Générer des explications contrefactuelles à l'aide d'un autoencodeur supervisé

Victor Guyomard, Françoise Fessant, Tassadit Bouadi, Thomas Guyet

Générer des explications contrefactuelles à l'aide d'un autoencodeur supervisé

Victor Guyomard[*,**], Françoise Fessant[*]
Tassadit Bouadi[**] Thomas Guyet[***]

[*]Orange, Lannion, France
[**]Univ Rennes, Inria, CNRS, IRISA, Rennes, France
[***]Inria – Grenoble Rhône-Alpes, Villeurbanne, France

Résumé. Dans cet article nous proposons une manière d'améliorer l'interprétabilité des explications contrefactuelles. Une explication contrefactuelle se présente sous la forme d'une version modifiée de la donnée à expliquer qui répond à la question : que faudrait-il changer pour obtenir une prédiction différente ? La solution proposée consiste à introduire dans le processus de génération du contrefactuel un terme basé sur un auto-encodeur supervisé. Ce terme contraint les explications générées à être proches de la distribution des données et de leur classe cible. La qualité des contrefactuels produits est évaluée sur un jeu de données d'images par le biais de différentes métriques. Nous montrons que notre solution s'avère compétitive par rapport à une méthode de référence de l'état de l'art.

1 Introduction

L'apprentissage automatique est désormais massivement utilisé pour automatiser la prise de décision dans de nombreux domaines, et en particulier dans des domaines qui impactent notre vie quotidienne. Les modèles utilisés sont généralement complexes et opaques. C'est le phénomène de la « boite noire ». L'IA explicable (ou XAI) vise à limiter ce problème en fournissant un ensemble de méthodes pour qu'un utilisateur humain comprenne les facteurs qui ont motivé la décision d'un modèle. L'enjeu de l'explicabilité devient crucial que ce soit pour l'acceptation de l'IA ou le respect des réglementations. Par exemple, le règlement général sur la protection des données de l'union européenne, entré en application en 2018, introduit un droit à l'explication pour les individus lorsque la prise de décision automatisée les affecte significativement. De nombreuses approches d'explicabilité ont été développées récemment et plusieurs typologies existent pour les classifier (Molnar, 2019). On s'intéresse ici aux méthodes d'interprétabilité *post-hoc* locales, c'est à dire qui s'appliquent après l'apprentissage du modèle de classification et pour une prédiction donnée. Les explications contrefactuelles appartiennent à cette catégorie. Le principe est d'expliquer la décision du modèle de classification à l'aide d'un exemple, proche de l'exemple à expliquer, qui montre comment celui-ci devrait changer pour que sa prédiction change.

La plupart des méthodes d'explications contrefactuelles sont basées sur la perturbation de l'instance originale grâce à l'optimisation d'une fonction de coût (Wachter et al., 2018). Se-

lon les propriétés souhaitées pour l'explication on rajoute des contraintes dans le processus d'optimisation sous la forme de termes supplémentaires dans la fonction de coût. Par exemple, on peut souhaiter un contrefactuel le plus proche possible de l'exemple à expliquer, avec le moins de variables perturbées possible, actionnable (où seules certaines variables peuvent être perturbées) ou encore réaliste. Pour Van Looveren et Klaise (2021), Mahajan et al. (2019) ou Dhurandhar et al. (2018) le réalisme consiste à faire en sorte que le contrefactuel respecte la distribution des données (tout du moins qu'il ne s'en écarte pas trop). Cette propriété est atteinte par l'ajout dans la fonction de coût d'un terme basé sur un autoencodeur. L'autoencodeur est un réseau de neurones artificiels qui cherche à reconstruire le plus fidèlement possible en sortie les exemples qui lui ont été présentés en entrée (Kramer, 1991). On cherche donc à générer des contrefactuels qui minimisent cette erreur de reconstruction ; une erreur importante indiquant un contrefactuel loin de la distribution des données et ainsi moins réaliste.

Dans cet article nous proposons une extension du travail de Van Looveren et Klaise (2021) avec le remplacement de l'autoencodeur par un autoencodeur supervisé de manière à construire un contrefactuel qui respecte la distribution des données mais également la distribution des caractéristiques des classes cibles. Après avoir rappelé l'algorithme de Van Looveren et Klaise (2021), nous décrivons notre proposition et nous comparons la qualité des contrefactuels produits sur un jeu de données d'images par le biais de différentes métriques. Nous montrons que l'organisation de l'espace latent de l'autoencodeur par rapport aux classes des exemples lors de l'apprentissage, conduit à des explications contrefactuelles fidèles à l'ensemble de la distribution des données mais aussi à la distribution des données de la classe cible.

