 2.

Training

Most models use analytical expressions of $p'_{X,Y}$ and can be evaluated on a consumer grade computer on CPU in less than an hour for each task set. Selector-predictors require a full training procedure and were trained in parallel on four *GeForce GTX 1080* GPUs. Reported running times are aggregated.

Method	η^*
fANOVA	0.76
GA^2M	0.75
GA^3M	0.75
GA^4M	0.76
Archipelago	0.66
InterpretableNN	0.26

Table 2. Tuned multiplicative coefficient μ to estimate subset selection from feature attribution.

E. Issues with model-based interpretations

Following the reviewing process of our paper, we have decided to add a discussion on the comparison/applicability of our formalisation to feature-based interpretations methods that rely on the inspection of the internal of trained models.

As first remark, though we only work on synthetic data distribution $p'(y|x)$ in our experiment section, our framework is perfectly applicable to a model induced distribution $p_\theta(y|x)$. As mentioned in introduction, this can make evaluation trickier with the added difficulty of disentangling model prediction errors from interpretation errors on unlabeled data, and requires to manage out-of-distribution artifacts impacting the produced interpretations, which is also the case of models architecture and hyperparameters choice (Dombrowski et al., 2019; Kumar et al., 2020; Slack et al., 2020). The approach in itself would however remain *model-agnostic* and solely inspect the learnt input-output association. This is arguably a common principle in the interpretation field (*e.g.* (Ribeiro et al., 2016; Lundberg and Lee, 2017)), but a concern was raised of whether inspecting trained weights would not simplify the attribution problem.

For instance, in the case of a decision tree – that are often considered one of the models with the highest transparency level (Arrieta et al., 2020) – a commonly used principle to find responsible features is to aggregate encountered features on which the branching are done from root to leaf to form a prediction rationale. The interpretability of trees and other simple models has already been disputed before (Lipton, 2018; Dinu et al., 2020); here, to fix ideas, we highlight an example where the two explanation principles would differ. Consider figure 3: on one side, we have seen that our method allows to derive a *unique* minimal instance-wise selection solution. Conversely, though the tree-based suggestion seems reasonable at first glance, there exists two optimal trees classifying all three clusters perfectly,

$$
\begin{aligned}
T_1(x) := \ & \texttt{if } x_1 < 0 \texttt{ then } 0 \\
& \texttt{else (if } x_2 > 0 \texttt{ then } 1 \texttt{ else } 0) \\
T_2(x) := \ & \texttt{if } x_2 < 0 \texttt{ then } 0 \\
& \texttt{else (if } x_1 > 0 \texttt{ then } 1 \texttt{ else } 0)
\end{aligned}
$$

leading to *two contradicted selections that may be equiprobably returned* on several runs:

(x_1, x_2)	T_1	T_2	**Our**
-0.125, 0.125	$\{1\}$	$\{1,2\}$	$\{1\}$
0.125, 0.125	$\{1,2\}$	$\{1,2\}$	$\{1,2\}$
0.125, -0.125	$\{1,2\}$	$\{2\}$	$\{2\}$

Trees suffer from *identifiability* issues leading to unstable explanations, which seems unsuitable to gain general knowledge. The clusters symmetry between X_1 and X_2 also seems a good argument in favor of our solution.

Leveraging model inner mechanisms for interpretation has lead to many well performing algorithms, but this also paves the ways to many undesired side-effects that are not immediately visible when working with a high-performing trained model. Our message is that prediction performance and computation transparency does not necessarily translate into interpretation performance. Optimistically, we believe that the study we have conducted on synthetic data helps find those inconsistencies and failure points and may lead to better behaved interpretable models.

