Yale

Explainability of Neural Networks (XAI)

CPSC680: Trustworthy Deep Learning

Rex Ying

Readings

- Readings are updated on the website (syllabus page)
- **Lecture 5 readings**:
  - LIME (local interpretation)
  - SHAP (attribution)

Content

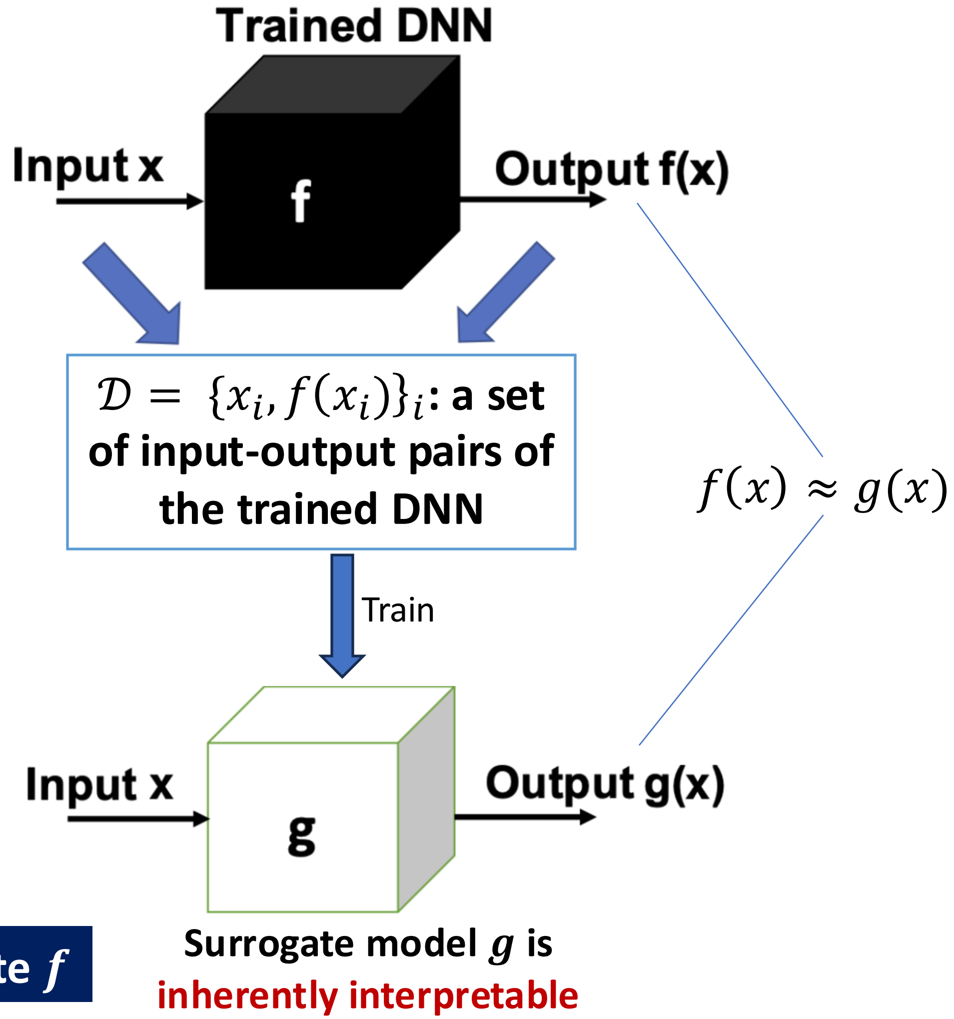
- Methods using Surrogate Models
- Counterfactual Explanations

Content

- Methods using Surrogate Models
- Counterfactual Explanations

Local Explanations with Surrogate Models

- **Explanation with Surrogate Model**
  - **Post-hoc, model-agnostic** explanation
  - Learn **an inherently interpretable model** (e.g. decision trees, linear models) that **(locally) approximates** the behaviors of the original model.
  - We can analyze the **local** behaviors of f around x_0 using g.

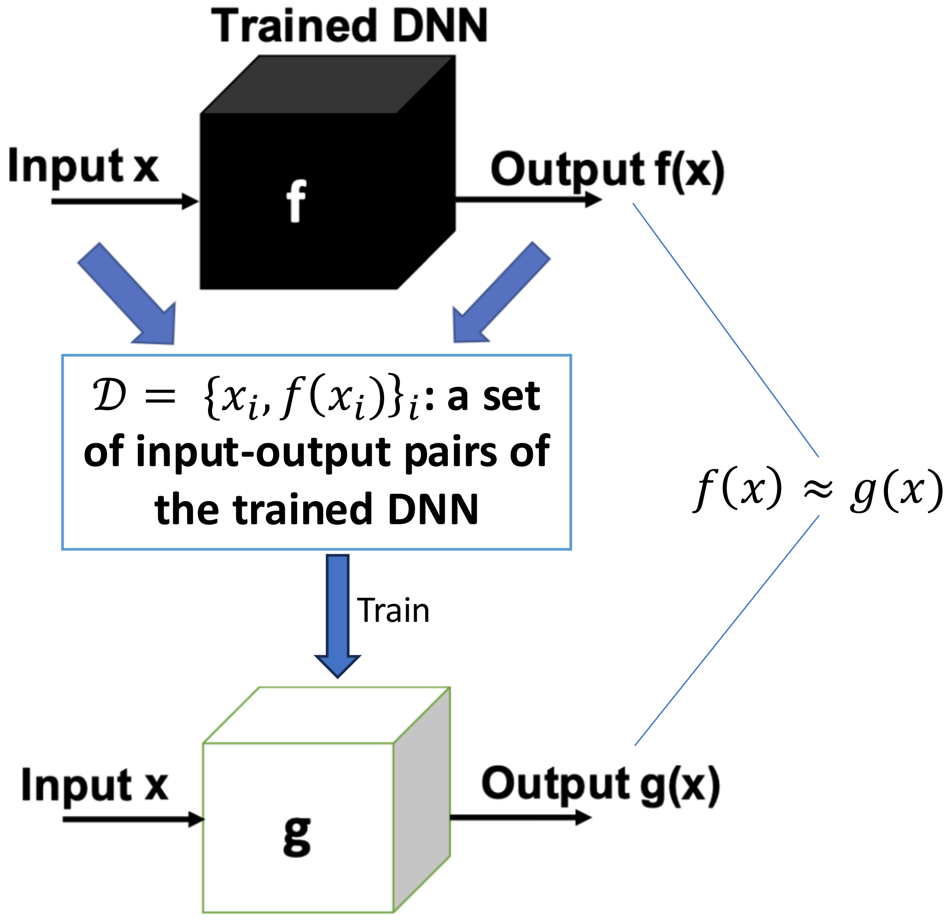


Use $\mathcal{D} = \{x_i, f(x_i)\}_i$ as traing data for g to approximate f

Local Explanations with Surrogate Models

- **Explanation with Surrogate Model**
- Given input instance x_0 and model f.
- Steps:
    1. Sample points around x_0
    2. Use model to predict labels for each example $(x_i, f(x_i))$
    3. Weight examples according to the distance to the input instance x_0
    4. Learn a linear model on weighted samples
    5. Use simple linear model to explain