2 Génération d'explications contrefactuelles *post-hoc*

Cette section formalise le processus de génération d'explications contrefactuelles *post-hoc*, ainsi que la méthode de l'état de l'art dont nous proposons l'extension.

On notera $\mathcal{X} \subseteq \mathbb{R}^p$ l'espace des variables de dimension p, $D = \{(x_i, y_i)\}_{i=1}^n$ un ensemble d'apprentissage tel que $x_i \in \mathcal{X}$ et $y_i \in \{1, \ldots, l\}$ pour tout $i \in \{1, \ldots, n\}$. L'explication *post-hoc* consiste à appliquer une méthode d'explicabilité une fois le modèle de classification $f_{\text{pred}} : \mathcal{X} \mapsto \{1, \ldots, l\}$ entraîné. Ce modèle prédit, pour un exemple donné, l'appartenance à une classe parmi les l classes possibles et $[f_{\text{pred}}(x)]_y$ représente la probabilité d'appartenance à la classe y pour un exemple x. Dans ce contexte, la génération d'exemples contrefactuels peut être vue comme la solution d'un problème d'optimisation (Wachter et al., 2018). Soit x_0 l'exemple à expliquer, un contrefactuel sera alors un autre exemple $x_{\text{cf}} = x_0 + \delta$ proche de x_0 mais classifié par le modèle de classification f_{pred} avec une classe différente (δ est la perturbation à appliquer à l'exemple pour qu'il change de classe). Il est ainsi possible de définir un contrefactuel comme l'*exemple le plus proche* de l'exemple à expliquer mais dont la classe prédite est différente.

Cependant, cette approche peut conduire à des contrefactuels peu réalistes ou peu actionnables (De Oliveira et Martens, 2021). Plusieurs stratégies ont été proposées pour agir sur le réalisme des contrefactuels comme par exemple l'ajout de contraintes supplémentaires dans l'algorithme d'optimisation (Mothilal et al., 2020), ou dans le processus de génération de contrefactuels afin que ces derniers soient proches de la distribution des données. Li et al. (2018) ou Pawelczyk et al. (2020) proposent d'utiliser pour cela un autoencodeur entraîné sur l'ensemble d'apprentissage.

Van Looveren et Klaise (2021) proposent de combiner les deux types d'approches présentés ci-dessus, pour construire de manière *post-hoc* des contrefactuels interprétables, par le biais d'une fonction de coût comprenant différents termes de pénalisation.

Plus précisément, un contrefactuel $x_{\text{cf}} = x_0 + \delta \in \mathcal{X}$ est obtenu en optimisant la fonction de coût suivante :

$$\min_{\delta} \left(c \cdot f_\tau(x_0, \delta) + ||\delta|| + \gamma \cdot L_{AE} + \theta \cdot L_{\text{proto}} \right) \tag{1}$$

$f_\tau(x_0, \delta)$ encourage la classe prédite y_i du contrefactuel x_{cf} à être différente de la classe y_0 de l'exemple à expliquer x_0 :

$$f_\tau(x_0, \delta) = \max \left([f_{\text{pred}}(x_0 + \delta)]_{y_0} - \max_{y_i \in \{1,\dots,l\}, y_i \neq y_0} [f_{\text{pred}}(x_0 + \delta)]_{y_i},\ -\tau \right)$$

où $\tau \geq 0$ cape la différence entre $[f_{\text{pred}}(x_0 + \delta)]_{y_0}$ et $[f_{\text{pred}}(x_0 + \delta)]_{y_i}$.

$||\delta|| = \beta \cdot \|\delta\|_1 + \|\delta\|_2^2$ cherche à minimiser la distance entre x_0 et x_{cf} de manière à générer une perturbation "creuse", c'est-à-dire pour laquelle peu de valeurs d'attributs sont modifiées.

$L_{AE} = \|x_0 + \delta - \text{AE}_D(x_0 + \delta)\|_2^2$ représente l'erreur de reconstruction de x_{cf} évaluée par un autoencoder AE entraîné sur D. Cet autoencodeur est composé d'un encodeur (ENC_D) et d'un décodeur (DEC_D) tels que $\text{AE}_D(x) = \text{DEC}_D(\text{ENC}_D(x))$ pour tout $x \in \mathcal{X}$.

Intuitivement, cela pénalise les contrefactuels qui seraient loin des exemples dans $\mathcal{X}$.

