Utilização da técnica de *Algorithmic recourse* para classificação linear

Leonardo Santos Miranda[1]
[1]Departamento de Ciência da Computação - UFMG

1693 Words

Student ID:2020726313

Keywords: algorithmic recourse, interpretabilidade, aprendizagem de máquina

2 de abril de 2021

---

1 Objetivo

No contexto de Aprendizagem de Máquina, a técnica de *Recourse* é a habilidade de uma pessoa obter uma saída desejada de um modelo automatizado [1]. *Algorithmic recourse* é uma técnica que busca explicar a saída obtida pelo modelo, focando nas ações que um indivíduo pode realizar para reverter o resultado, possivelmente adverso. Dessa forma, o objetivo desse projeto é entender o quão diferentes características podem ser mudadas de maneira que afete a saída do modelo, analisando o custo e a viabilidade do *Recourse* sobre a base de dados.

2 Problema, contexto e motivação

Através de intervenções podemos imaginar como o mundo seria caso diferentes decisões tivessem sido tomadas, por exemplo: "Seria aquele paciente sido curado se tivessemos aplicado o tratamento X ao invés do tratamento Y?" [2]. A técnica de *Algorithmic Recourse* possui a motivação em justamente tentar fornecer explicações e recomendações que são raramente consideradas por um modelo totalmente automatizado de aprendizagem de máquina.

Sistemas que fazem o uso de algoritmos de aprendizagem de máquina estão sendo cada vez mais utilizados em áreas mais sensíveis, como a medicina, onde uma tomada de decisão pode impactar diretamente a vida de um certo indivíduo [3]. Por este motivo, além dos modelos de aprendizagem de máquina precisarem serem cada vez mais robusto e preciso, valores socialmente relevantes como privacidade, responsabilidade e explicabilidade desempenham um papel importante para a adoção e impacto da tecnologia e também devem ser fortemente considerados, o que acaba sendo um dos maiores problemas destes modelos automatizados. Buscar minimizar esse problema através da busca de explicações dos resultados destes modelos de aprendizagem de máquinas é o objetivo da técnica de *Recourse* proposta nesse projeto.

Um outro exemplo de uso da técnica de *Algorithmic Recourse*, além da medicina: "Imagine que um sujeito gostaria de pedir empréstimo a um banco para comprar uma nova casa, o banco então utiliza um algoritmo de aprendizagem de máquina para a classificação binária e rejeita o sujeito" [4]. Através dessa situação hipotética, podemos nos perguntar: "Por que o sujeito foi rejeitado para o empréstimo? Qual atributo do sujeito causou a rejeição?". O *Algorithmic Recourse* tenta buscar explicações ou recomendações dos resultados obtidos através de um sistema automático.

3 Dados e técnicas

Para o projeto proposto, será utilizado a técnica de *Algorithmic recourse*, buscando reverter um resultado possívelmente adverso e entender quais mudanças podem ocorrer através de possíveis interferências em suas características. Nesse projeto foi escolhido uma base de dados pública sobre cancer de mama, disponível no Kaggle pelo link https://www.kaggle.com/merishnasuwal/breast-cancer-prediction-dataset. A base de dados está no formato tabular (.csv) e possui características de um caroço suspeito do câncer de mama., como o raio médio, textura média, pêrimetro médio, área média e a suavidade média. Utilizaremos a técnica de *Recourse* em cima dessa base de dados, analisando a viabilidade e o custo da aplicação de *recourse*. Para isso, utilizaremos a biblioteca Actionable Recourse disponibilizada para uso gratuitamente no github pelo link http://github.com/ustunb/actionable-recourse. Um maior detalhe e exemplo de uso desta biblioteca pode ser encontrado no artigo *Actionable recourse in linear classification* [1].

4 Objetivos Específicos

Utilizar a biblioteca de Actionable Recourse na linguagem de programação Python, treinando um modelo de classificação linear, realizando alterações nos atributos de um caroço suspeito através da técnica de *Recourse*, analisando o comportamento das predições através das mudanças que foram realizadas e estimando tanto a viabilidade tanto o custo das possíveis mudanças, utilizando as ferramentas da biblioteca anteriormente descrita.

5 Trabalhos Relacionados

Dentre os trabalhos relacionados estudados, o trabalho entitulado "Actionable recourse in linear classification" [1] foi a base para este projeto. Neste trabalho, é descrito os detalhes e procedimentos que devem ser seguidos ao aplicar a técnica de *Recourse* em dados tabulares, além de fornecer uma biblioteca na linguagem de programação Python que possui ferramentas gratuítas para o uso de *Recourse*, que foi utilizada nesse projeto. Além deste trabalho, o artigo "Causality for Machine Learning [5] foi de grande relevância para o estudo de inferência causal no âmbito de Machine Learning. Os trabalhos foram de igual importância para o entendimento do tema proposto. Por fim, o trabalho "The seven tools of causal inference, with reflections on machine learning" [6] foi de suma importância para o entendimento deste projeto, explicando de maneira detalhada os pontos mais importantes que devem ser reforçados ao praticar inferência causal.

