人工智能实践第一次实验报告

20337221 马浩铭

实验过程

1 完成 `estimate_causal_effect` 函数

如图所示

Estimation of ATE

True ATE: $\mathbb{E}[Y(1) - Y(0)] = 1.05$

Identification: $\mathbb{E}[Y(1) - Y(0)] = \mathbb{E}_X\left[\mathbb{E}[Y \mid T = 1,\ X] - \mathbb{E}[Y \mid T = 0,\ X]\right]$

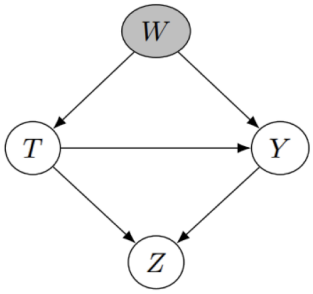
Estimation: $\frac{1}{n}\sum_i [\underbrace{\mathbb{E}[Y \mid T = 1, X = x_i]}_{\text{Model (linear regression)}} - \underbrace{\mathbb{E}[Y \mid T = 0, X = x_i]}_{\text{Model (linear regression)}}]$

Estimates:

$X = \{\}$ (naive): 5.33 $\quad \frac{|5.33 - 1.05|}{1.05} \times 100\% = 407\%$ error

$X = \{W, Z\}$ (last week): 0.85 $\quad$ 19% error

$X = \{W\}$ (unbiased): 1.0502 $\quad$ 0.02% error



按照该公式首先对集合X中不同变量分类，我这里直接进行去重操作。

去重完成后，每一项都是一个xi，直接使用模型预测、计算ATE即可。

最终函数如下：

```
def estimate_causal_effect(Xt, y, model=LinearRegression(), treatment_idx=0,regression_coef=False):
    # TODO 1: 完成estimate_causal_effect函数
    model.fit(Xt, y)
    if regression_coef:
        # TODO
        Xt_ = pd.DataFrame.copy(Xt)
        Xt_.drop_duplicates(inplace=True)
        Xt1 = pd.DataFrame.copy(Xt_)
        Xt0 = pd.DataFrame.copy(Xt_)
        Xt1[Xt.columns[treatment_idx]] = 1
        Xt0[Xt.columns[treatment_idx]] = 0

        return sum(model.predict(Xt1) - model.predict(Xt0)) / len(Xt0)
    else:
        Xt1 = pd.DataFrame.copy(Xt)
        Xt1[Xt.columns[treatment_idx]] = 1
        Xt0 = pd.DataFrame.copy(Xt)
        Xt0[Xt.columns[treatment_idx]] = 0
        # TODO
        return sum(model.predict(Xt1) - model.predict(Xt0)) / len(Xt0)
```

2 仅以age作为调整集进行估计，并说明理由

从上课课件中可知，不能以干预变量的后代作为条件计算条件概率，因为这可能会阻塞因果关系，可能会引入额外的相关关系(Collider bias)。

实际计算结果验证了理论，如下图所示。结果只用age作为调整集效果远远好于使用所有的变量。

```
# Regression Coefficient Estimates #
Naive ATE estimate:                                          5.3285016809133054
ATE estimate adjusting for all covariates:       0.8537946429664504
ATE estimate adjusting for age:                          1.050212454041891


### Continuous Treatment Data ###

# Adjustment Formula Estimates #
Naive ATE estimate:                                          3.6283781967307194
ATE estimate adjusting for all covariates:       0.8532920320728831
ATE estimate adjusting for age:                          1.0497716560864117

# Regression Coefficient Estimates #
Naive ATE estimate:                                          3.6283781967307194
ATE estimate adjusting for all covariates:       0.8532920320728831
ATE estimate adjusting for age:                          1.0497716560864117
```

3 条件结果模型存在的问题，以及进一步的改进方案

将Treatment(即探求是否为原因的变量，本题中为sodium)作为参数，在训练的过程中只占有一个维度，效果可能被更多维度的其余参数盖过。

一种解决方案就是使用GCOM，对不同的Treatment训练不同的模型，直接分割开两种不同的模型训练。

当然也可以使用TODO 4 中的TARNet与X-Learner。

最终，模仿第二步中的函数，函数实现如下：

```
def estimate_causal_effect_with_GCOM(Xt, y, model1=LinearRegression(), model2=LinearRegression(), treatment_idx=0,regre
    Xt_1 = Xt[Xt[Xt.columns[treatment_idx]].isin([1])]
    y_1 = y[Xt[Xt.columns[treatment_idx]].isin([1])]
    Xt_0 = Xt[Xt[Xt.columns[treatment_idx]].isin([0])]
    y_2 = y[Xt[Xt.columns[treatment_idx]].isin([0])]
    # TODO 3: 使用GCOM(Grouped Conditional Outcome Modeling)进行估计
    model1.fit(Xt_1, y_1)
    model2.fit(Xt_0, y_2)
    if regression_coef:

        Xt_ = pd.DataFrame.copy(Xt)
        Xt_.drop_duplicates(inplace=True)
        Xt1 = pd.DataFrame.copy(Xt_)
        Xt0 = pd.DataFrame.copy(Xt_)
        Xt1[Xt.columns[treatment_idx]] = 1
        Xt0[Xt.columns[treatment_idx]] = 0

        return sum(model1.predict(Xt1) - model2.predict(Xt0)) / len(Xt0)
    else:
        Xt1 = pd.DataFrame.copy(Xt)
        Xt1[Xt.columns[treatment_idx]] = 1
        Xt0 = pd.DataFrame.copy(Xt)
        Xt0[Xt.columns[treatment_idx]] = 0

        return sum(model1.predict(Xt1) - model2.predict(Xt0)) / len(Xt0)
```

4 TARNet, X-Learner

由于TARNet的训练需要一个分层模型，所以这里就只实现X-Learner了。

实现如下：

```
def estimate_causal_effect_with_X_Learner(Xt, y, model1=LinearRegression(), model2=LinearRegression(), model3=LinearReg
    Xt_1 = Xt[Xt[Xt.columns[treatment_idx]].isin([1])]
    y_1 = y[Xt[Xt.columns[treatment_idx]].isin([1])]
    Xt_0 = Xt[Xt[Xt.columns[treatment_idx]].isin([0])]
    y_2 = y[Xt[Xt.columns[treatment_idx]].isin([0])]
    # TODO 3: 使用GCOM(Grouped Conditional Outcome Modeling)进行估计
    model1.fit(Xt_1, y_1)
    model2.fit(Xt_0, y_2)

    d1 = y_1 - model2.predict(Xt_1)
    d0 = model1.predict(Xt_0) - y_2

    model3.fit(Xt_1, d1)
    model4.fit(Xt_0, d0)


    rt =  gx * sum(model3.predict(Xt_1) + (1 - gx) *  model4.predict(Xt_1)) / len(Xt_1) - gx * sum(model3.predict(Xt_0)

    return rt
```

真正实用中，需要将参数gx替换为其他参数，这里我就不去调整了，仅尝试实现原理。