Use $\mathcal{D} = \{x_i, f(x_i)\}_i$ as traing data for g to approximate f

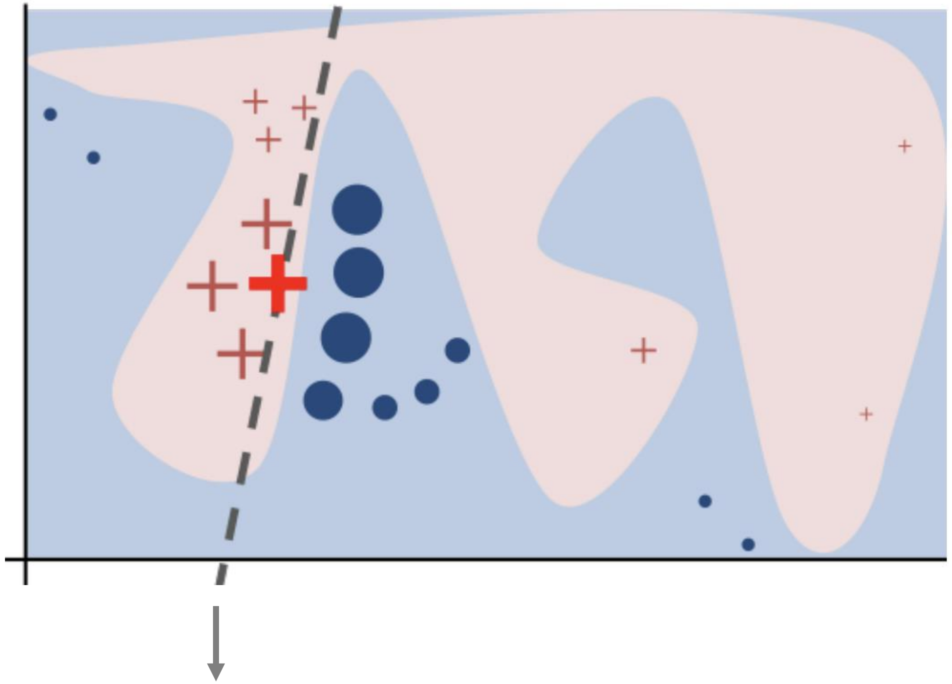


Surrogate model g is inherently interpretable

Intuition of LIME

- **Local Interpretable Model-Agnostic Explanations (LIME)**

decision boundary of a complex model



Locally linear decision boundary learned by LIME

- the pink and blue background: complex decision function f of the black-box model
- ➕ : instance to be explained
- ● ＋: instances with different predicted labels by f
- **Size** of ● ＋: the **proximity** to the instance to be explained
  (larger size ⟺ closer to the explained instance)

LIME samples instances around the given instance ➕, and weighs them by the **proximity** to the instance ➕.
Goal of LIME: learn a local surrogate model around the given instance ➕

LIME for Tabular Data

- **LIME**
- Given input instance x_0 and model f.
- Steps:
  1. **Sample points around x_0**
  2. Use model to predict labels for each example $\left(x_i, f(x_i)\right)$
  3. Weight examples according to the distance to the input instance x_0
  4. Learn a linear model on weighted samples
  5. Use simple linear model to explain

Sampling Mechanism

- **For continuous features**: Adding a Gaussian noise $x_i \sim \mathcal{N}(x_0, \sigma I)$
- **For categorical features**: perturb by sampling according to the training distribution

LIME for Tabular Data

- **LIME**
- Given input instance x_0 and model f.
- Steps:
  1. Sample points around x_0
  2. Use model to predict labels for each example $(x_i, f(x_i))$
  3. **Weight examples according to the distance to the input instance x_0**
  4. Learn a linear model on weighted samples
  5. Use simple linear model to explain

Weighting Function

$\boldsymbol{\pi_{x_0}(z)}$**: similarity kernel** for the recovered representation z around the original input x_0

$$\pi_{x_0}(x_i) = e^{-\frac{\boldsymbol{D}(x_0, x_i)^2}{\sigma^2}}$$

- $\boldsymbol{D}$ **is the distance function** (e.g., L_2 distance)
- σ is a hyper-parameter for the kernel

LIME targets at **local approximation** $\Rightarrow$ samples z with relatively **larger** $\boldsymbol{\pi_{x_0}(z)}$ (i.e., smaller distance $\boldsymbol{D}(x_0, x_i)$) should have larger weights during the training of the surrogate model

LIME for Tabular Data

- **LIME**
- Given input instance x_0 and model f.
- Steps:
  1. Sample points around x_0
  2. Use model to predict labels for each example $\left(x_i, f(x_i)\right)$
  3. Weight examples according to the distance to the input instance x_0
  4. **Learn a linear model g on weighted samples**
  5. Use simple linear model to explain

Training Objective

The training objective of LIME to search a surrogate model around input $\boldsymbol{x}$:

$$g^* = \underset{g \in \mathcal{G}}{\operatorname{argmin}} \mathcal{L}\left(f, g, \pi_{x_0}\right) + \Omega(g)$$

- $\mathcal{G}$ is the class of interpretable models (linear model or decision tree)
- $\mathcal{L}\left(f, g, \pi_{x_0}\right)$ is the loss function that penalizes the differences between f and g
- $\Omega(g)$ controls the complexity of the model g

LIME for Tabular Data: Objective and Loss

Training objective of LIME to search a surrogate model around input $\boldsymbol{x_0}$:

$$g^* = \underset{g}{\operatorname{argmin}}\, \mathcal{L}\left(f, g, \pi_{x_0}\right) + \boxed{\Omega(g)}$$