6 Metodologia

Como dito anteriormente o objetivo do *Recourse* é realizar mudanças nos atributos da base de dados de forma que o modelo obtenha a saída desejada pelo usuário. Ao aplicar a técnica de *Recourse*, portanto, apenas as caracteristicas acionáveis são alteradas e algumas restrições discretas são consideradas: [1]

- Atributos imutáveis não podem ser alterados e portanto não são consideradas no *Recourse* (Exemplo: graduado $\rightarrow$ True não pode ser mudado para graduado $\rightarrow$ False).
- As variáveis mudáveis não sofreram mudaças que sejam inviáveis (Exemplo: qtde-cartao-credito 5 $\rightarrow$ 0.5)

Considerando as restrições, as seguintes ferramentas foram criadas pela biblioteca para a realização do *Recourse*:

1. Um procedimento para avaliar a viabilidade e a dificuldade do *Recourse* de um classificador linear sobre sua população alvo. Dado um classificador e uma amostra de características da população alvo, o procedimento dessa biblioteca irá estimar a viabilidade e a dificuldade do *Recourse* na população solucionando um problema de optimização para cada ponto que recebeu um resultado não desejavél.

2. Um método para gerar uma lista de mudanças acionáveis para um usúario obter a saída desejada de um classificador linear, essa lista é referida como *flipset* e pode ser vista na figura abaixo:

FEATURES TO CHANGE	CURRENT VALUES		REQUIRED VALUES
n_credit_cards	5	$\longrightarrow$	3
current_debt	$3,250	$\longrightarrow$	$1,000
has_savings_account	FALSE	$\longrightarrow$	TRUE
has_retirement_account	FALSE	$\longrightarrow$	TRUE

Tanto o custo e a viabilidade do *Recourse* são garantidas e provadas no artigo Actionable Recourse in Linear Classification [1].

7 Detalhes da Implementação

A implementação será feita em Python utlizando os recursos da biblioteca de *Actionable Recourse*, como anteriormente descrito. Primeiramente, vai ser realizado um breve estudo sobre cada *feature* da base de dados para definir se há algum atributo imutável ou se há algum limite ou intervalo que precisa ser pré-determinado antes de realizar o *Recourse*. Para isso, um pequeno trecho de código que realiza a leitura da base de dados e exibe um resumo dos dados foi executado:

```
1  import pandas as pd
2  import numpy as np
3  import recourse as rs
4
5  # import data
6  url = 'https://www.kaggle.com/merishnasuwal/breast-cancer-prediction-dataset/download'
7  df = pd.read_csv('Breast_cancer_data.csv')
8  y, X = df.iloc[:, -1], df.iloc[:, 0:-1]
9  print (X)
```

```
     mean_radius  mean_texture  mean_perimeter  mean_area  mean_smoothness
0          17.99         10.38          122.80     1001.0          0.11840
1          20.57         17.77          132.90     1326.0          0.08474
2          19.69         21.25          130.00     1203.0          0.10960
3          11.42         20.38           77.58      386.1          0.14250
4          20.29         14.34          135.10     1297.0          0.10030
..           ...           ...             ...        ...              ...
564        21.56         22.39          142.00     1479.0          0.11100
565        20.13         28.25          131.20     1261.0          0.09780
566        16.60         28.08          108.30      858.1          0.08455
567        20.60         29.33          140.10     1265.0          0.11780
568         7.76         24.54           47.92      181.0          0.05263

[569 rows x 5 columns]
```

Na base de dados escolhidas, temos um total de 5 atributos que possui informações de um caroço suspeito:

- mean_radius: média das distâncias do centro aos pontos do pêrimetro.
- mean_texture: desvio padrão dos valores da escala de cinza.
- mean_perimeter: tamanho médio do tumor central.
- mean_area: área média do tumor central.
- mean_smoothness: média de variação local em comprimentos de raio.

Analisando as caracteristicas, percebomos que não há nenhuma variável imutável o que significa que todas as *features* podem ser alteradas, de alguma forma. O objetivo da classificação é o que chamamos de váriavel y e nesse caso, corresponde ao diagnóstico do caroço suspeito:

- se a váriavel y possui o valor de 0, o caroço é benigno.
- se a variavel y possui o valor de 1, o caroço é maligno.

Dito isso, o próximo passo é treinar um classificador simples que faça a predição do caroço suspeito. Para isso, utilizaremos o algoritmo de Regressão Logística, dísponivel na biblioteca Sckit-learn para essa tarefa, visto que a nossa base de dados é binária e este é um ótimo algorito para tal classificação. O seguinte trecho de código em Python realiza o procedimento de treinamento do modelo:

```
11  # Treinando um classificador
12  from sklearn.linear_model import LogisticRegression
13  clf = LogisticRegression(max_iter = 1000)
14  clf.fit(X, y)
15  yhat = clf.predict(X)
```

