UNIVERSITAT DE BARCELONA

CAUSAL INFERENCE AND MACHINE LEARNING

About the course

01 **Introduction**
Observational and Interventional Distributions

02 **Potential Outcomes**
Fundamental Problem of Causal Inference

03 **Causal Graphs**
Do Calculus

04 **Estimand-based Estimation**
Metalearners

05 **Estimand-agnostic Estimation**
Counterfactuals

06 **Causal Machine Learning**
Supervised and Reinforcement Learning

07 **Practical Causal Inference**
Exercises

The relationship between causality and artificial intelligence can be seen from two points of view: how causality can help solve some of the current problems of AI and how causal inference can leverage machine learning techniques. In this course we will review the two points of view with special emphasis on examples and practical cases.

04 - Estimand-based Estimation

Meta learners

In this section we'll see

- Causal effects estimation under the backdoor adjustment
- Estimation using meta learners
- Propensity score methods
- Deep Learning methods
- A real case application example by Netflix

Preliminaries and notation

- $ITE = Y_i(1) - Y_i(0)$
- $ATE = \mathbb{E}[Y_i(1) - Y_i(0)]$
- $CATE = \mathbb{E}[Y_i(1) - Y_i(0) \mid X = x]$

T:	Observed treatment
Y:	Observed outcome
i:	Specific individual subscript
$Y_i(1)$:	Outcome under treatment
$Y_i(0)$:	Outcome under no treatment
X	Vector of covariates

ITEs (Individual Treatment Effect) are the most specific kind of cause effects. They are almost always impossible to estimate, as it would require very strong assumptions or being able to observe multiple treatments for the same individual

Preliminaries and notation

- $ITE = Y_i(1) - Y_i(0)$
- $ATE = \mathbb{E}[Y_i(1) - Y_i(0)]$
- $CATE = \mathbb{E}[Y_i(1) - Y_i(0) \mid X = x]$

T:	Observed treatment
Y:	Observed outcome
i:	Specific individual subscript
$Y_i(1)$:	Outcome under treatment
$Y_i(0)$:	Outcome under no treatment
X	Vector of covariates

ATE (Average Treatment Effect) is the average over ITEs. It is the objective of most AB tests and reflects the effects of a treatment in a certain population.

Preliminaries and notation

- $ITE = Y_i(1) - Y_i(0)$
- $ATE = \mathbb{E}[Y_i(1) - Y_i(0)]$
- $CATE = \mathbb{E}[Y_i(1) - Y_i(0) \mid X = x]$

T:	Observed treatment
Y:	Observed outcome
i:	Specific individual subscript
$Y_i(1)$:	Outcome under treatment
$Y_i(0)$:	Outcome under no treatment
X	Vector of covariates

CATE (Conditional Average Treatment Effect) are the average effects when conditioning for a vector of covariates X. CATE is used when we want to estimate causal effects more individualised than the ATE.
If X contains all the exogenous variables that have a causal association to Y, CATE = ITE.

Remember - Backdoor Adjustment

Backdoor Criterion: A set of variables X satisfies the backdoor criterion relative to T and Y if the following are True:

1. X blocks all backdoor paths from T to Y
2. X does not contain any descendants of T

If X satisfies the Backdoor Criterion:

$$\mathbb{E}[Y(1)] - \mathbb{E}[Y(0)] = \mathbb{E}_X[\mathbb{E}[Y \mid Y = 1, X] - \mathbb{E}_X[\mathbb{E}[Y \mid Y = 0, X]]$$

Remember - Backdoor Adjustment

Backdoor Criterion: A set of variables X satisfies the backdoor criterion relative to T and Y if the following are True:

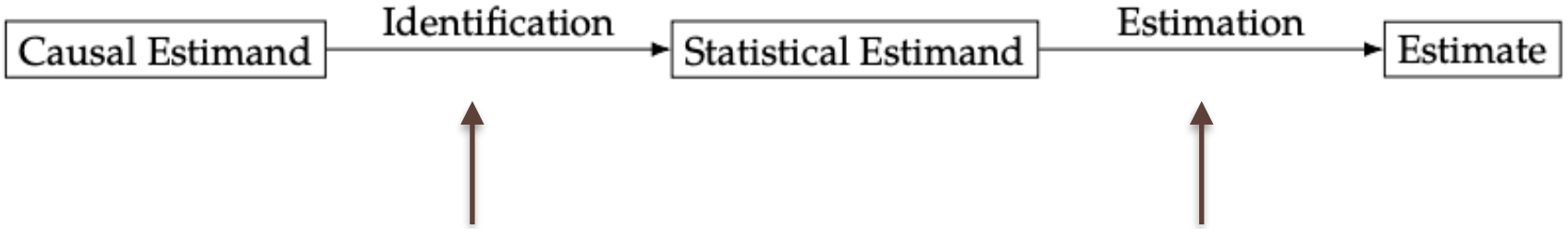
1. X blocks all backdoor paths from T to Y
2. X does not contain any descendants of T