Control complexity of the interpretable model

where

$$\mathcal{L}(f, g, \pi_{x_0}) = \sum_{x_i \in \mathcal{D}} \boxed{\pi_{x_0}(\mathrm{x_i})}\boxed{\left(f(\mathrm{x_i}) - g(x_i)\right)}^2$$

$$\boxed{\pi_{x_0}(x_i) = e^{-\frac{D(x_0, x_i)^2}{\sigma^2}}}$$

Sample x_i closer to x_0 contributes more to the loss

optimize g to locally approximate the behavior of f

- $\Omega(g)$: a measure of **complexity** of the surrogate model g
  - $\Omega(g)$: the depth of tree for a decision tree; the number of weights for a linear model
  - **A simple local surrogate is prefered! ([Occam'z Razor](#))**
  - **Lasso is a possible choice**

LIME for Images: Data Representation (1)

- Given an original input $x \in \mathbb{R}^d$, let $x' \in \{0,1\}^M$ denote a binary vector as its **interpretable representation**
  - $\boldsymbol{M}$ is the number of features in the interpretable representation
  - For **images**, x' can be a binary vector indicating the “presence” or “absence” of **a patch of similar pixels**
  - For **text**, x' can be a binary vector indicating the “presence” or “absence” of **a word**
  - **Example:**

x: Original image

1 denotes “presence” of the pixel
0 denotes “absence” of the pixel

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0
0	0	1	1	1	1	0	0	0
0	0	1	1	1	1	0	0	0
0	0	1	1	1	1	0	0	0
0	0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0

x': Interpretable representation

Choice of Interpretable Representation

- Given an original input $x \in \mathbb{R}^d$, let $x' \in \{0, 1\}^M$ denote a binary vector as its **interpretable representation**
  - In the context of images, a segmentation method can be used to decompose the image into interpretable components (see skimage.segmentation)

Original Image

Interpretable Components

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0
0	0	1	1	1	1	0	0	0
0	0	1	1	1	1	0	0	0
0	0	1	1	1	1	0	0	0
0	0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0

Mask corresponding to a super-pixel (segment)

Details of LIME: Data Representation (2)

- Based on an **interpretable representation** $x' \in \{0,1\}^M$, a **perturbed sample** $z' \in \{0,1\}^M$ contains **a fraction of the nonzero elements of** x'
- $z \in \mathbb{R}^d$: **recovered explanation** in the original domain
- Example:

$x \in \mathbb{R}^d$
Original image

0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0
0	0	0	1	1	1	1	0	0
0	0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0	0
0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

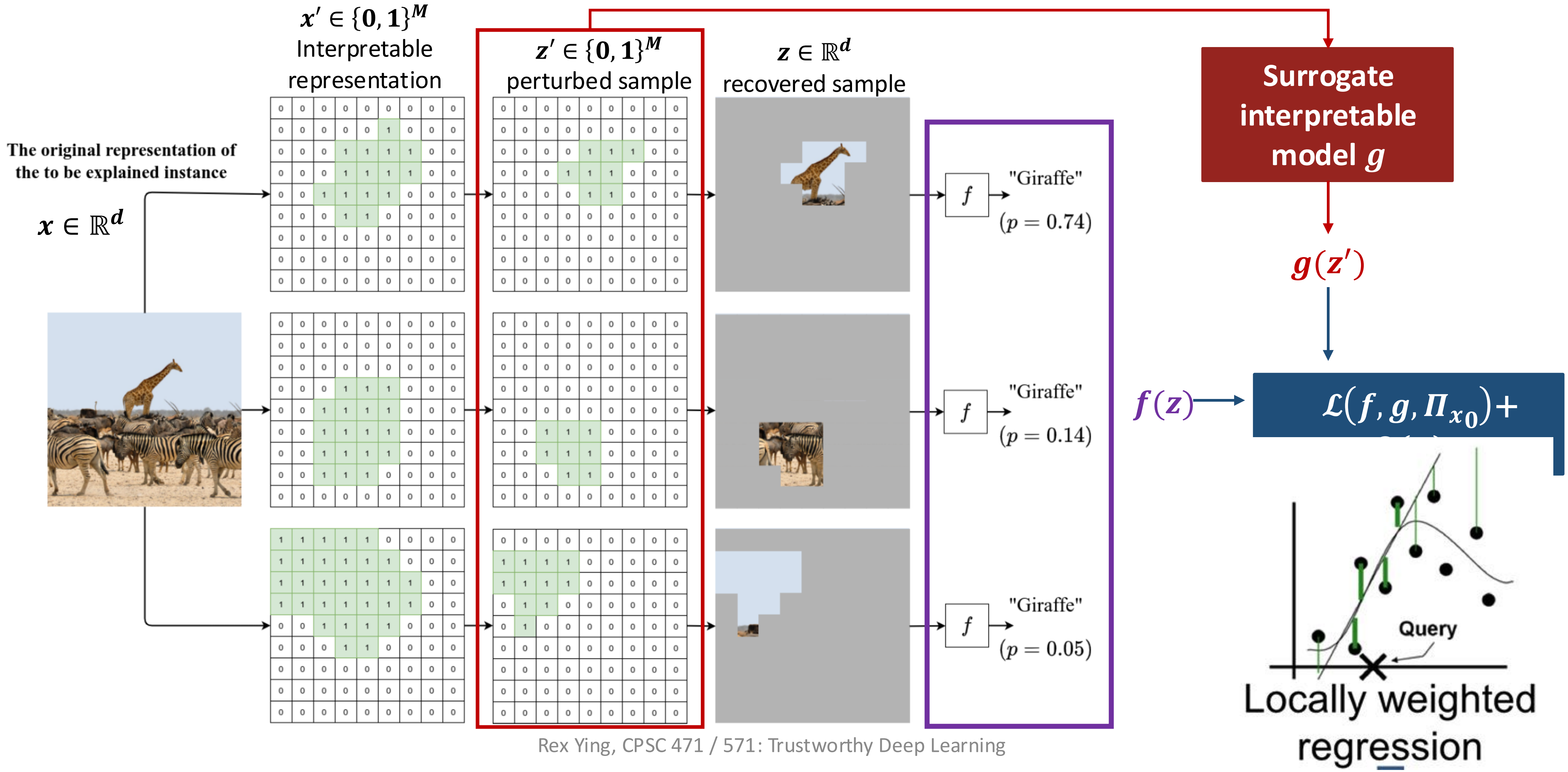
$x' \in \{0,1\}^M$
Interpretable representation

