

# 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

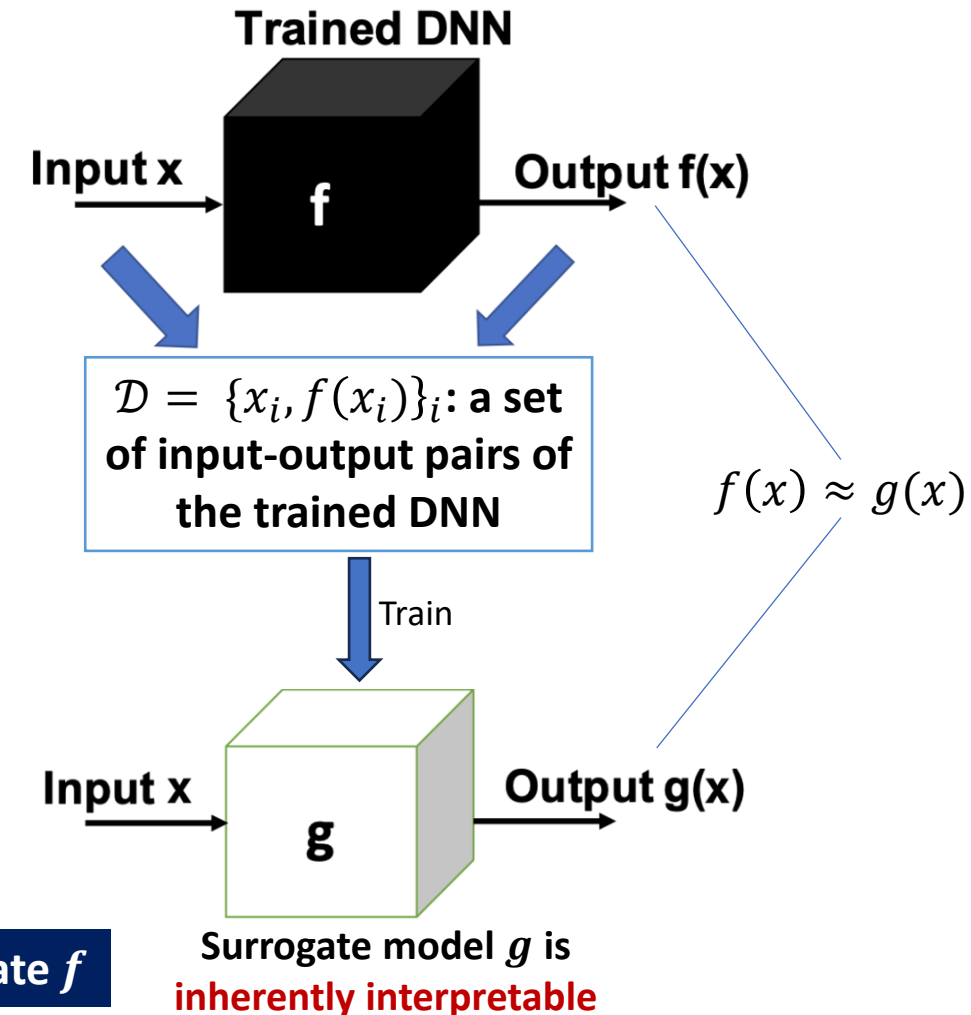
# Content

- 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  $g$ .