If X satisfies the Backdoor Criterion:

$$\mathbb{E}[Y(1)] - \mathbb{E}[Y(0)] = \mathbb{E}_X[\mathbb{E}[Y \mid Y = 1, X] - \mathbb{E}_X[\mathbb{E}[Y \mid Y = 0, X]]$$

Note that descendants of Y are not to be included in the set of X as they are descendants of T

Remember - Estimand based Causal Inference Workflow



Assumptions,
Backdoor,
Frontdoor,
[…]

In this section we will deal with the estimation phase of the Causal Inference Workflow when the identification method is the **Backdoor Criterion**

Why backdoor: 1 - Its a usual scenario

- Most real life Causal Inference problems fall into a scenario that can be identified using the backdoor criterion
- After applying the backdoor adjustment, the statistical estimand we obtain can be estimated using all the classical methods in machine learning and statistics

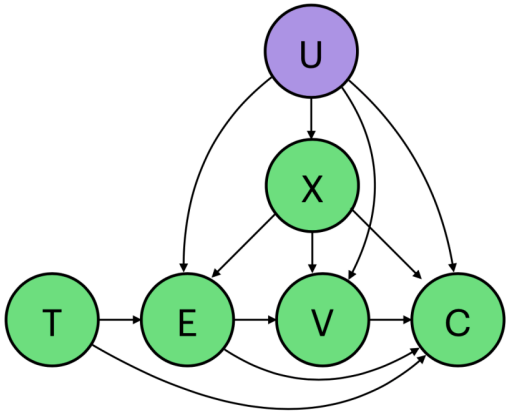


Figure 2: Causal graph of the online advertising system

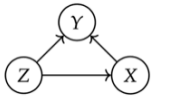
Example: Proposed Causal Graph in the Criteo benchmark dataset. X are user attributes, E are clicks, V are visits and C are conversions. U are potential unobserved confounders
https://ailab.criteo.com/criteo-uplift-prediction-dataset/

Why backdoor: 2 - Other methods require tailored estimation

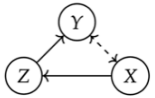
Answering the query

→ However, depending on the graph, even the same query results in different estimands.

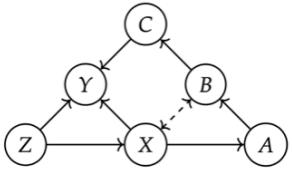
→ $P(Y \mid do(X = x))$:



$\mathbb{Q} = \mathbb{E}_Z[P(Y \mid x, Z)]$



$\mathbb{Q} = \mathbb{E}_{Z|x}[\mathbb{E}_X[P(Y \mid X, Z)]]$



$\mathbb{Q} = \mathbb{E}_{Z,C}[\frac{P(Y \mid x, Z, C)}{P(Z, C)} \cdot \mathbb{E}_{A|x}[\mathbb{E}_X[P(Z, C \mid X, A)]]]$

Estimation using meta learners

Meta Learners

→ Meta learners are discrete treatment **CATE estimators** that that can take advantage of any supervised learning or regression method in machine learning and statistics

Meta Learners

- Meta learners are discrete treatment **CATE estimators** that that can take advantage of any supervised learning or regression method in machine learning and statistics
- They build on base algorithms such as Random Forests or Gradient Boosted Trees to estimate CATE, thus being able to leverage their strengths

Meta Learners

- Meta learners are discrete treatment **CATE estimators** that that can take advantage of any supervised learning or regression method in machine learning and statistics
- They build on base algorithms such as Random Forests or Gradient Boosted Trees to estimate CATE, thus being able to leverage their strengths
- Meta learners rely on the unconfoundedness assumption, assuming that X is a sufficient adjustment set. In other words, assuming it satisfies the **backdoor criterion**