Finalement, le dernier terme L_{proto} est défini de la façon suivante :

$$L_{\text{proto}} = \|\text{ENC}_D(x_0 + \delta) - \text{proto}_{y_j}\|_2^2,$$

où proto_{y_j} est un prototype (un exemple représentatif) calculé comme la moyenne dans l'espace latent, des K plus proches exemples de l'exemple à expliquer x_0 parmi ceux dont la classe prédite par f_{pred} est y_j :

$$\text{proto}_{y_j} = \frac{1}{K} \sum_{k=1}^{K} \text{ENC}_D \left(x_k^j \right) \tag{2}$$

où x_k^j est le k-ième exemple le plus proche dans l'espace latent ayant pour classe prédite y_j, avec $y_j = \text{argmin}_{y_i \in \{1,\dots,l\}, y_i \neq y_0} \|\text{ENC}(x_0) - \text{proto}_{y_i}\|_2$. Ce qui signifie que le prototype choisi est le plus proche de x_0 pour une classe $y_j \neq y_0$ dans l'espace latent. L'idée est de guider la recherche de contrefactuels autour de proto_{y_j}. Notons que proto_{y_j} est un prototype *abstrait* car il est défini dans l'espace latent et non dans l'espace initial des exemples. Cependant, il est toujours possible d'obtenir un prototype dans l'espace des exemples en utilisant $DEC_D(\text{proto}_{y_j})$. Définir les prototypes dans l'espace latent est intéressant dans la mesure où les distances dans cet espace capturent mieux la sémantique des exemples (Li et al., 2018; Pawelczyk et al., 2020).

K est un hyperparamètre qui représente le nombre d'exemples utilisés pour calculer les prototypes. Les autres hyperparamètres (c, β, γ, θ), et τ sont les pondérations de la fonction objectif. Pour une description complète des paramètres et de leur impact sur la recherche de contrefactuels, se reporter à Van Looveren et Klaise (2021).

3 Autoencodeur supervisé pour la génération des contrefactuels

Dans cette section nous allons présenter notre contribution.

La méthode proposée par Van Looveren et Klaise (2021) qui guide la recherche des contrefactuels avec des prototypes dans l'espace latent permet d'obtenir, en théorie, des contrefactuels qui sont plus proches de la distribution des données d'une classe. Cependant, en pratique, on a pu observer que ce n'est pas toujours le cas, avec des contrefactuels parfois loin de la distribution de leur classe de prédiction. La Figure 1 donne une intuition de ce phénomène, avec un exemple issu de la base MNIST [1]. L'exemple à expliquer ici est prédit comme un `5` par le modèle de classification et le contrefactuel correspondant est prédit comme un `3`. On peut observer que le contrefactuel est plus proche de la distribution des données de la classe `5` que de celle des données de sa classe prédite (classe `3`). Intuitivement, le contrefactuel « ressemble plus » à un `5` qu'à un `3`.

Les prototypes dans l'espace latent étant des exemples représentatifs de chaque classe, il semble ainsi pertinent de prendre en compte la notion de classe dans la construction de l'espace latent. Nous proposons donc d'apprendre l'espace latent avec un autoencodeur supervisé. Un autoencodeur supervisé est un type d'autoencodeur qui apprend conjointement une tâche d'apprentissage supervisé et une tâche de reconstruction (Le et al., 2018). Ce type de réseau possède deux sorties : l'une qui retourne l'exemple reconstruit et l'autre qui retourne la décision associée à l'exemple. La fonction d'erreur globale utilisée pour l'apprentissage est définie comme la somme pondérée des erreurs de classification et de reconstruction :

$$L((f_{\text{pred}}, \text{AE}), D) = \underbrace{E(f_{\text{pred}}, D)}_{\text{Erreur de classification}} + \lambda \underbrace{R(\text{AE}, D)}_{\text{Erreur de reconstruction}} \tag{3}$$

où

— L'erreur de reconstruction est donnée par la moyenne des distances L_2 sur un ensemble d'apprentissage de *n* exemples :

$$R(\text{AE}, D) = \frac{1}{n} \sum_{i=1}^{n} \|\text{AE}(x_i) - x_i\|_2^2$$

— L'erreur de classification est définie comme une entropie croisée sur l'ensemble d'apprentissage (D) :

$$E(f_{\text{pred}}, D) = \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{l} -\mathbb{1}_{[y_i=k]} \log\left([f_{\text{pred}}(x_i)]_k\right)$$

λ est l'hyperparamètre qui permet d'ajuster le compromis entre les deux termes.