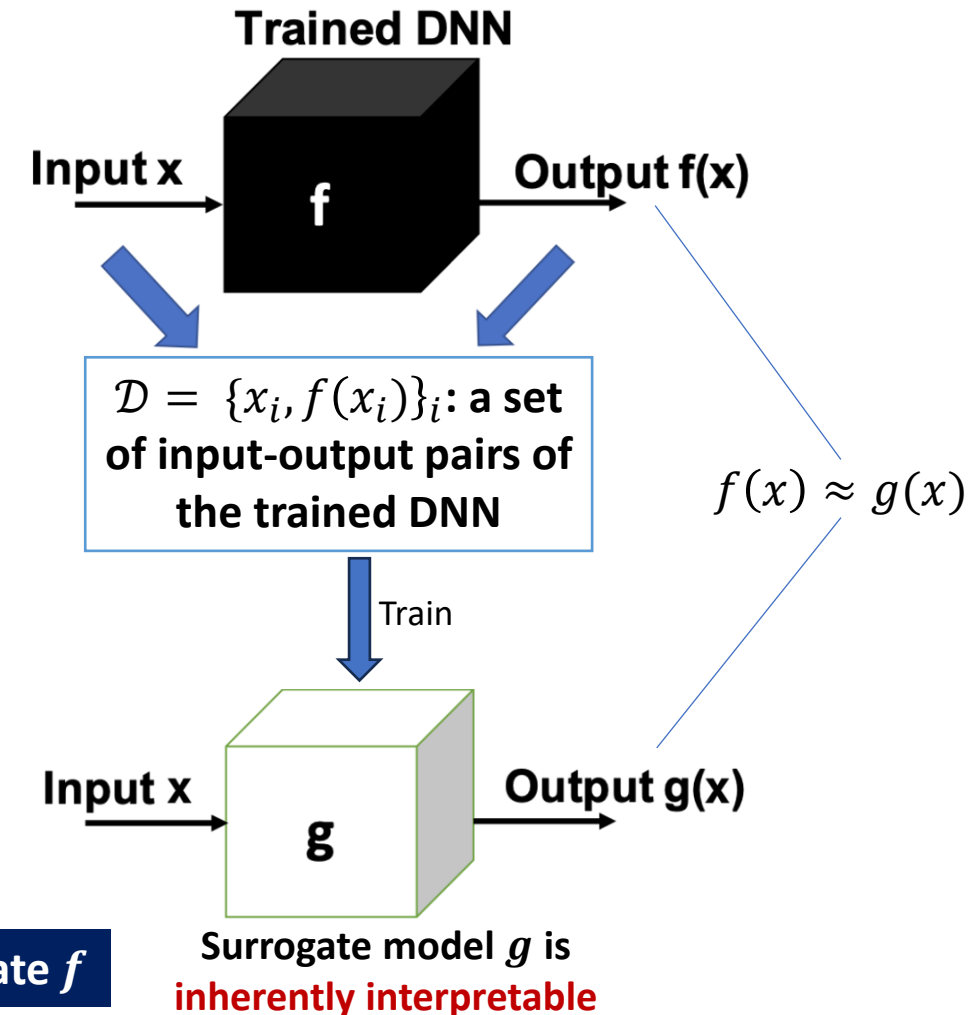
Use  $\mathcal{D} = \{x_i, f(x_i)\}_i$  as traing data for  $g$  to approximate  $f$

# Local Explanations with Surrogate Models

- **Explanation with Surrogate Model**

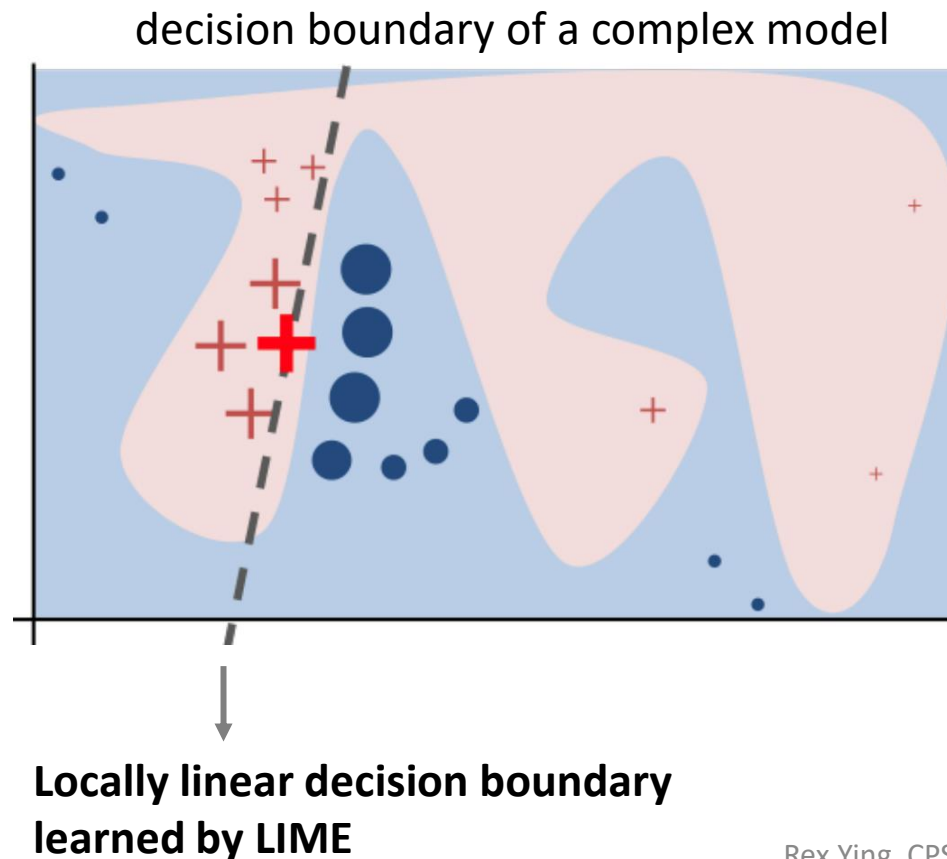
- Given input instance  $x_0$  and model  $f$ .
- Steps:
  1. Sample points around  $x_0$
  2. Use model to predict labels for each example  $(x_i, f(x_i))$
  3. Weight examples according to the distance to the input instance  $x_0$
  4. Learn a linear model on weighted samples
  5. Use simple linear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