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	0	0
0	0	0	1	1	1	0	0	0
0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

$z' \in \{0,1\}^M$
perturbed sample

$z \in \mathbb{R}^d$
explanation

LIME for Images - Architecture



Evaluation: Important Feature Selection (1)

- **Dataset:** sentiment analysis datasets (BOOKs and DVDs)[1]
- **Features: bag of words (BOW)**

	she	loves	pizza	is	delicious	a	good	person	people	are	the	best
Input sentence: She loves pizza, pizza is delicious	1	1	2	1	1	0	0	0	0	0	0	0

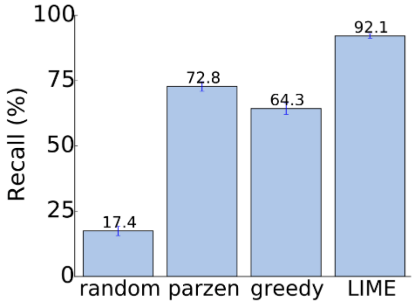
BOW: number of times each word occurs in a sentence

- Measure **faithfulness of explanations** for classifiers that are intrinsically interpretable
  - Train a **sparse logistic regression** or **decision tree** to select 10 most important features as the ground truth
- Generate explanations for each prediction in the test set and compute **the fraction of truly important features** recovered by the explanations

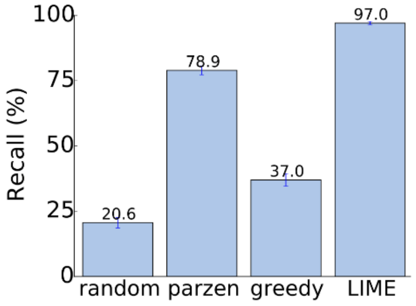
[1]J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification.

Evaluation: Important Feature Selection (2)

- **Baselines:**
  - **Random:** randomly pick 10 features
  - **Parzen**: pick 10 features with the highest absolute gradients
  - **Greedy:** greedily remove 10 features that contribute the most to the predicted class and take these 10 features as an explanation
- **Recall**: $\frac{TP}{TP+FN}$ (**true positive rate**; TP: True Positve, FN: False Negative)

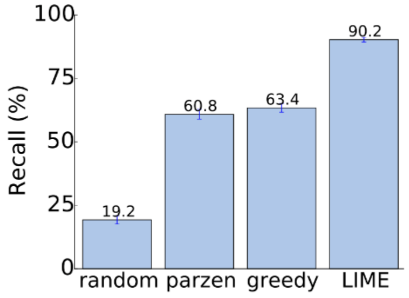


(a) Sparse LR

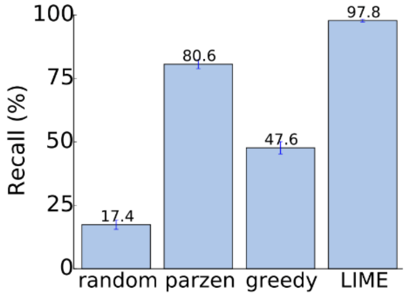


(b) Decision Tree

BOOKs dataset



(a) Sparse LR



(b) Decision Tree

DVDs dataset

LIME provides faithful explanations with > 90% recall for both logistic regression and decision trees!

Shapley Additive Explanation

- **SHAP**: a **local additive feature attribution** methods based on **Shapley values** for each input feature
- Given a black-box model f and an input $x \in \mathbb{R}^d$ to be explained
  - $z \in \{0,1\}^M$: interpretable representation of x

    recall the representation used in LIME
  - h_x: $\{0,1\}^M \to \mathbb{R}^d$ recovering function that maps to original domain $\mathbb{R}^d$
- **Surrogate model:**

null output

Shapley value of the i-th feature

$$g(z) = \boxed{\phi_0} + \sum_{i=1}^{M} \boxed{\phi_i} z_i$$

  - $z_i \in \{0,1\}$: the i-th element of z_i, indicating the **presence/absence of the i-th feature**
  - $g(z)$ locally approximates $f(x)$ when $\boxed{x = h_x(z)}$

map to the original domain

Shapley Values in Cooperative Game Theory

- The concept of Shapley values is originally from **cooperative game theory**
- **Cooperative games** model scenarios where agents can benefit by cooperating together and a binding agreement.
  - Probably not the part of game theory you've heard of

Cooperative game

- Players can benefit by cooperating
- Binding agreements are possible
- Answer the question: How to divide the surplus when joining the grand coalition?

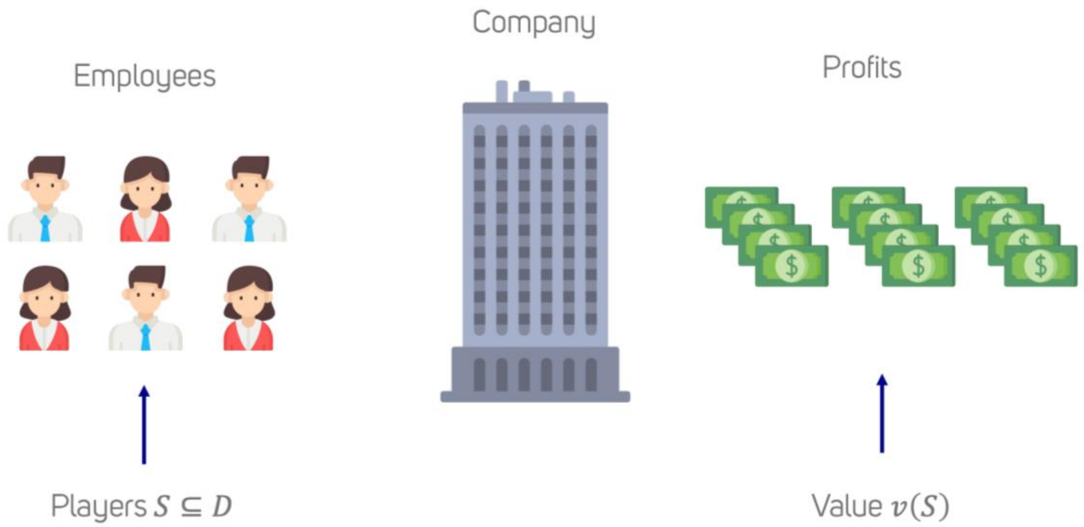
Non-cooperative game

- Players are independent
- No cooperation, focus on individual actions
- Answer the question: What is the "good" strategy for each player to maximize their ***individual*** return?