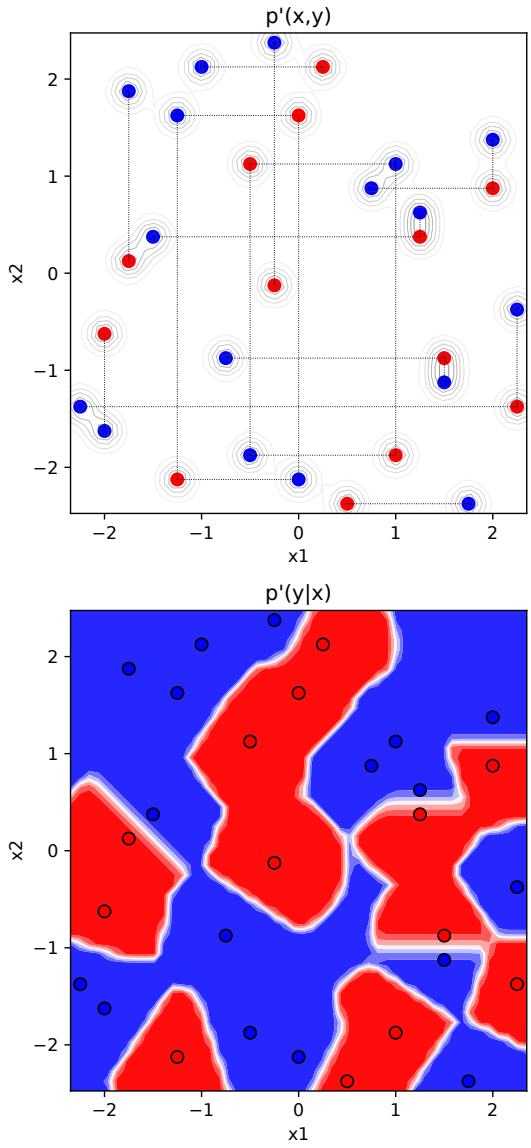

Figure 5. **Task 2_100** The legend is similar to figures 1 and 2.

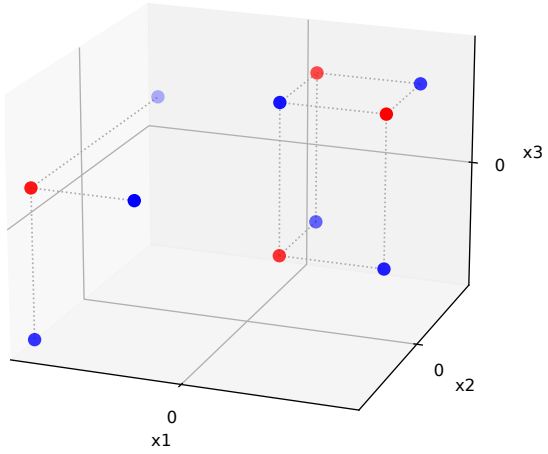

Figure 6. **Task 3_12** We only display centroids and neighbors

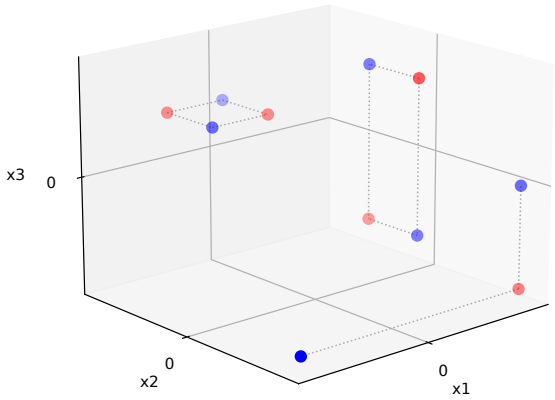

Figure 7. **Task 3_27**

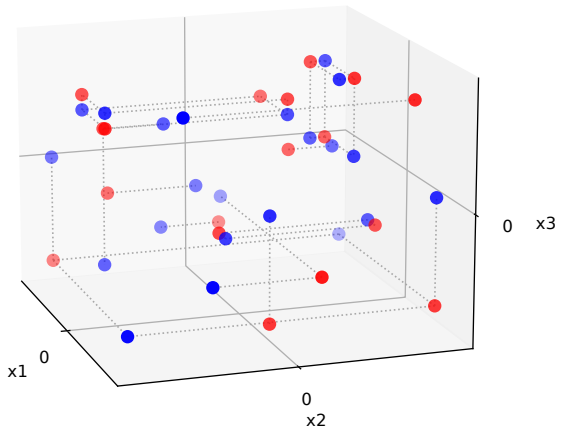

Figure 8. **Task 3_100**

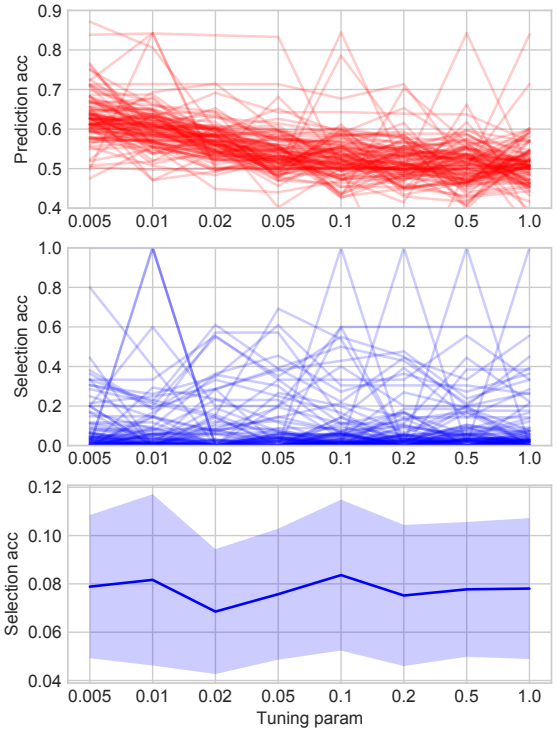

Figure 9. **Tuning curves for *INVASE*** We plot the *predictor* performances at predicting centroid labels in red, and the *selector* estimated selection mask accuracies in blue. We superpose the tuning curves from each task in the tuning task set and display a clearer aggregated selection accuracy curve with 95% confidence intervals. Our chosen λ^* maximises upper confidence bound.