Notre proposition est maintenant d'utiliser cet autoencodeur supervisé pour générer des contrefactuels selon le principe défini dans la Section 2. L'intuition étant que la supervision de l'autoencodeur va produire un espace latent organisé en fonction des classes. Les prototypes correspondent à des exemples moyens dans l'espace latent. L'organisation de ce dernier devrait

1. `http://yann.lecun.com/exdb/mnist/`

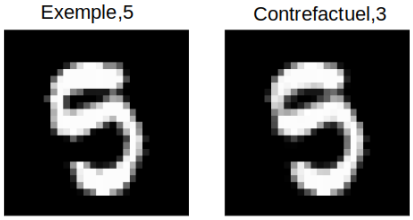

FIG. 1 – *Exemple original issu de MNIST(à gauche) et contrefactuel correspondant généré par la méthode de Van Looveren et Klaise (2021) (à droite). Le contrefactuel proposé n'est pas un bon représentant de la classe à laquelle il appartient.*

alors conduire à des prototypes plus représentatifs d'une classe donnée, et donc à la génération de contrefactuels également plus représentatifs et interprétables. Le modèle de prédiction utilisé pour la génération des contrefactuels sera celui obtenu par supervision de l'autoencodeur (f_{pred}, obtenu en minimisant l'équation 3).

4 Expérimentations

L'objectif des expérimentations présentées dans cette section est d'évaluer l'effet de la supervision de l'autoencodeur sur la qualité des contrefactuels générés.

La première question que l'on se pose est : *les contrefactuels générés sont-ils plus interprétables quand ils sont obtenus avec un autoencoder supervisé que non supervisé ?* Nous allons donc générer deux ensembles de contrefactuels à partir de la fonction d'optimisation décrite dans la Section 2 : l'un obtenu avec l'utilisation d'un autoencodeur supervisé (Section 3), et l'autre sans supervision de l'autoencodeur que l'on nommera *baseline* dans la suite de l'article. Les ensembles de contrefactuels seront comparés à l'aide de différentes métriques.

La deuxième question que l'on se pose est : *La supervision d'un autoencodeur permet-elle l'obtention d'un espace latent où les exemples sont mieux organisés suivant les classes ?* Nous allons pour cela projeter des exemples dans un espace latent de dimension 2, avec et sans supervision.

4.1 Métriques

Nous proposons d'évaluer la qualité des contrefactuels produits à l'aide de trois métriques différentes : le gain de prédiction, le réalisme ainsi que l'actionnabilité (Nemirovsky et al., 2020)

— **Gain de prédiction :** Le gain de prédiction est donné par la différence entre la probabilité prédite du contrefactuel et la probabilité prédite de l'exemple à expliquer, pour la classe du contrefactuel.

$$\text{Gain} = [f_{\text{pred}}(x_{cf})]_{y_i} - [f_{\text{pred}}(x_0)]_{y_i} \tag{4}$$

où y_i est la classe prédite pour le contrefactuel.
Le gain de prédiction est borné dans $[0, 1]$, et une valeur plus élevée implique plus de confiance dans le changement de classe du contrefactuel.

— **Réalisme :** Le réalisme correspond à l'erreur de reconstruction du contrefactuel évaluée par un autoencodeur ($\mathrm{AE}_{\mathrm{evaluate}}$).

$$\text{Réalisme} = \left\| \mathrm{AE}_{\mathrm{evaluate}}\left(x_{cf}\right) - x_{cf} \right\|_2^2 \tag{5}$$

Une valeur faible indique que le contrefactuel est plus proche de la distribution des données.

— **Actionnabilité :** L'actionnabilité est la distance L_1 entre l'exemple et le contrefactuel.

$$\text{Actionnabilité} = \left\| x_{cf} - x_0 \right\|_1 = \left\| \delta \right\|_1 \tag{6}$$

Une faible valeur indique une perturbation plus creuse, ce qui correspond au changement de moins de caractéristiques.

On notera que d'autres métriques d'évaluation ont été proposées, en particulier IM1 et IM2 (Van Looveren et Klaise, 2021) qui sont basées sur des erreurs de reconstruction d'autoencodeurs. Plusieurs auteurs ayant souligné le manque de robustesse de ces métriques (Labaien et al., 2021; Schut et al., 2021), nous ne les avons pas considérées dans le cadre de cet article.