Nash equilibrium, zero-sum game

Shapley Values in Cooperative Game Theory

- **Example of a cooperative game:**

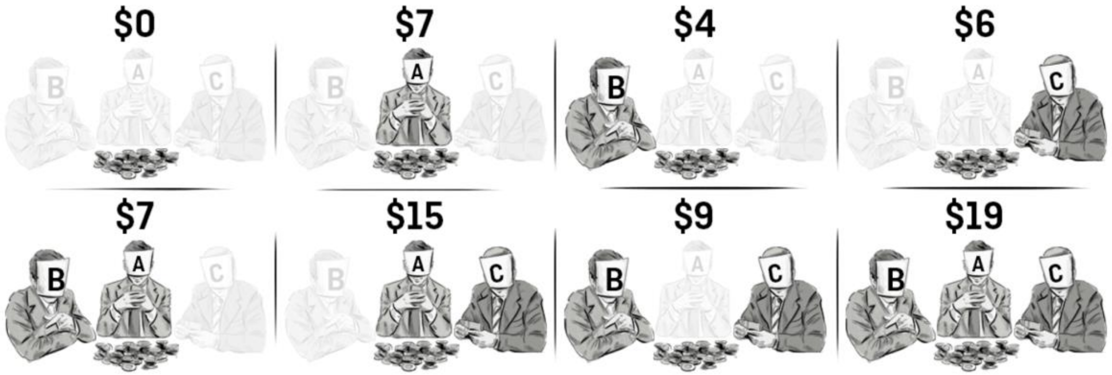


- **Question:** How to measure each player's contribution? How to divide a surplus (profit) to shareholders so that everyone is satisfied?
- **Lloyd Shapley's idea**: members should receive payments **proportional to their average marginal contributions $\Longrightarrow$ Shapley Values**

Shapley Values in Cooperative Game Theory

- Consider the following example where there are three players and got 19$
- The marginal contribution of player A
  - to coalition $S = \emptyset$ is 7 – 0 = 7$ / (3 * 1)
  - to coalition $S = \{B\}$ is 7 – 4 = 3$ / (3 * 2)
  - to coalition $S = \{C\}$ is 15 – 6 = 9$ / (3 * 2)
  - to coalition $S = \{B, C\}$ is 19 – 9 = 10$ / (3 * 1)
- The player A should get

$$\frac{1}{3}\left(\frac{1}{\binom{2}{0}}7 + \frac{1}{\binom{2}{1}}3 + \frac{1}{\binom{2}{1}}9 + \frac{1}{\binom{2}{2}}10\right) = 7.667$$



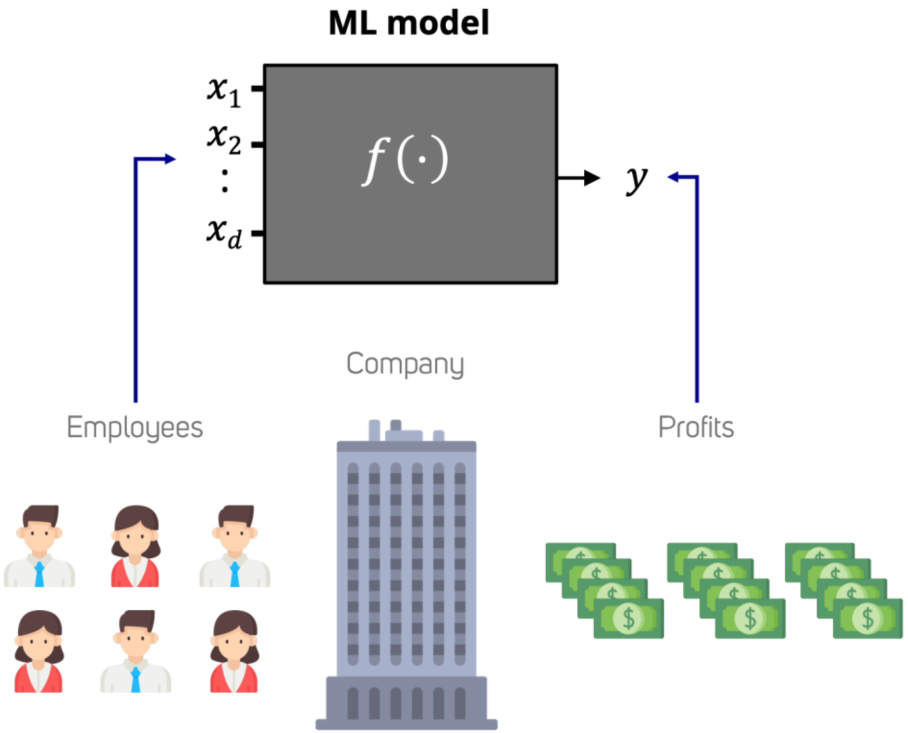
Marginal contribution

Every coalition with the same size is equally likely to appear

All sizes are equally likely to appear.

Shapley Values in Cooperative Game Theory

- **How do we use Shapley Value in the context of a deep learning system?**



features⇔ players   prediction⇔ profit

Calculating Shapley Values

- f: black-box model; $x \in \mathbb{R}^d$: input to be explained; $\boldsymbol{h_x}$**: reverse map to** $\mathbb{R}^d$
- **Shapley value** for the i-th input feature:

Setting $z'_i = 0$ (removing the i-th feature)

$$\phi_i(f,x) = \sum_{z' \subseteq Z} \boxed{\frac{|z'|!\,(M - |z'| - 1)!}{M!}} [f(\boldsymbol{h_x}(z')) - f(\boldsymbol{h_x}(z' \setminus i))]$$

All possible binary representations of subset features of x, $|Z| = 2^M$

Very small $|z'|$ (close to 0) or very large $|z'|$ (close to M) result in larger weights. Why?