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



# Intuition of LIME

- **Local Interpretable Model-Agnostic Explanations (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  $\Leftrightarrow$  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  $(x_i, f(x_i))$
  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

## 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

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

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

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

LIME targets at **local approximation**  $\Rightarrow$  samples  $z$  with relatively **larger  $\pi_{x_0}(z)$**  (i.e., smaller distance  **$D(x_0, x_i)$** ) 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  $(x_i, f(x_i))$
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  $x$ :

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

- $\mathcal{G}$  is the class of interpretable models (linear model or decision tree)
- $\mathcal{L}(f, g, \pi_{x_0})$  is the loss function 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  $x_0$ :

$$g^* = \operatorname{argmin}_g \mathcal{L}(f, g, \pi_{x_0}) + \boxed{\Omega(g)} \rightarrow \text{Control complexity of the interpretable model}$$

where

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

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

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 preferred! (Occam's 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**
  - $M$  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:**



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

$x$ : Original image

$x'$ : Interpretable representation

# Choice of Interpretable Representation

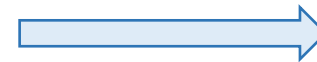
- Given an original input  $x \in \mathbb{R}^d$ , let  $x' \in \{0, 1\}^M$  denote a binary vector as its **interpretable representation**
  - In the context of images, a segmentation method can be used to decompose the image into interpretable components (see `skimage.segmentation`)



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

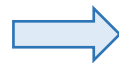
Mask corresponding to a  
super-pixel (segment)

# 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:

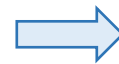


$x \in \mathbb{R}^d$   
Original image



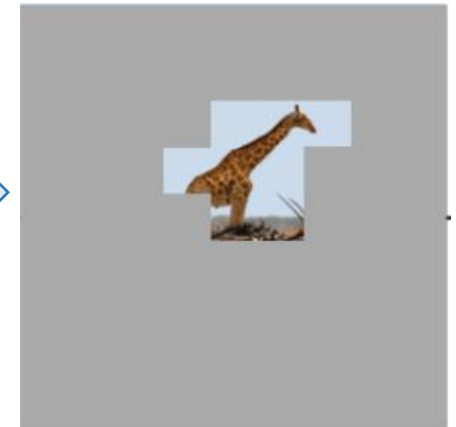
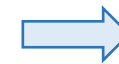
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	1	1	1	1	0	0	0	0
0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