4.2 Comparaison de la qualité des contrefactuels

Le protocole expérimental peut être résumé à l'aide des étapes listées ci-dessous :

1. **Contrefactuels basés sur un autoencodeur supervisé**
   (a) Entraîner un autoencodeur supervisé, sur un ensemble de données d'apprentissage fixé, en minimisant la fonction d'erreur pondérée qui pénalise la reconstruction et la classification (Section 3).
   (b) Extraire l'autoencodeur et le classifieur correspondant.
   (c) Selectionner un ensemble d'exemples de test.
   (d) Construire les contrefactuels pour les exemples de l'étape 1c, avec la méthode décrite dans la Section 2.
2. **Contrefactuels basés sur un autoencodeur non supervisé**
   (a) Entraîner un nouvel autoencodeur, avec la même architecture que l'autoencodeur supervisé, mais sans la supervision, (*i.e.* avec une fonction d'erreur de reconstruction uniquement) sur le même ensemble d'apprentissage que précédemment.
   (b) Construire les contrefactuels (pour les même exemples de test que dans 1) avec ce nouvel autoencodeur et le classifieur de l'étape 1a.
3. Calculer les métriques sur les deux ensembles de contrefactuels obtenus.

4.2.1 Comparaison des métriques

Ce protocole expérimental a été évalué sur le jeu de données MNIST pour les métriques présentées dans la Section 4.1. Un échantillon aléatoire de $5,000$ exemples de test a été sélectionné pour la génération des contrefactuels. Les architectures précises des modèles, ainsi que

les hyperparamètres sont détaillés en annexe. Les performances relatives à l'apprentissage sont également fournies en annexe. Le Tableau 1 donne les résultats numériques (moyenne et écart type pour chacune des métriques sur les exemples de l'ensemble de test) pour notre approche (autoencodeur supervisé) et la *baseline* (autoencodeur sans supervision).

Métriques	*Baseline*	Autoencodeur supervisé
↑ Gain de prédiction	0.552±0.106	**0.839±0.160**
↓ Réalisme	0.253±0.010	**0.249±0.012**
↓ Actionnabilité	**26.174±13.762**	38.360 ±18.465

TAB. 1 – *Comparaison des résultats pour les différentes métriques. Les flèches indiquent si une valeur faible ↓ ou élevée ↑ implique un meilleur résultat. Les meilleures performances sont indiquées en gras.*

Les résultats numériques montrent que notre méthode permet d'obtenir un meilleur gain de prédiction (la moyenne est supérieure de 0.287 points), avec un réalisme équivalent à la méthode *baseline*. Un meilleur gain de prédiction signifie plus de confiance dans la classe prédite du contrefactuel. Cependant nous perdons en actionnabilité ce qui signifie que les contrefactuels impliquent en moyenne la modification de plus nombreux pixels. Afin de mieux comprendre ces différences de comportement, il peut être intéressant de regarder ce qui se passe au niveau local, c'est-à-dire pour certains exemples.

4.2.2 Comparaison des contrefactuels générés

La Figure 2 présente les contrefactuels produits par les deux méthodes pour trois exemples différents. La colonne de gauche montre les exemples à expliquer, les deuxième et troisième colonnes représentent respectivement les contrefactuels obtenus avec un autoencodeur supervisé et la méthode *baseline*.

Nous observons globalement plus de changements de pixels dans le cas de l'autoencodeur supervisé. Par exemple, sur la première ligne d'images, un exemple prédit comme un `3` aura un contrefactuel prédit comme un `6` dans les deux cas. Cependant, on observe que plus de pixels ont été modifiés avec la méthode basée sur l'autoencodeur supervisé (avec la création de la boucle qui permet d'obtenir un `6`). De plus, le contrefactuel obtient une plus grande probabilité de prédiction pour la classe `6` (0.99 vs 0.41), ce qui conduit à un gain de prédiction plus important. Sur la deuxième ligne, l'exemple à expliquer est prédit comme un `9` et les contrefactuels comme un `5`. Nous observons le même comportement que précédemment sur les contrefactuels. Dans le cas de l'autoencodeur supervisé, des pixels sont modifiés dans la boucle du `9` afin d'obtenir un `5` alors que moins de pixels sont modifiés dans le cas de la *baseline*, ce qui induit moins de changements visuels (le contrefactuel est prédit comme un `5` mais ressemble visuellement à un `9`). Dans la dernière ligne, l'exemple est prédit comme un `2` et le contrefactuel produit est prédit comme un `1` dans le cas de l'autoencodeur supervisé, et comme un `8` dans le cas de la *baseline*. Pour la *baseline*, la perturbation produit un contrefactuel qui apparaît comme étant visuellement similaire à l'exemple alors que la classe prédite est `8` (avec une probabilité de prédiction de 0.43). Avec l'autoencodeur supervisé, le contrefactuel est