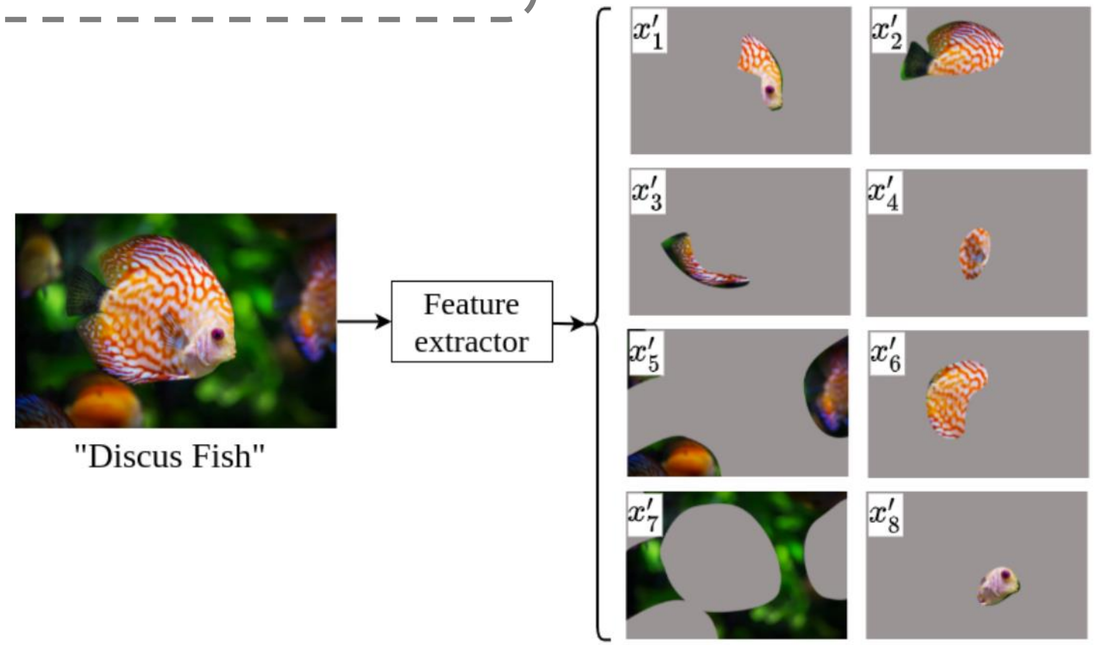
  - $z' \subseteq x'$ represents all z' vectors where the non-zero entries are a **subset** of the non-zero entries in x'
  - $|z'|$: the number of none-zero entries in z'
  - M: number of features in the interpretable representation

Example of SHAP

Example: calculating the **Shapley value of feature 4:**

x'	x_1	x_2	x_3	x_4	......
$z' \subseteq x'$	x_1	x_2	x_3	x_4	......
$z' \backslash 4$	x_1	x_2	x_3	x_4	......

x': interpretable representation (8 features in total)

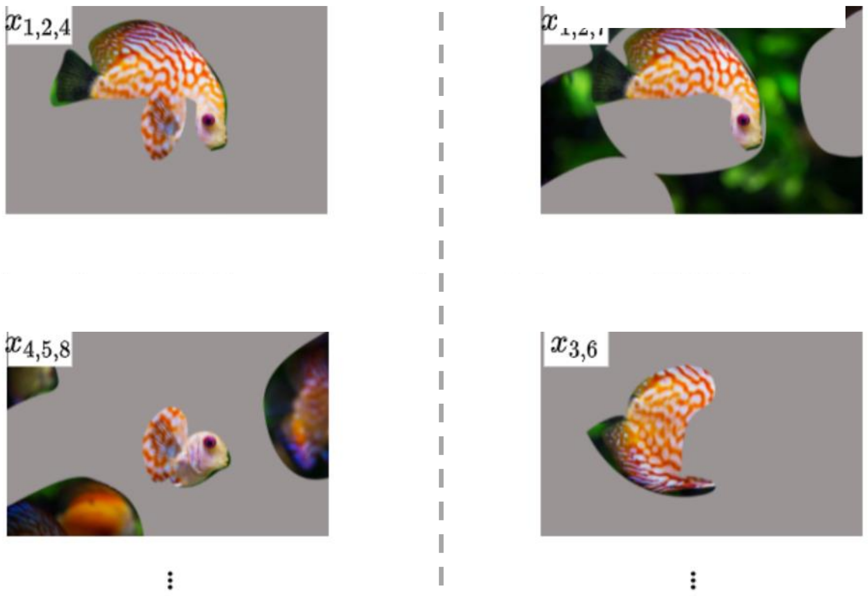


"Discus Fish"

Compute $f\left(h_x(z')\right) - f\left(h_x(z' \backslash 4)\right)$, and weighted average over all possible subset z'

$f\left(h_x(z')\right)$ $\quad$ $f\left(h_x(z' \backslash 4)\right)$

$z' \subseteq x'$ $\quad$ $z' \subseteq x'$



Properties of SHAP (1)

- **Linearity:** The importance score of a linear combination of two models is a linear combination of the importance score

$$\phi_i(f_1 + \alpha f_2) = \phi_i(f_1) + \alpha\phi_i(f_2)$$

- **Dummy:** If the model is not sensitive to a feature, its importance score should be zero

$$\forall z \quad \delta_i(z) := f(h_x(z)|z_i = 1) - f(h_x(z)|z_i = 0) = 0 \Rightarrow \phi_i = 0$$

- **Symmetry:** Symmetric features get similar importance scores

$$\forall z \quad \delta_i(z) = \delta_j(z) \Rightarrow \phi_i = \phi_j$$

- **Efficiency**: The sum of importance scores of all features recovers the prediction

$$f(x) = g(\mathrm{z}) = \phi_0 + \sum_{i=1}^{M} \phi_i z_i$$

The model output is fully distributed to all input features.

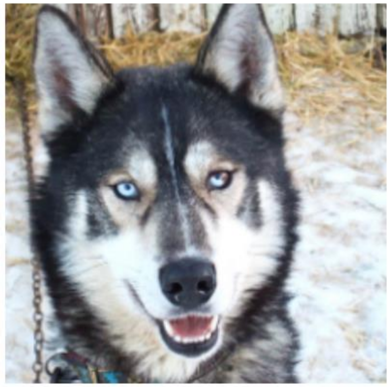
Properties of SHAP (2)

- **Theorem:** Shapley value is a ***unique*** solution concept that satisfies four axioms: *linearity*, *dummy*, *symmetry*, and *efficiency*.