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



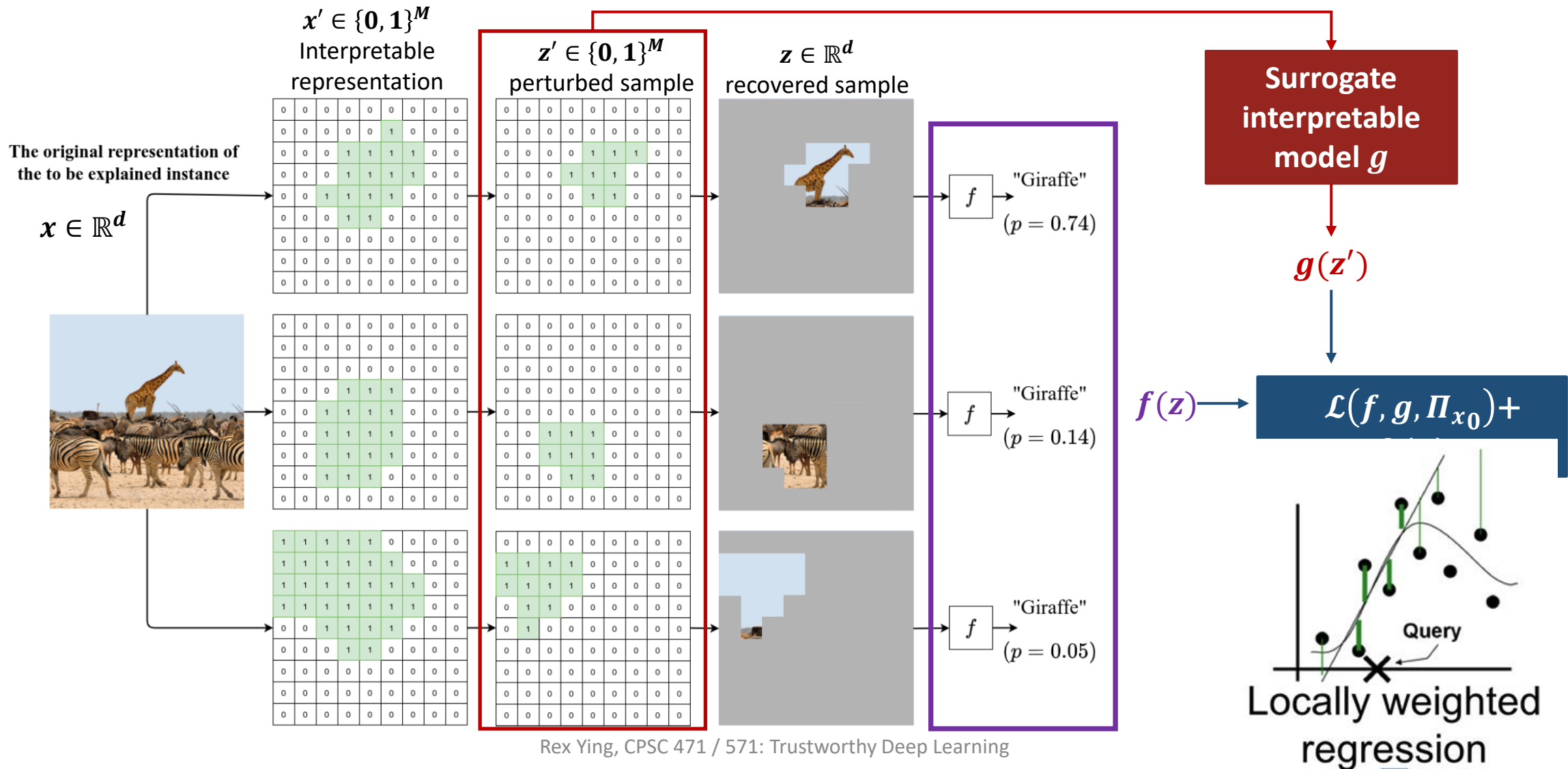
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$z' \in \{0, 1\}^M$   
**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	BOW: number of times each word occurs in a sentence
Input sentence	She loves pizza, pizza is delicious	1	1	2	1	1	0	0	0	0	0	0	0	

- 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

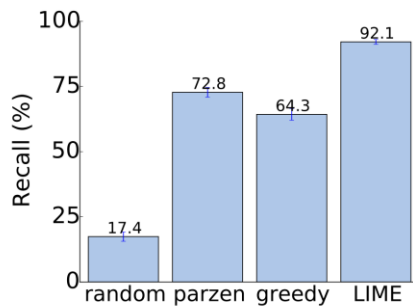


# Evaluation: important feature selection (2)

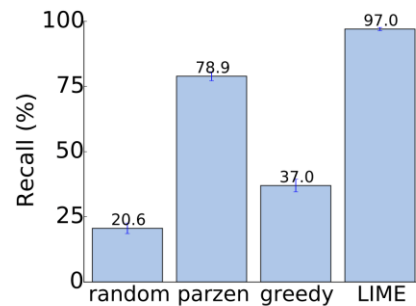
- **Baselines:**

- **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 Positive, FN: False Negative)