Estimation: SLearner

S-Learner

The S-Learner models $Y(0)$ and $Y(1)$ through one model that receives the treatment assignment T as an input feature (along with the features X). The estimated CATE is given by:

$$\begin{aligned}\hat{\tau}(x) &= E[Y \mid X = x, T = 1] - E[Y \mid X = x, T = 0] \\ &= \hat{\mu}(x, 1) - \hat{\mu}(x, 0)\end{aligned}$$

where $\hat{\mu} = M(Y \sim (X, T))$ is the outcome model for features X, T. Here, M is any suitable machine learning algorithm.

https://econml.azurewebsites.net/

Estimation: TLearner

T-Learner

The T-Learner models $Y(0)$, $Y(1)$ separately. The estimated CATE is given by:

$$\begin{aligned}\hat{\tau}(x) &= E[Y(1) - Y(0) \mid X = x] \\ &= E[Y(1) \mid X = x] - E[Y(0) \mid X = x] \\ &= \hat{\mu}_1(x) - \hat{\mu}_0(x)\end{aligned}$$

where $\hat{\mu}_0 = M_0(Y^0 \sim X^0)$, $\hat{\mu}_1 = M_1(Y^1 \sim X^1)$ are the outcome models for the control and treatment group, respectively. Here, M_0, M_1 can be any suitable machine learning algorithms that can learn the relationship between features and outcome.

https://econml.azurewebsites.net/

Model properties intuitions

- SLearner uses the treatment T as a covariate, so in cases where the number of variables is high, it's possible the model isn't making any use of it.
- Due to this, **SLearner** has a **bias for small treatment effects**.

Model properties intuitions

- SLearner uses the treatment T as a covariate, so in cases where the number of variables is high, it's possible the model isn't making any use of it.
- Due to this, **SLearner** has a **bias for small treatment effects**.
- **TLearner** models treatment and control group separately, so we have **less data to train each model**.
- In general, TLearner works well when treatment and control follow very different distributions

Example applying SLearn, TLearn - Data Generation

This contains an unrelated treatment arm and control arm, with input data generated by $X_i \sim N(0, I_{d \times d})$.

The treatment flag is a binomial variable whose d.g.p. is:

$$P(W_i = 1|X_i) = \frac{1}{1+\exp -X_{i1}+\exp -X_{i2}}$$

The outcome variable is:

$$y_i = \frac{1}{2}\big(max(X_{i1} + X_{i2} + X_{i3}, 0) + max(X_{i4} + X_{i5}, 0)\big) + (W_i - 0.5)(max(X_{i1} + X_{i2} + X_{i3}, 0) - max(X_{i4}, X_{i5}, 0))$$

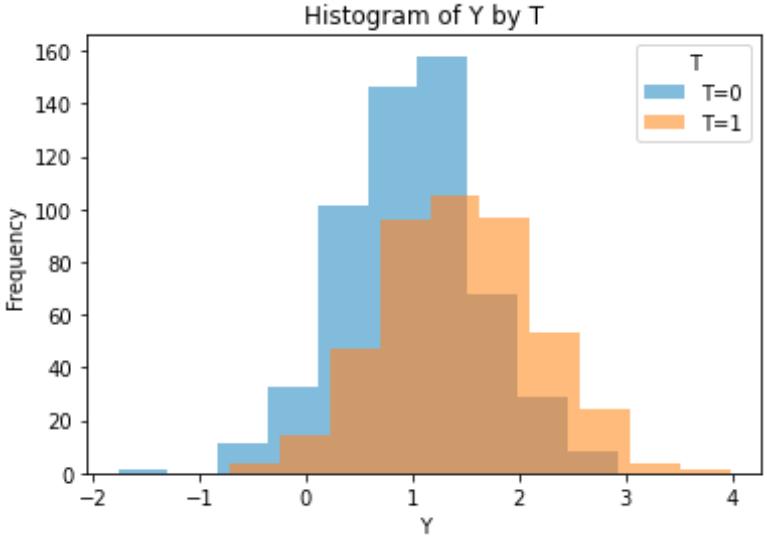
https://causalml.readthedocs.io/

Example applying SLearn, TLearn - Data Generation

```
X = (
  np.random.uniform(size=n * p) + np.random.normal(0, X_noise, size=n * p)
).reshape((n, -1))

b = (
    np.maximum(np.repeat(0.0, n), X[:, 0] + X[:, 1] + X[:, 2])
    + np.maximum(np.repeat(0.0, n), X[:, 3] + X[:, 4])
) / 2
e = 1 / (1 + np.exp(-X[:, 0]) + np.exp(-X[:, 1]))
e = expit(logit(e) - adj)
tau = np.maximum(np.repeat(0.0, n), X[:, 0] + X[:, 1] + X[:, 2]) - np.maximum(
    np.repeat(0.0, n), X[:, 3] + X[:, 4]
)

w = np.random.binomial(1, e, size=n)
y = b + (w - 0.5) * tau + Y_noise * np.random.normal(size=n)
```