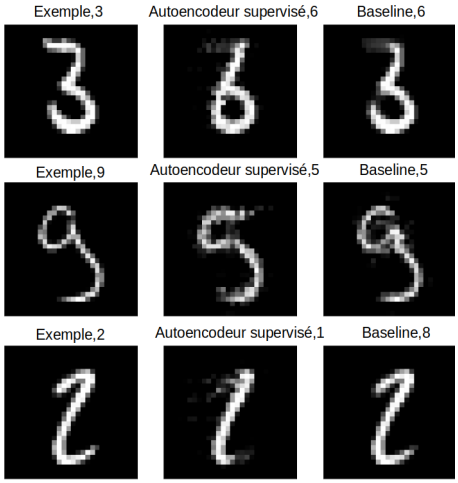

FIG. 2 – *Exemples à expliquer de MNIST (colonne de gauche) et les contrefactuels correspondants pour la méthode basée sur l'autoencodeur supervisé (au centre) et la méthode baseline (à droite).*

converti en un 1 avec une probabilité de 0.99. Il a été obtenu par la modification des pixels aux extrémités.

4.3 Impact de la supervision sur l'espace latent

Le protocole de cette expérimentation est le suivant :

1. **Espace latent d'un autoencodeur supervisé**
   (a) Entraîner un autoencodeur supervisé (même étape que celle du protocole précédent mais avec un espace latent de dimension 2)
   (b) Projeter un ensemble d'exemples de test dans cet espace latent .
   (c) Calculer des prototypes à partir de l'étape 1b comme étant des exemples moyens dans l'espace latent pour chacune des classes, puis les décoder.
2. **Espace latent d'un autoencodeur non supervisé**
   (a) Entraîner un nouvel autoencodeur, avec la même architecture que l'autoencodeur supervisé, mais sans la supervision (même étape que celle du protocole précédent mais également avec un espace latent de dimension 2)
   (b) Projeter le même ensemble de test utilisé à l'étape 1b dans ce nouvel espace latent.
   (c) Calculer les prototypes (étape 1c mais avec le nouvel espace latent)
3. Comparer visuellement les différentes projections ainsi que l'impact sur les prototypes décodés.

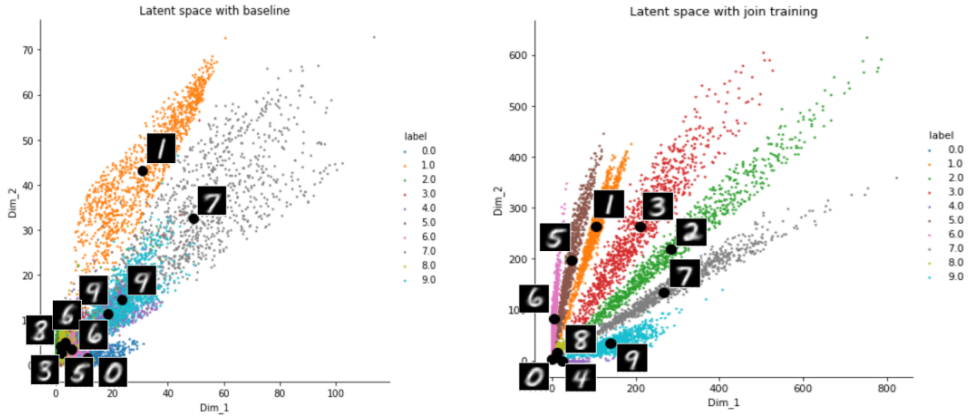

FIG. 3 – *Visualisation d'un espace latent de dimension 2 avec le jeu de donnée MNIST*

4.3.1 Résultats

La Figure 3 montre la projection des exemples de MNIST dans deux espaces latents d'autoencodeurs. La figure de gauche correspond à la *baseline*, celle de droite à l'autoencodeur supervisé. Un code couleur indique dans quelle classe l'exemple a été prédit. Chaque imagette noire correspond à un prototype dans l'espace de départ (obtenu à l'aide du décodeur).

Cette expérience confirme notre intuition de départ (*cf.* Section 3). Dans le cas de l'autoencodeur supervisé, l'espace latent est organisé suivant les classes, avec des clusters de points séparés par classe. On n'observe pas la même configuration avec l'autoencodeur *baseline* où la représentation ne prend pas en compte les classes : des exemples de classes différentes sont mélangés dans l'espace latent. Nous observons également que, dans le cas de l'autoencodeur supervisé, cela a pour effet de conduire à des prototypes plus représentatifs de chaque classe. En effet, les exemples moyens décodés ont des représentations disjointes contrairement au cas de l'autoencodeur *baseline* pour lequel la superposition d'exemples de classes différentes dans l'espace latent conduit à des prototypes ayant des représentations similaires.