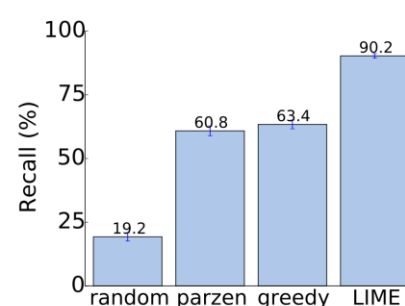


(a) Sparse LR

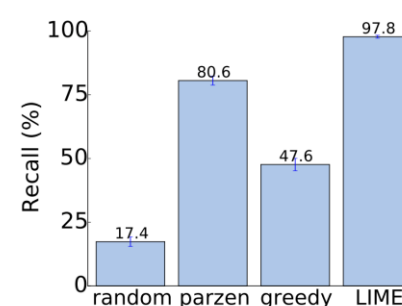


(b) Decision Tree

**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
  - $h_x: \{0,1\}^M \rightarrow \mathbb{R}^d$  recovering function that maps to original domain  $\mathbb{R}^d$

- **Surrogate model:**

$$g(z) = \overset{\text{null output}}{\boxed{\phi_0}} + \sum_{i=1}^M \overset{\text{Shapley value of the } i\text{-th feature}}{\boxed{\phi_i}} z_i$$

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

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?**

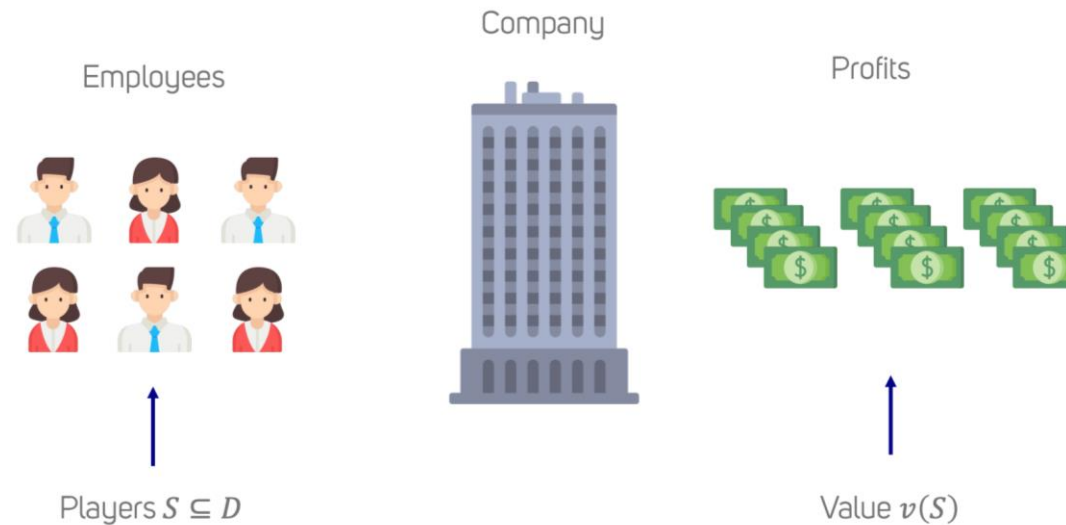
## Non-cooperative game

- 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  $\Rightarrow$  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

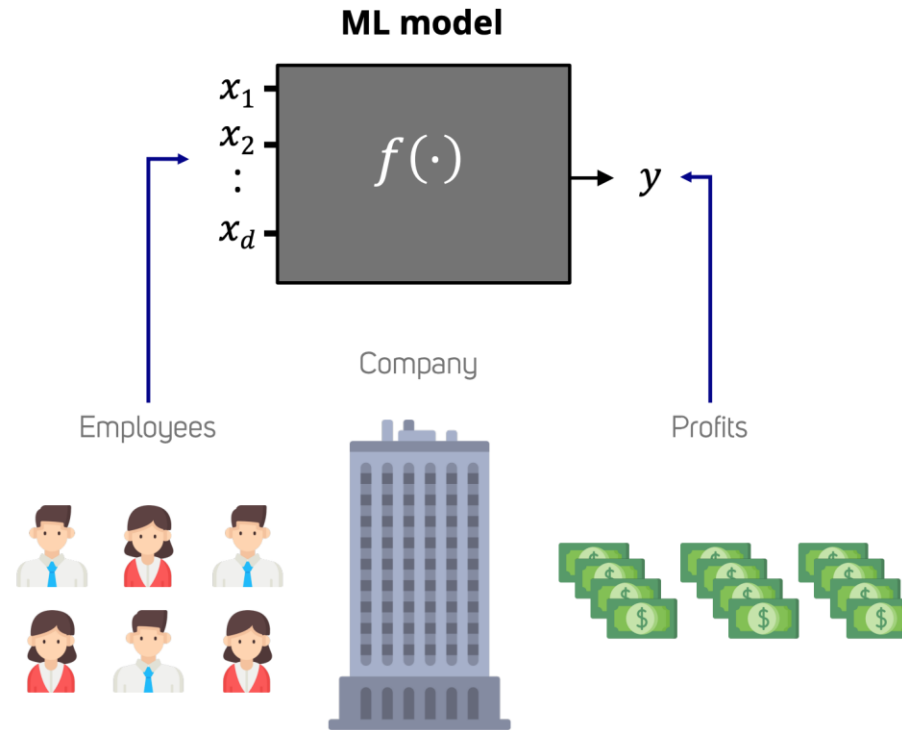
Every coalition with the same size is equally likely to appear