- **Pros: This uniqueness** implies that the Shapley values is the "best" (only one) method to allocate importance scores to input features if we accept four properties (axioms)
- **Cons: Computationally expensive**.
  - E.g., to compute exactly the Shapley value with 50 features, we need to compute the summation over $2^{50} > 10^{15}$ perturbation $z' \in Z$.
  - We need to do Monte-Carlo sampling to approximate the Shapley values. In practice, 1000 – 10000 perturbations are sufficient.

What are Pros and Cons of SHAP?

Questions!



(a) Husky classified as wolf

(b) Explanation

- **Efficiency**
- **Stability**
- Modeling **Correlation**
- **Robustness**
  - How likely will the identified explanations tend to be adversarial examples
- **Granularity**
  - Can the method find fine-grained explanations for any instance

The equivalence between LIME and SHAP

- **Theorem:** The Shapley values is a **special case** of the LIME framework

LIME	SHAP
• LIME solves the following optimization problem $$w^\star = \underset{w}{\operatorname{argmin}} \sum_{z' \in \mathfrak{D}} \pi_x(z')[f(z') - (w^\top z' + b)]^2 + \Omega(w)$$ where $\mathcal{D}$ is the set of perturbations around x	• Importance score using Shapley value $$\phi_i = \sum_{z' \subseteq x'} \frac{\lvert z'\rvert!\,(M - \lvert z'\rvert - 1)!}{M!}[f(\boldsymbol{h}_{\boldsymbol{x}}(z')) - f(\boldsymbol{h}_{\boldsymbol{x}}(z' \setminus i))]$$ x' is the interpretable representation of x

If
- $\Omega(w) = 0$ — No control over the complexity of g
- $\pi_x(z') = \frac{(M-1)}{\binom{M}{\lvert z'\rvert}\lvert z'\rvert(M-\lvert z'\rvert)}$ — Weighting function: Small $\lvert z'\rvert$ (few 1's in z') and large $\lvert z'\rvert$ (i.e. many 1's in z') get the largest weights
- $\mathfrak{D} = \{h_{x_0}(z') \mid z' \subseteq x'\}$ — The perturbation set is the set of all possible binary combinations of features in x, $\lvert\mathcal{D}\rvert = 2^M$

Then $w^\star = \phi_i$

See proof in the original paper

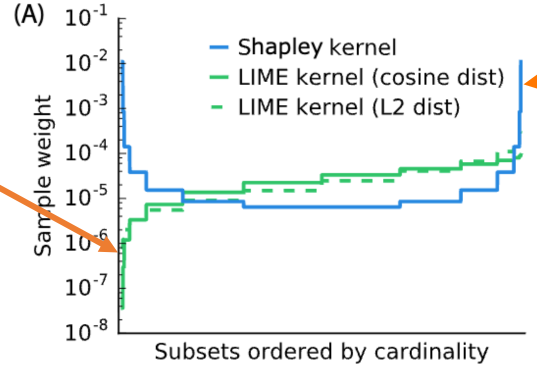
The equivalence between LIME and SHAP

- **Theorem:** The Shapley values is a **special case** of the LIME framework

LIME	SHAP
• LIME solves the following optimization problem	• Importance score using Shapley value
$w^\star = \underset{w}{\operatorname{argmin}} \sum_{z' \in \mathfrak{D}} \pi_x(z')[f(z') - (w^\top z' + b)]^2 + \Omega(w)$	$\phi_i = \sum_{z' \subseteq x'} \frac{\lvert z' \rvert!\,(M - \lvert z' \rvert - 1)!}{M!}[f(\boldsymbol{h}_{\boldsymbol{x}}(z')) - f(\boldsymbol{h}_{\boldsymbol{x}}(z' \setminus i))]$
where $\mathcal{D}$ is the set of perturbations around x	x' is the interpretable representation of x

$$\pi_x(z) = \exp \frac{-d(x,z)^2}{\sigma^2}$$

$d(x,z)$ can be L2 or cosine distance between the original image and the perturbed image



Cardinality is $\lvert z' \rvert$ (the number of 1s in z').

$$\pi_x(z') = \frac{(M-1)}{\binom{M}{\lvert z' \rvert} \lvert z' \rvert (M - \lvert z' \rvert)}$$