4.4 Bilan global

L'évaluation du premier protocole montre qu'un meilleur gain de prédiction est associé à une plus grande confiance dans la classe prédite du contrefactuel. Un meilleur gain s'obtient au prix d'une modification d'un plus grand nombre de caractéristiques de l'exemple à expliquer. La visualisation de l'espace latent confirme notre intuition qui est que la supervision permet une organisation suivant les classes cibles dans l'espace latent, ce qui conduit à des prototypes plus représentatifs de chacune des classes. Notre protocole a également été évalué sur un ensemble de données tabulaires numériques (*cf.* Annexe). Les résultats de l'évaluation du premier protocole sont fournis en annexe. Les conclusions sont globalement similaires à celles obtenues sur les données d'images.

5 Conclusion

Cet article a proposé une méthode de génération *post-hoc* d'explications contrefactuelles. La proposition consiste à introduire un terme basé sur un autoencodeur supervisé dans la fonction de coût qui permet d'obtenir le contrefactuel à partir de l'exemple à expliquer. Ce travail est une extension de celui de Van Looveren et Klaise (2021) avec l'objectif d'améliorer la fidélité des contrefactuels à la distribution des données de la classe cible. Les prototypes sont calculés dans un espace latent organisé suivant les classes, ce qui permet de guider la recherche de contrefactuels vers des prototypes plus représentatifs. La méthode a été évaluée sur un jeu de données d'images par le biais de différentes métriques. On a obtenu des contrefactuels avec une plus grande confiance dans leur classe de prédiction mais au prix d'une modification d'un plus grand nombre de caractéristiques de l'exemple à expliquer. Ces résultats restent à conforter sur plus de jeux de données de différents types.

Références

De Oliveira, R. M. B. et D. Martens (2021). A framework and benchmarking study for counterfactual generating methods on tabular data. *Applied Sciences 11*(16), 7274.

Dhurandhar, A., P.-Y. Chen, R. Luss, C.-C. Tu, P. Ting, K. Shanmugam, et P. Das (2018). Explanations based on the missing : Towards contrastive explanations with pertinent negatives. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, pp. 590–601.

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal 37*(2), 233–243.

Labaien, J., E. Zugasti, et X. De Carlos (2021). DA-DGCEx : Ensuring validity of deep guided counterfactual explanations with distribution-aware autoencoder loss. arXiv :2104.09062.

Le, L., A. Patterson, et M. White (2018). Supervised autoencoders : Improving generalization performance with unsupervised regularizers. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, pp. 107–117.

Li, O., H. Liu, C. Chen, et C. Rudin (2018). Deep learning for case-based reasoning through prototypes : A neural network that explains its predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 3530–3537.

Mahajan, D., C. Tan, et A. Sharma (2019). Preserving causal constraints in counterfactual explanations for machine learning classifiers. In *Proceedings of the workshop Microsoft at NIPS – "CausalML : Machine Learning and Causal Inference for Improved Decision Making"*.

Molnar, C. (2019). *Interpretable Machine Learning*. `https://christophm.github.io/interpretable-ml-book/`.

Mothilal, R. K., A. Sharma, et C. Tan (2020). Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT)*, pp. 607–617.

Nemirovsky, D., N. Thiebaut, Y. Xu, et A. Gupta (2020). CounteRGAN : Generating realistic counterfactuals with residual generative adversarial nets. arXiv :2009.05199.

Pawelczyk, M., K. Broelemann, et G. Kasneci (2020). Learning model-agnostic counterfactual explanations for tabular data. In *Proceedings of The Web Conference*, pp. 3126–3132.

Schut, L., O. Key, R. McGrath, L. Costabello, B. Sacaleanu, M. Corcoran, et Y. Gal (2021). Generating interpretable counterfactual explanations by implicit minimisation of epistemic and aleatoric uncertainties. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, Volume 130, pp. 1756–1764.

Van Looveren, A. et J. Klaise (2021). Interpretable counterfactual explanations guided by prototypes. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*, pp. 650–665.

Wachter, S., B. Mittelstadt, et C. Russell (2018). Counterfactual explanations without opening the black box : Automated decisions and the GDPR. *Harvard journal of law & technology 31*, 841–887.