All sizes are equally likely to appear.



# Shapley Values in Cooperative Game Theory

- In context of a deep learning system:



**features  $\Leftrightarrow$  players      prediction  $\Leftrightarrow$  profit**

# Calculating Shapley Values

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

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

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

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

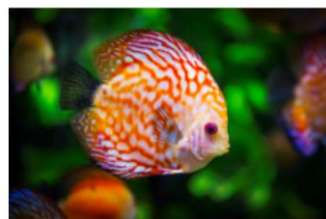
Very small  $|z'|$  (close to 0) or very large  $|z'|$  (close to  $M$ ) result in larger weights. Why?

- $z' \subseteq x'$  represents all  $z'$  vectors where the non-zero entries are a **subset** of the non-zero entries in  $x'$
- $|z'|$ : the number of non-zero entries in  $z'$
- $M$ : number of features in the **interpretable representation**

# Example of SHAP

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

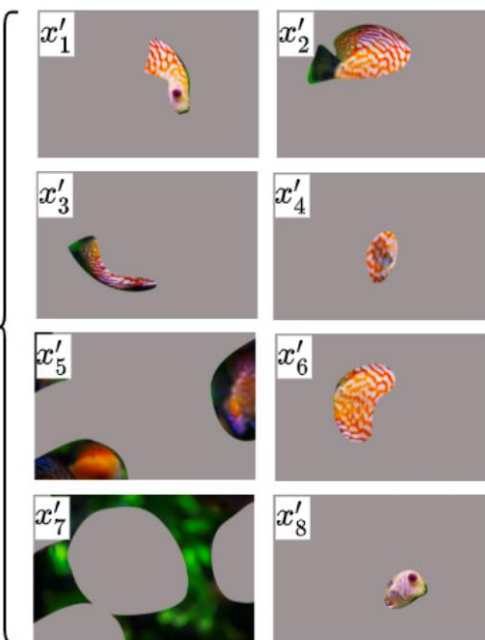
$x'$	$x_1$	$x_2$	$x_3$	$x_4$	.....
$z' \subseteq x'$	$x_1$	$x_2$	$x_3$	$x_4$	.....
$z' \setminus 4$	$x_1$	$x_2$	$x_3$	$x_4$	.....



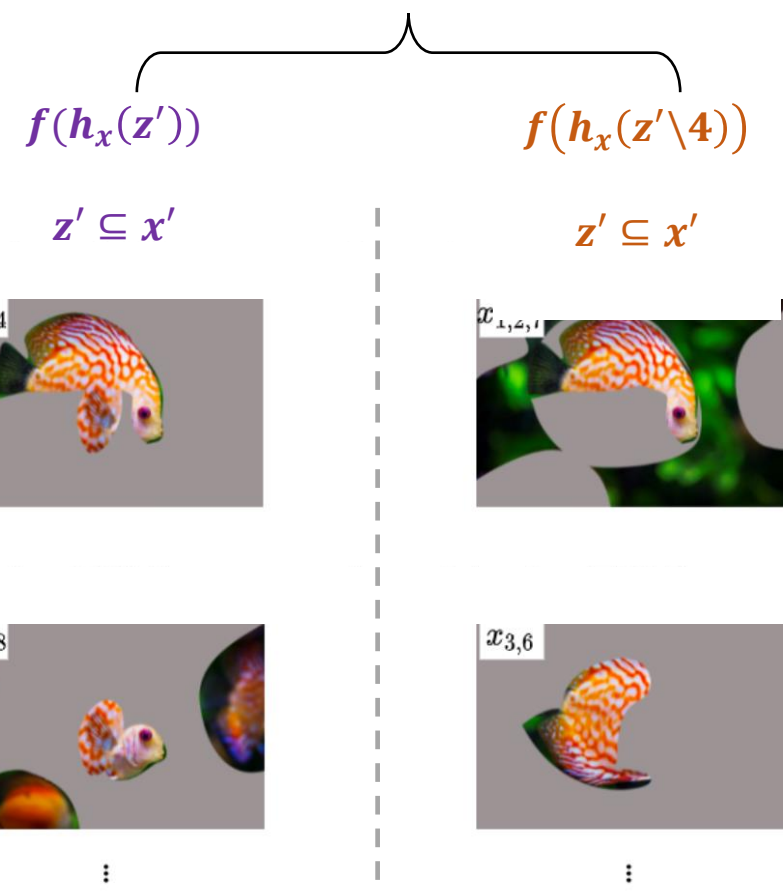
"Discus Fish"