Python Code SLearner

```
def slearn(df_o, T, X, Y, model=LinearRegression()):

    df = df_o.copy()
    m = model.fit(df[X + [T]].to_numpy(), df[Y].to_numpy())

    df[T] = 0
    preds0 = m.predict(df[X + [T]].to_numpy())

    df[T] = 1
    preds1 = m.predict(df[X + [T]].to_numpy())

    return np.mean(preds1 - preds0)
```

Python Code TLearner

```
def tlearn(df, T, X, Y, model=LinearRegression()):

    df0 = df[df[T] == 0]
    m0 = model.fit(df0[X].to_numpy(), df0[Y].to_numpy())
    preds0 = m0.predict(df[X].to_numpy())

    df1 = df[df[T] == 1]
    m1 = model.fit(df1[X].to_numpy(), df1[Y].to_numpy())
    preds1 = m1.predict(df[X].to_numpy())

    return np.mean(preds1 - preds0)
```

Results

$$ATE = \mathbb{E}[Y_i(1) - Y_i(0)]$$

```
df = data.train_df
T = "T"
X = data.X_features
Y = "Y"

print(data.train_ite.mean())

effects = tlearn(df, T, X, Y, model=xgboost.XGBRegressor())
print(f"tlearn_xgb: {effects}")

effects = slearn(df, T, X, Y, model=xgboost.XGBRegressor())
print(f"slearn_xgb: {effects}")

print(df[df["T"] == 1]["Y"].mean() - df[df["T"] == 0["Y"].mean())
```

Real ATE	TLearner	SLearner	Naive diff
0.467	0.470	0.459	0.499

Improving data efficiency: XLearner

Intuition: The goal of XLearner is to estimate the treatment and no treatment separately, like the TLearner, but making a more efficient use of the data.

There are 3 steps to the XLearner:

The first step is the exact same as the TLearner, split the data into treatment and no treatment and fit a model for each group.

We will use M0 for the model of the group T=0 and M1 for T=1

Improving data efficiency: XLearner

Second step

The second step is where the X comes from. In this step we use the models from step 1 to predict the unobserved outcome for each data point (For those with T=1 we predict T=0 and vice versa).

Then we use these predicted unobserved outcomes to compute estimated ITEs ($\hat{\tau}$).

$$\hat{\tau}_1, i = Y_i(1) - M0(X_i)$$
$$\hat{\tau}_0, i = M1(X_i) - Y_i(0)$$

We then can fit two more models, one for $\hat{\tau}_1$ and one for $\hat{\tau}_0$

Improving data efficiency: XLearner

Third step

Finally we use the models from the second step to compute the two ITE predictions and combine them in a weighted manner (the weight as a hyper parameter)

More formal definition:

The X-Learner models $Y(1)$ and $Y(0)$ separately in order to estimate the CATT (Conditional Average Treatment Effect on the Treated) and CATC (Conditional Average Treatment Effect on the Controls). The CATE estimate for a new point x is given by the propensity-weighted average of CATT and CATC. A sketch of the X-Learner procedure is given below:

$$
\begin{aligned}
\hat{\mu}_0 &= M_1(Y^0 \sim X^0) \\
\hat{\mu}_1 &= M_2(Y^1 \sim X^1) \\
\hat{D}^1 &= Y^1 - \hat{\mu}_0(X^1) \\
\hat{D}^0 &= \hat{\mu}_1(X^0) - Y^0 \\
\hat{\tau}_0 &= M_3(\hat{D}^0 \sim X^0) \\
\hat{\tau}_1 &= M_4(\hat{D}^1 \sim X^1) \\
\hat{\tau} &= g(x)\hat{\tau}_0(x) + (1 - g(x))\hat{\tau}_1(x)
\end{aligned}
$$

where $g(x)$ is an estimation of $P[T = 1|X]$ and M_1, M_2, M_3, M_4 are suitable machine learning algorithms.

https://econml.azurewebsites.net/

Propensity score

$$propensity\ score = e(X) = P(T = 1 \mid X)$$

Intuition:

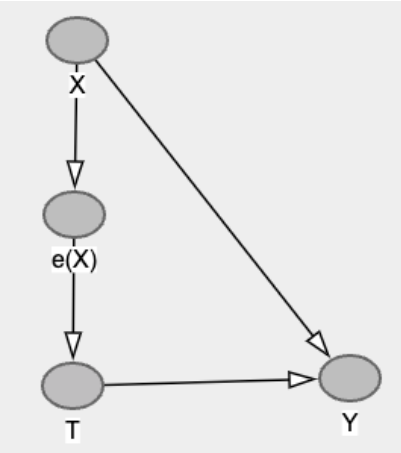
→ The propensity score is the probability of being assigned the treatment T given the covariates X.