Annexe

Cette section décrit les architectures des différents modèles implémentés et donne les valeurs des hyperparamètres utilisés lors des expérimentations.

Autoencodeur supervisé Cette architecture est composée d'une partie autoencodeur ainsi que de 2 couches denses (couches de classification) à la suite de l'encodeur. La sortie finale est la concaténation de la dernière couche de classification et de la sortie du décodeur. La partie autoencodeur est la même que celle utilisée par Van Looveren et Klaise (2021). La partie classification est composée de deux couches denses à la suite de l'encodeur. Cette partie prend en entrée un vecteur d'exemples encodés (par l'encodeur) et est ensuite connectée à une couche dense de taille 128 avec une activation *ReLU* et une régularisation de type L_1. Cette couche dense est suivie d'une autre couche dense de taille 10 avec activation *softmax*. La fonction d'erreur est une somme pondérée de fonctions d'erreur de reconstruction et de classification comme détaillé dans la Section 3. Afin de fixer λ dans l'équation 3, nous avons entraîné notre modèle pour différentes valeurs de λ et choisi le meilleur compromis entre précision et erreur de reconstruction sur le jeu de test. Les résultats de cette expérience sont disponibles dans le Tableau 2, et les meilleures performances sont obtenues pour une valeur de $\lambda = 10$.

λ	$\uparrow$ *Précision*	$\downarrow$ Erreur de reconstruction
0.1	0.984	0.0069
1	0.984	0.0028
10	0.981	0.0012
100	0.979	0.0011

TAB. 2 – *Précision et erreur de reconstruction pour chaque λ sur le jeu de données de test*

La fonction d'erreur de classification est une *categorical cross-entropy* et la fonction d'erreur de reconstruction est une erreur quadratique moyenne. L'entraînement est effectué avec 128 exemples par batch, pour 25 époques, avec un optimiseur de type Adam.

Autoencodeur *baseline* L'architecture de l'autoencodeur *baseline* est la même que celle de l'autoencodeur supervisé mais sans les couches de classification. La fonction d'erreur de reconstruction est une erreur quadratique moyenne, l'entraînement est effectué avec 128 exemples par batch, pour 18 époques, avec un optimiseur de type Adam. Notre modèle atteint une erreur de reconstruction de 0.0016 sur le jeu de donnée de test.

Hyperparamètres pour la construction des contrefactuels Les valeurs d'hyperparamètres sont fixées selon le paramétrage utilisé par Van Looveren et Klaise (2021) ($\gamma = 100$, $\tau = 0$, $c = 1$, $\beta = 0.1$, $\theta = 100$, $K = 5$).

Autoencodeur pour l'évaluation de métriques Nous avons utilisé la même architecture que Van Looveren et Klaise (2021). Le jeu d'entraînement est le même que celui utilisé pour l'entraînement des autoencodeurs *baseline* et supervisé.

Expérience sur données tabulaires Nous avons évalué nos résultats sur des données tabulaires numériques obtenues à partir de la fonction $make\_blobs$ de sklearn. Nous avons généré est un problème de classification à 6 classes à partir d'une mixture de 6 gaussiennes, chacune ayant un écart-type de 10 (une valeur choisie volontairement élevée afin de tester la capacité de l'autoencodeur supervisé à séparer les classes par rapport à l'autoencodeur baseline). Notre jeu de données contient $50,000$ exemples, dont $30,000$ en apprentissage, $10,000$ en validation et $10,000$ en test. Un échantillon aléatoire de $1,000$ exemples de test a été sélectionné pour la génération des contrefactuels. Les résultats d'évaluation des métriques de la section 4.1 sont reportés dans le Tableau 3. On observe ici également une amélioration du gain de prédiction au détriment de l'actionnabilité pour la solution basée sur l'autoencodeur supervisé.

Métriques	*Baseline*	Autoencodeur supervisé
↑ Gain de prédiction	0.442±0.11	**0.736±0.117**
↓ Réalisme	**0.021±0.009**	0.035±0.056
↓ Actionnabilité	**1.902±0.833**	2.851 ±2.699

TAB. 3 – *Comparaison des résultats pour les différentes métriques des données numériques.*

Summary

In this work, we investigate the problem of generating counterfactuals explanations that are both close to the data distribution, and to the distribution of the target class. Our objective is to obtain counterfactuals with likely values (*i.e.* realistic). We propose a method for generating realistic counterfactuals by using class prototypes. The novelty of this approach is that these class prototypes are obtained using a supervised auto-encoder. Then, we performed an empirical evaluation across several interpretability metrics, that shows competitive results with a state-of-the-art method.