Agpra que temos um classificador treinado com a base de dados, o passo seguinte é personalizar um conjunto de ações para as mudanças – nessa etapa será definido quais são as váriaveis mudáveis e o limite inferior e superior a ser considerado nas mudanças. Dessa forma, utilizaremos as ferramentas da biblioteca de *Recourse* para configurar um limite padrão para as

mudanças – no nosso caso, todos os atributos são numéricos e analisando os dados podemos definir os seguintes limites inferiores e superiores para cada atributo através do seguinte trecho de código:

```
# Cálculo de viabilidade e custo das mudanças que ocorreram
auditor = rs.RecourseAuditor(A, coefficients = clf.coef_, intercept = clf.intercept_)
audit_df = auditor.audit(X)
print("Viabilidade: ",audit_df['feasible'].mean()) # Viabilidade do recourse
print("Custo: ",audit_df['cost'].mean()) # Custo do recourse
```

Foi definido que todas as váriaveis do conjunto serão mudáveis, através da linha 19 e um limite superior de 50 para os atributos meam_radius e mean_texture, 3000 para mean_area e 300 para mean_perimeter. O limite inferior não será menor do que 0 para todas os atributos.

Agora que o conjunto de ações a serem realizados foram definidos, iremos construir uma tabela (*flipset) com 10 amostras da base de dados que possui a predição não desejada – no nosso caso, quando a variável y é igual a 0 (câncer benigno). Esta tabela gerada será utilizada para estimar a viabilidade e o custo do Recourse posteriormente. O trecho de código seguinte realiza tal tarefa:*

```
27 # Construção da tabela flipset para um câncer benigno
28 i = np.flatnonzero(yhat <= 0).astype(int)[0]
29 fs = rs.Flipset(x = X.values[i], action_set = A, clf = clf)
30 fs.populate(enumeration_type = 'distinct_subsets', total_items = 10)
31 fs.to_html()
```

Finalmente, utilizaremos mais uma função da biblioteca para o cálculo da viabilidade e custo das mudanças realizadas pelo Recourse – de forma resumida, esta função irá estimar o custo e a viabilidade média das mudanças que foram realizadas na base de treinamento, comparando com a tabela flipset que criamos anteriormente para o cálculo de viabilidade. Esta tarefa é feito pelo seguinte trecho de código:

```
33 # Cálculo de viabilidade e custo das mudanças que ocorreram
34 auditor = rs.RecourseAuditor(A, coefficients = clf.coef_, intercept = clf.intercept_)
35 audit_df = auditor.audit(X)
36 print(audit_df['feasible'].mean()) # Viabilidade do recourse
37 print(audit_df['cost'].mean()) # Custo do recourse
```

Executando o trecho de código acima, obtemos o resultado do Recourse realizado em cima da base de dados proposta e discutiremos os mesmos na próxima seção desse trabalho.

8 Análise dos Resultados

Tanto a viabilidade e o custo da realização do Recourse foram obtidos utilizando as ferramentas da biblioteca e pode ser visualizado abaixo:

```
Viabilidade:  1.0
Custo:  0.022574786538991747
```

Com a viabilidade de 1.0 – o que significa que as mudanças são viáveis pela biblioteca, e um custo baixo de aproximadamente 0.02, a técnica de Recourse se demonstrou viável para as mudanças das features numéricas propostas para um contexto computacional, mas não podemos dizer com clareza se os resultados são viáveis para um contexto geral – teriamos que ter a ajuda de um especialista na área de câncer de mama para analisar as mudanças que foram feitas pelo algoritmo de Recourse.

9 Conclusões e Perspectivas

Primeiramente a proposta desse projeto teve que ser mudada em comparação a primeira versão que eu enviei no moodle pela dificuldade e complexidade de aplicar a técnica de Recourse em imagens médicas – aquela proposta era o meu tema de Mestrado. Para não mudar muito, eu escolhi usar uma base de dados tabular ao invés de imagens e continuando ainda na área médica. Foi escolhido o tema de Actionable Recourse como sugestão do meu orientador visto que também é o tema do meu Mestrado e realizar esse projeto no mesmo tema iria me ajudar a entender melhor o problema de Recourse. Espero que o estudo realizado nesse trabalho e a técnica escolhida seja relevante para o projeto da disciplina.

10 Referências

1. Ustun, B., Spangher, A. & Liu, Y. *Actionable recourse in linear classification* em *Proceedings of the Conference on Fairness, Accountability, and Transparency* (2019), 10–19.
2. Prosperi, M., Guo, Y., Sperrin, M., Koopman, J. S., Min, J. S., He, X., Rich, S., Wang, M., Buchan, I. E. & Bian, J. Causal inference and counterfactual prediction in machine learning for actionable healthcare. *Nature Machine Intelligence* **2,** 369–375 (2020).
3. Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., Van Der Laak, J. A., Van Ginneken, B. & Sánchez, C. I. A survey on deep learning in medical image analysis. *Medical image analysis* **42,** 60–88 (2017).
4. Karimi, A.-H., Barthe, G., Schölkopf, B. & Valera, I. A survey of algorithmic recourse: definitions, formulations, solutions, and prospects. *arXiv preprint arXiv:2010.04050* (2020).
5. Schölkopf, B. Causality for machine learning. *arXiv preprint arXiv:1911.10500* (2019).
6. Pearl, J. The seven tools of causal inference, with reflections on machine learning. *Communications of the ACM* **62,** 54–60 (2019).