Propensity score

$$propensity\ score = e(X) = P(T = 1 \mid X)$$

Intuition:

- The propensity score is the probability of being assigned the treatment T given the covariates X.
- We can think of the propensity as a mediator of the effect of X on T, so we can see e(X) blocks all backdoor paths from T to Y.



Propensity score theorem

$$(Y(1), Y(0)) \perp T \,|\, X \implies (Y(1), Y(0)) \perp T \,|\, e(X)$$

Propensity score theorem: Unconfoundedness given X implies unconfoundedness given e(X).

→ This theorem means that we can use e(X) in place of X, which is of relevant importance when X is high-dimensional

Propensity score theorem

$$(Y(1), Y(0)) \perp T \,|\, X \implies (Y(1), Y(0)) \perp T \,|\, e(X)$$

Propensity score theorem: Unconfoundedness given X implies unconfoundedness given e(X).

→ This theorem means that we can use e(X) in place of X, which is of relevant importance when X is high-dimensional

BUT… We usually don't have access to e(X), so we have to model it. We do this by training a model to predict T from X.

Propensity score methods

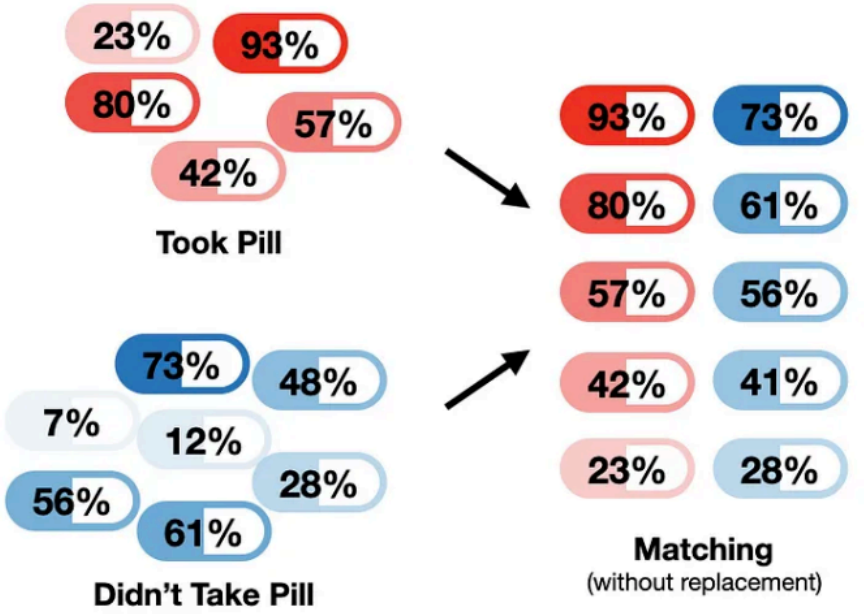
Apart from being sufficient for obtaining unconfoundedness, propensity score can be used to compute causal effects using different strategies.

- Matching
- Stratification
- IPW

Propensity - matching

- Matching consist of creating **treated-untreated pairs** with similar propensity scores

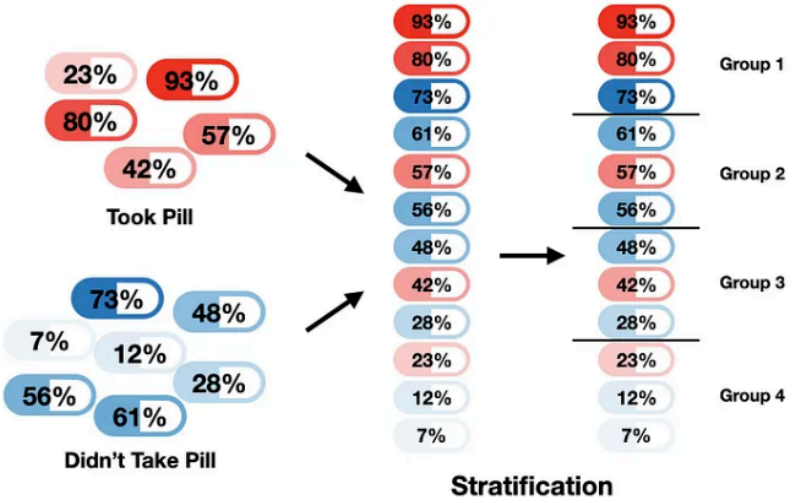
- Using these pairs, treatment effects can be computed directly by comparing the treated and untreated populations