SHAP gives higher weights for subsets with smaller cardinality

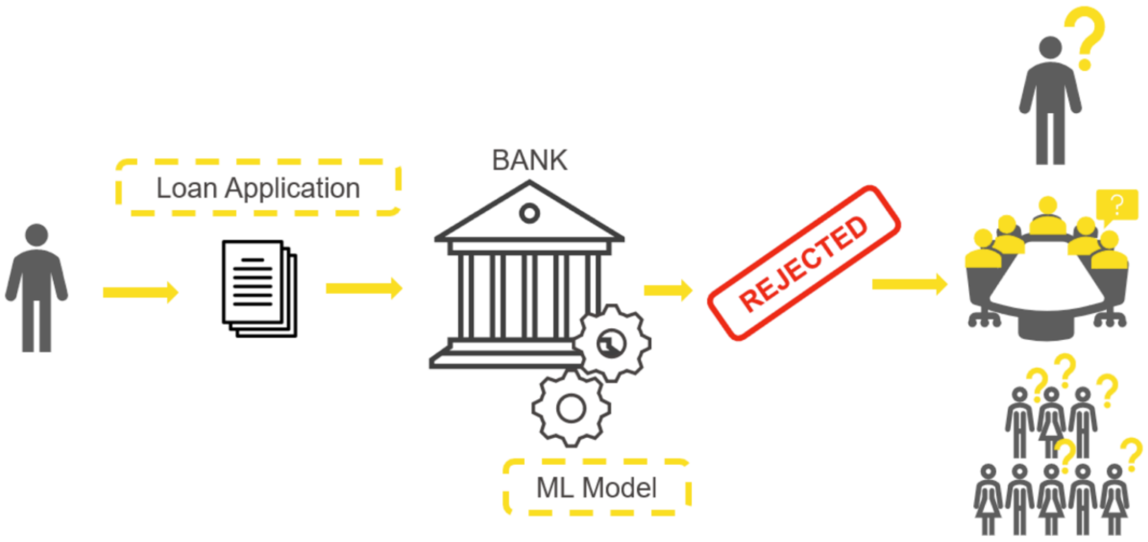
See proof in the original paper

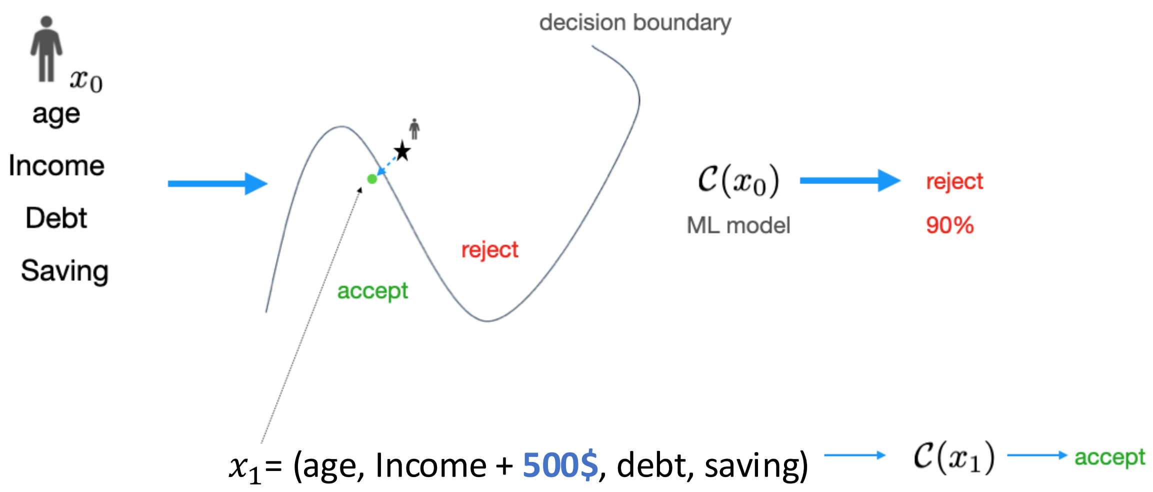
Content

- Methods using Surrogate Models
- Counterfactual Explanation

Counterfactual Explanations

- Counterfactual explanation considers **"what-if" scenarios** of model predictions, addressing the question of **how slight adjustments** in the input can lead to different model predictions.
  - useful in consequential applications such as loan approvals, university admission, etc.





You should increase your income by 500$ in order to be accepted

Counterfactual Explanations

- **Goal**: Find a counterfactual example
  - can change the model prediction to a desired outcome
  - a change in the input instance should be minimal in order to reduce the implementation cost for users (loan applicants, students, etc)

distance function (implementation cost)

- **Formulation**:

$$\boxed{x_{CF}} = \underset{\boxed{x' \in \mathcal{X}}}{\operatorname{argmin}} \boxed{d(x_0, x')}$$

$$s.t. \quad \boxed{f(x') = y^{\text{target}}}$$

Counterfactual Explanation

Feasible action space

Constraint to get desired outcome

Counterfactual Explanations

- **Formulation**:

$$x_{CF} = \underset{x' \in \mathcal{X}}{\operatorname{argmin}}\, d(x_0, x')$$
$$s.t. \quad f(x') = y^{\text{target}}$$

The above problem has a non-linear constraint, which is difficult to solve

- **Reformulation**: Using Largange multipliers to convert to unconstrained optimization problem:

$$x_{CF} = \underset{x' \in \mathcal{X}}{\operatorname{argmin}}\, d(x_0, x') + \boxed{\lambda}\mathcal{L}(f(x'), y^{\text{target}})$$

Lagrange multiplier

where

$$\boxed{\mathcal{L}(f(x'), y^{\text{target}}) = (f(\mathrm{x}') - y^{\text{target}})^2}$$

Least Square Error

This problem is differentiable and can be solved by Projected Gradient Descent Algorithm

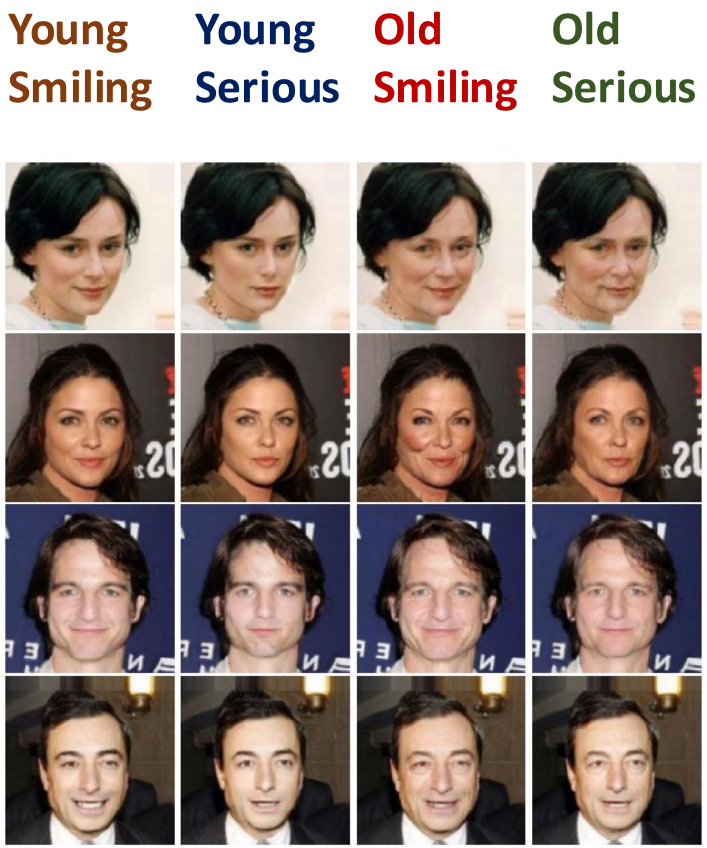
Generative Approaches

- $x_{CF} = \underset{x' \in \mathcal{X}}{\operatorname{argmin}}\, d(x_0, x')$

  $s.t. \quad f(x') = y^{\text{target}}$

- x' can be obtained through **generative models**
  - Variational autoencoder
  - Diffusion model

- **Objective 1**: reconstruction (also distribution loss)
- **Objective 2**: target constraint (y^{target})



Summary

- **Surrogate models** learns a local approximation of the decision boundary at a given instance
  - The local approximation is done by a simple explainable model such as linear regression or decision tree
  - Explanation is at the granularity of "interpretable representation"
  - A kernel function is used
- **Shapley value** has a game theoretic interpretation of contribution in a coalition game
  - Can be used to weight the contribution of each perturbation
- **Counterfactual explanation** deals with the "what-if" question
  - The model finds a perturbation that causes the model to switch its prediction