XAI - TME 2

Explication par génération d'exemples contre-factuels divers

L'objectif du TME est de mettre en œuvre la méthode de génération d'exemples contre-factuels divers DiCE, proposée dans l'article

Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. *Proc. of the 2020 Conf. on Fairness, Accountability, and Transparency, FAT* 20*, pp. 607-617. 2020

Les auteurs en fournissent une implémentation comme indiqué dans le premier point ci-dessous.

1. **Installation**

   Utiliser les commandes

   `pip install -U --user dice-ml` ou `conda install -c conda-forge dice-ml`

   Le fichier python doit alors contenir dans son en-tête `import dice_ml`

2. **Données**

   Utiliser d'abord les données `half_moons`, en dimension 2, qui permettent une visualisation des résultats, puis des données classiques (voir Q5).

   DiCE considère que les données sont stockées dans un `DataFrame`, qui est ensuite transformé en une structure interne par la commande suivante

   `data = dice_ml.Data(dataframe, continuous_features, outcome_name)`

   `continuous_features` est une liste de chaînes de caractères correspondant à des noms d'attributs dans le `DataFrame`, permettant de spécifier les attributs numériques.

   `outcome_name` est une chaîne de caractères donnant le nom de l'attribut de classe.

3. **Classifieurs**

   DiCE est une méthode d'explication post-hoc, qui prend donc en entrée un classifieur pré-entraîné, de type

   `trained_classifier = clf.fit(x_train, y_train)`

   Ce classifieur est ensuite chargé au format DiCE par la commande

   `model = dice_ml.Model(model=trained_classifier, backend)`

   `backend` est une chaîne de caractères valant `sklearn`, `TF1` (pour TensorFlow 1.x), `TF2` (pour TensorFlow 2.x) ou `PYT` (pour PyTorch).

4. **Génération d'ensembles contre-factuels**

   Un expliqueur est alors initialisé par la commande
   `explainer = dice_ml.Dice(data, model)`

   Si le modèle provient de sklearn, il faut de plus préciser un champ `method` qui indique comment les exemples contre-factuels sont identifiés : elle peut prendre pour valeur la chaîne de caractères `random`, `genetic` ou `kdtree`. Sinon [1], DiCE identifie les exemples contre-factuels en optimisant de la fonction de coût présentée dans les transparents.

   DiCE permet ensuite de générer des explications par la fonction suivante, qui permet de spécifier, si souhaité, de nombreux paramètres :

```
counterfactuals = explainer.generate_counterfactuals(
                      query_instance,
                      total_CFs,
                      desired_class="opposite",
                      proximity_weight=0.5,
                      diversity_weight=1.0,
                      features_to_vary="all",
                      permitted_range=None,
                      posthoc_sparsity_param=0.1)
```

1. voir par exemple https ://www.kaggle.com/code/autuanliuyc/logistic-regression-with-tensorflow

`features_to_vary` est une liste de chaînes de caractères correspondant aux noms des attributs qui peuvent être modifiés. `permitted_range` est un dictionnaire dont les clés sont des noms d'attributs numériques et les valeurs associées des intervalles

5. **Récupération des exemples contre-factuels générés**

   On peut enfin visualiser les exemples générés dans un data frame :

   `counterfactuals.visualize_as_dataframe()`

   ou y accéder par `counterfactuals.cf_examples_list[0].final_cfs_df`

6. **Visualisation et expérimentations**

   Visualiser, comme pour l'algorithme Growing Spheres, dans le cas des `half_moons`, les données d'apprentissage, la frontière de décision, la requête, l'ensemble des contre-factuels générés.

   Examiner les résultats obtenus en faisant varier l'exemple requête, les paramètres ou le classifieur.

   Implémenter des critères d'évaluation numériques, incluant par exemple la validité (proportion de candidats générés qui sont effectivement de la classe souhaité), la proximité et la parcimonie. Observer leur évolution lorque les paramètres de la méthode varient.

   Utiliser également les jeux de données classiques, en dimension supérieure à 2.

7. **Extension de Growing Spheres**

   Proposer une variante de GS permettent à l'utlisateur de spécifier des attributs non modifiables et/ou des plages de variation autorisées