https://towardsdatascience.com/propensity-score-5c29c480130c

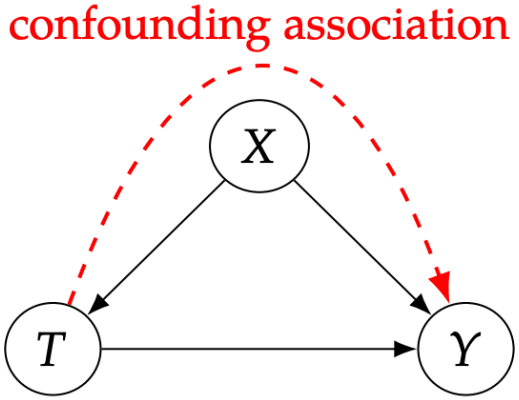
Propensity - stratification

- Similar to matching
- First step is to **order the observations** according to the propensity score
- Then, divide into groups and treat them as comparable and compute treatment effects
- Finally, group effects can be combined to obtain ATE



https://towardsdatascience.com/propensity-score-5c29c480130c

Propensity - Inverse Probability Weighting

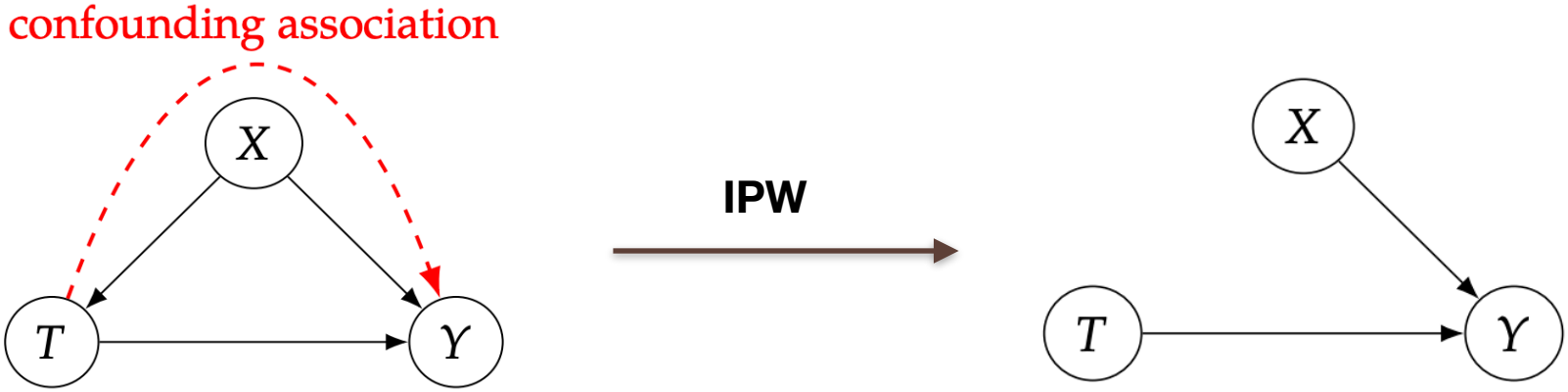


→ In the case, association is not causation because X is a common cause of both T and Y

→ In other words: $P(T \mid X) \neq P(T)$

Intuition: What if we could sample the data in way to make the edge from X -> T disappear?