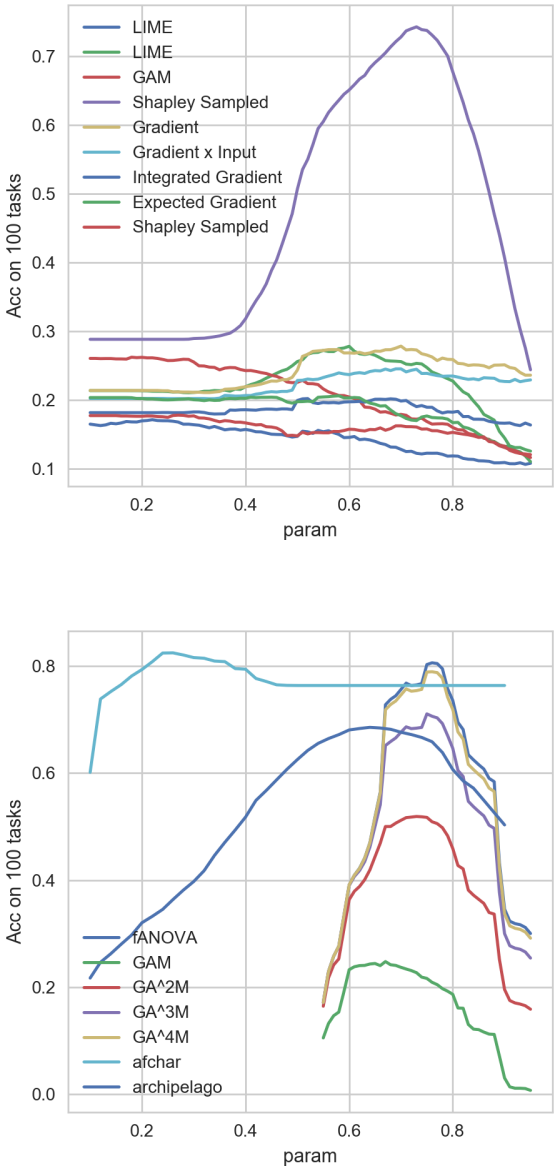

Figure 10. Tuning curves of subset attribution methods

References

Afchar, D. and Hennequin, R. (2020). Making neural networks interpretable with attribution: Application to implicit signals prediction. In *RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020*, pages 220–229. ACM.

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al. (2020). Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115.

Chen, J., Song, L., Wainwright, M. J., and Jordan, M. I. (2018). Learning to explain: An information-theoretic perspective on model interpretation. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80, pages 882–891. PMLR.

Dinu, J., Bigham, J., and Kolter, J. Z. (2020). Challenging common interpretability assumptions in feature attribution explanations. *NeurIPS 2020 ML-Retrospectives, Surveys & Meta-Analyses Workshop*.

Dombrowski, A., Alber, M., Anders, C. J., Ackermann, M., Müller, K., and Kessel, P. (2019). Explanations can be manipulated and geometry is to blame. In *Advances in Neural Information Processing Systems 32, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13567–13578.

Erion, G., Janizek, J. D., Sturmfels, P., Lundberg, S., and Lee, S.-I. (2019). Learning explainable models using attribution priors. *arXiv preprint arXiv:1906.10670*.

Foldes, S. (1977). A characterization of hypercubes. *Discrete Mathematics*, 17(2):155–159.

Kumar, I. E., Venkatasubramanian, S., Scheidegger, C., and Friedler, S. A. (2020). Problems with shapley-value-based explanations as feature importance measures. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119, pages 5491–5500. PMLR.

Lipton, Z. C. (2018). The mythos of model interpretability. *Queue*, 16(3):31–57.

Lundberg, S. M. and Lee, S. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30, December 4-9, 2017, Long Beach, CA, USA*, pages 4765–4774.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "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, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144. ACM.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. *Workshop, ICLR*.

Slack, D., Hilgard, S., Jia, E., Singh, S., and Lakkaraju, H. (2020). Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186.

Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70, pages 3319–3328. PMLR.

Tsang, M., Rambhatla, S., and Liu, Y. (2020). How does this interaction affect me? interpretable attribution for feature interactions. *Advances in Neural Information Processing Systems*.

Yoon, J., Jordon, J., and van der Schaar, M. (2019). INVASE: instance-wise variable selection using neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.