Feature  
extractor

$x'$ : interpretable representation  
(8 features in total)



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





# 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

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

**The model output is fully distributed to all input features.**

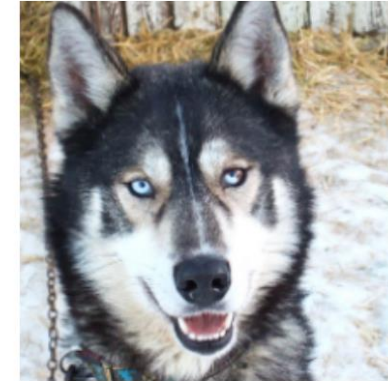
# Properties of SHAP (2)

- **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.

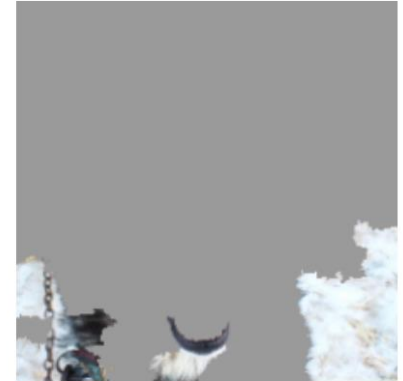
# What are Pros and Cons of SHAP?

## Questions!

- **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



(a) Husky classified as wolf



(b) Explanation

# The equivalence between LIME and SHAP

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

LIME	SHAP
<ul style="list-style-type: none"> <li>• LIME solves the following optimization problem</li> </ul> <div style="border: 1px solid orange; padding: 10px; margin: 10px 0;"> <math display="block">w^* = \operatorname{argmin}_w \sum_{x \in \mathcal{D}} \pi_x(z) [f(x) - (w^\top x + b)]^2 + \Omega(w)</math> </div> <p>where <math>\mathcal{D}</math> is the set of perturbations around <math>x_0</math></p>	<ul style="list-style-type: none"> <li>• Importance score using Shapley value</li> </ul> <div style="border: 1px solid red; padding: 10px; margin: 10px 0;"> <math display="block">\phi_i = \sum_{z' \subseteq x'} \frac{ z' ! (M -  z'  - 1)!}{M!} [f(\mathbf{h}_x(z')) - f(\mathbf{h}_x(z' \setminus i))]</math> </div> <p><math>x'</math> is the interpretable representation of <math>x</math></p>