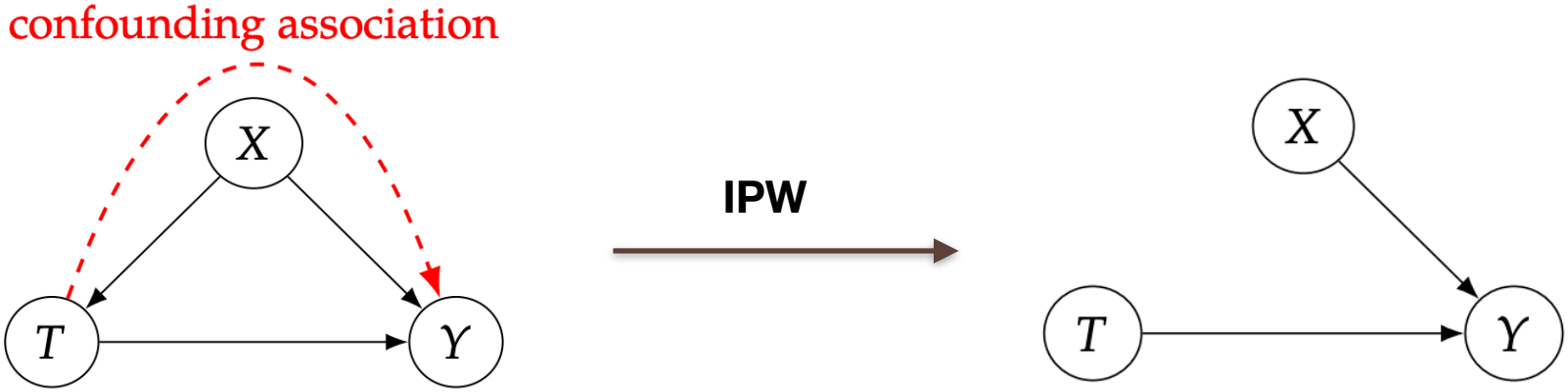
Propensity - Inverse Probability Weighting



IPW consist on removing this edge by reweighing each data point by its inverse probability of receiving its treatment value: **1/P(T|X)**

⚠ In the binary case, this corresponds to 1/e(X) for T = 1 and 1/(1 - e(X)) for T= 0

Propensity - Inverse Probability Weighting



Note that during IPW, the propensity score is used in the denominator, leading to potential numerical issues when propensity is close to 0.

Double Robust learning

- We have seen that we can estimate causal effects my modelling:
  - $\mathbb{E}[Y \mid T, X]$
  - $e(X)$
- It is possible to estimate causal effects using **both modelling approaches**, and estimators that do this are called double robust
- Double robust estimators have some theoretical properties, such as being consistent* if either of the models is well-specified

*An estimator is consistent if it converges in probability to its estimand as the number of samples grows.

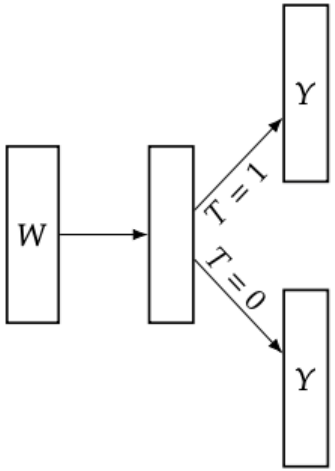
Deep Learning specific methods: TARNet

→ TARNet is a **deep learning approach** that lies in the middle ground between SLearner (1 single model and T as part of the covariates) and TLearner (2 models, one for T=1 and one for T=0)

Deep Learning specific methods: TARNet

- TARNet is a **deep learning approach** that lies in the middle ground between SLearner (1 single model and T as part of the covariates) and TLearner (2 models, one for T=1 and one for T=0)

- TARNet consists of a **main network that branches** into two sub-networks

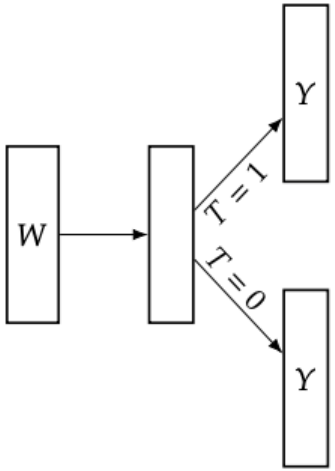


https://www.bradyneal.com/causal-inference-course

Deep Learning specific methods: TARNet

- TARNet is a **deep learning approach** that lies in the middle ground between SLearner (1 single model and T as part of the covariates) and TLearner (2 models, one for T=1 and one for T=0)

- TARNet consists of a **main network that branches** into two sub-networks
- This model **makes use of all the datapoints** and is forced to take into account T
- Each subnetwork is still only trained with treatment group data



https://www.bradyneal.com/causal-inference-course

Existence of other estimand based methods

- There exists other estimand based methods when the backdoor criterion doesn't hold:
  - Frontdoor adjustment methods
  - Instrumental Variables (IV)
- These cases are not as common as the backdoor cases and usually require a more customised approach

Out of scope for this course

Real Case example by Netflix

Netflix Case

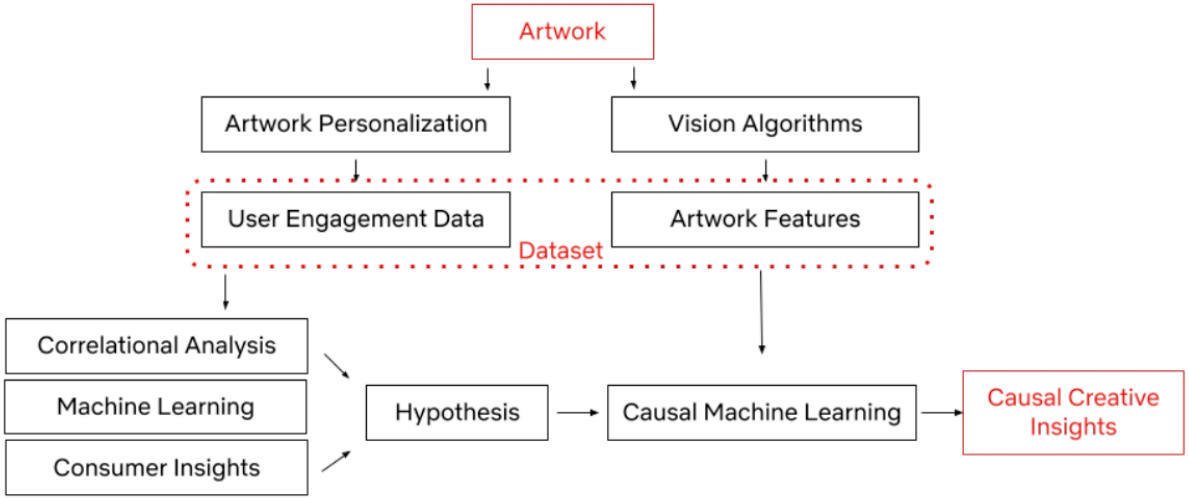
https://netflixtechblog.medium.com/causal-machine-learning-for-creative-insights-4b0ce22a8a96

The Challenge

Given Netflix's vast and increasingly diverse catalog, it is a challenge to design experiments that both work within an A/B test framework and are representative of all genres, plots, artists, and more. In the past, we have

Netflix Case

https://netflixtechblog.medium.com/causal-machine-learning-for-creative-insights-4b0ce22a8a96



Netflix Case

https://netflixtechblog.medium.com/causal-machine-learning-for-creative-insights-4b0ce22a8a96

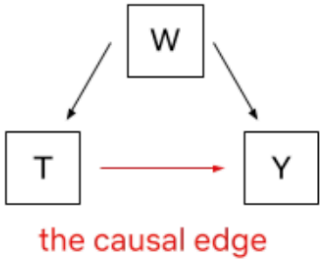
The Hypothesis and Assumptions

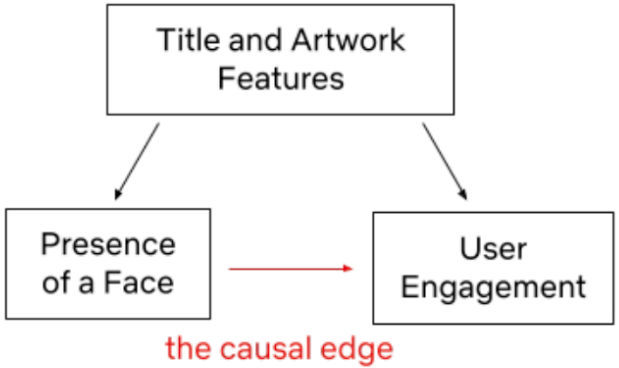
We will use the following hypothesis in the rest of the script: *presence of a face in an artwork causally improves the asset performance*. (We know that faces work well in artwork, especially images with an expressive facial emotion that's in line with the tone of the title.)

To make sure our hypothesis is fit for the causal framework, it's important we go over the *identification assumptions*.

Netflix Case

https://netflixtechblog.medium.com/causal-machine-learning-for-creative-insights-4b0ce22a8a96





Y : outcome variable (take rate)

T : binary treatment variable (presence of a face or not)

W: a vector of covariates (features of the title and artwork)

Netflix Case

https://netflixtechblog.medium.com/causal-machine-learning-for-creative-insights-4b0ce22a8a96

> For the estimation of the nuisance functions (i.e., the propensity score model and the outcome model), we have implemented the propensity model as a classifier (as we have a binary treatment variable — the presence of face) and the potential outcome model as a regressor (as we have a continuous outcome variable — adjusted take rate). We have used grid search for tuning