If

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

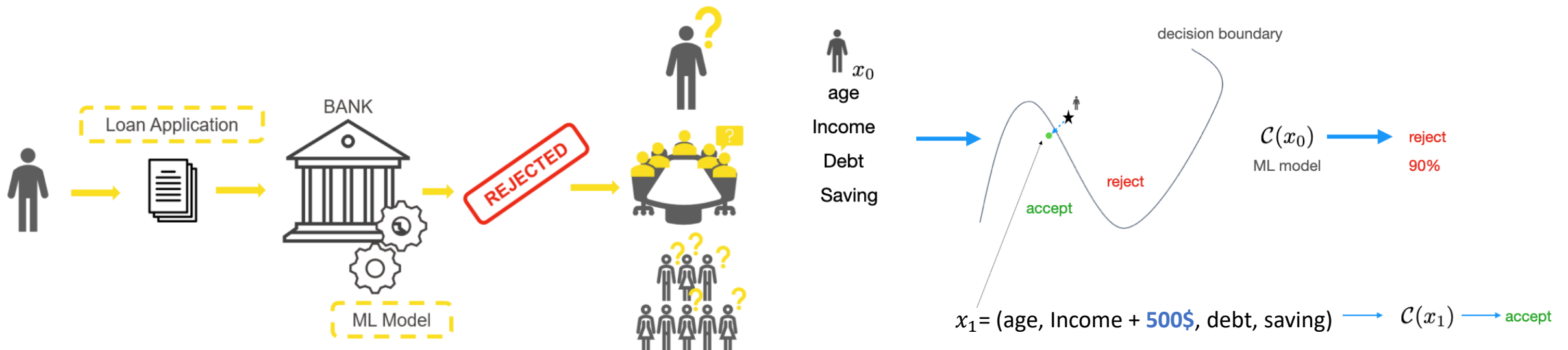
Then  $w^* = \phi_i$

# 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.



Rex Ying, CPSC 471 / 571: Trustworthy Deep Learning

**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)

- **Formulation:**

The diagram illustrates the mathematical formulation for finding a counterfactual explanation. It features the equation  $x_{CF} = \operatorname{argmin}_{x' \in \mathcal{X}} d(x_0, x')$  with constraints  $s.t. f(x') = y^{\text{target}}$ . Annotations include: a green arrow pointing from  $x_{CF}$  to the text 'Counterfactual Explanation'; a blue arrow pointing from  $d(x_0, x')$  to the text 'distance function (implementation cost)'; an orange arrow pointing from  $x' \in \mathcal{X}$  to the text 'Feasible action space'; and a red arrow pointing from the constraint equation to the text 'Constraint to get desired outcome'.

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

Counterfactual Explanation

distance function (implementation cost)

Feasible action space

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

Constraint to get desired outcome

# Counterfactual Explanations

- **Formulation:**

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

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

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

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

where

$$\mathcal{L}(f(x'), y^{\text{target}}) = (f(x') - y^{\text{target}})^2 \longrightarrow \text{Least Square Error}$$

Lagrange multiplier

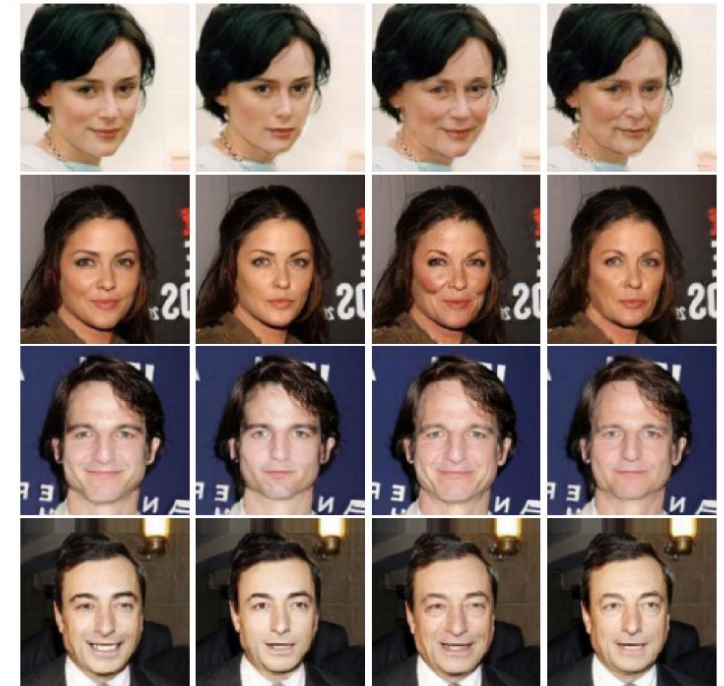
This problem is differentiable and can be solved by Projected Gradient Descent Algorithm



# Generative Approaches

- $x_{CF} = \operatorname{argmin}_{x' \in \mathcal{X}} d(x_0, x')$   
s.t.  $f(x') = y^{\text{target}}$
- $x'$  can be obtained through **generative models**
  - Variational autoencoder
  - Diffusion model
- **Objective 1**: reconstruction (also distribution loss)
- **Objective 2**: target constraint ( $y^{\text{target}}$ )

Young    Young    Old    Old  
Smiling   Serious   Smiling   Serious



# 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