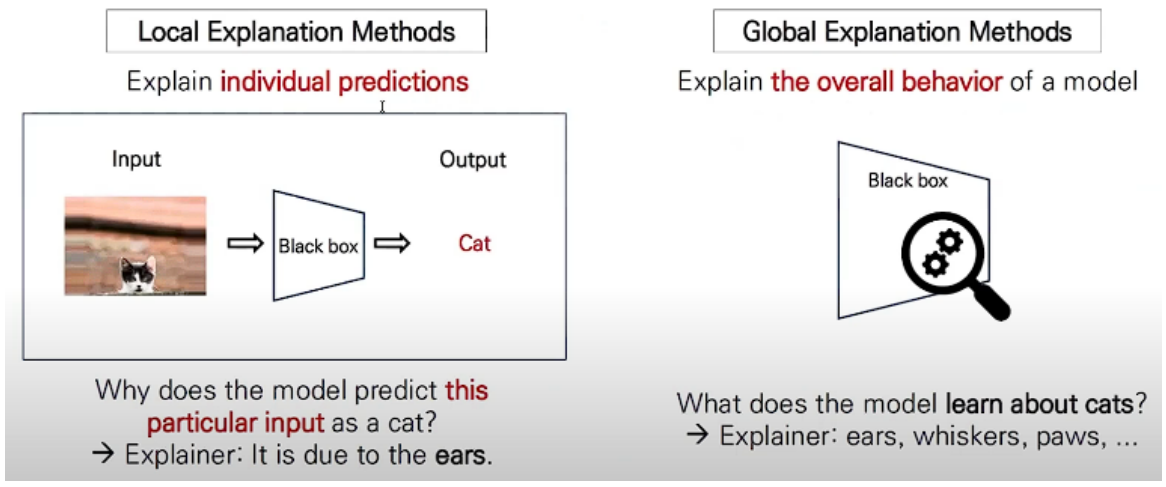
02_XAI Methods 1: Local Explanation Methods

1. What is the Local Explanation Method?

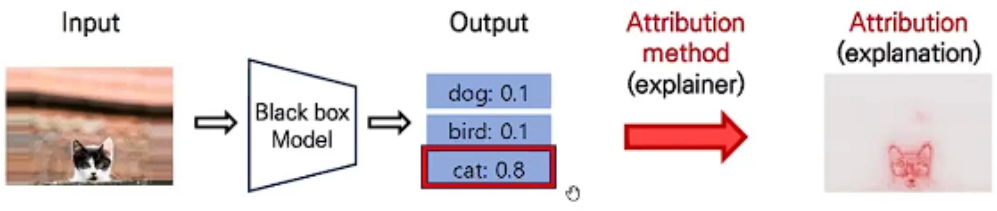
Local vs Global

- Local Explanation Methods는 개별 prediction을 설명한다고 생각하면 된다.
  - 특정 input에 대하여 왜 고양이라고 예측을 했는지에 대한 설명을 제공하는 방법.
- Global Explanation Methods는 개별 prediction이 아닌 전반적으로 살펴보는 방법이다.
  - 모델이 고양이를 분류하도록 학습이 되었다면, 전반적으로 고양이에 대해서 어떤 것을 학습을 하였는지
  - 여러 input을 넣었을 때, 평균적인 output이 어떻게 되는지



Local Explanation Methods

- 어떤 모델 function에 대한 전반적인 behavior가 아닌 특정 포인트에서의 prediction이 왜 그렇게 된 것인지 설명하는 방법이다.
- 설명의 방법은 매우 다양하다. 이번엔 가장 흔하게 사용되는 방법 중 하나인 Input attribution에 대해서 알아보자.
  - Input attribution : input의 여러 feature들이 각output을 만든 기여도를 계산



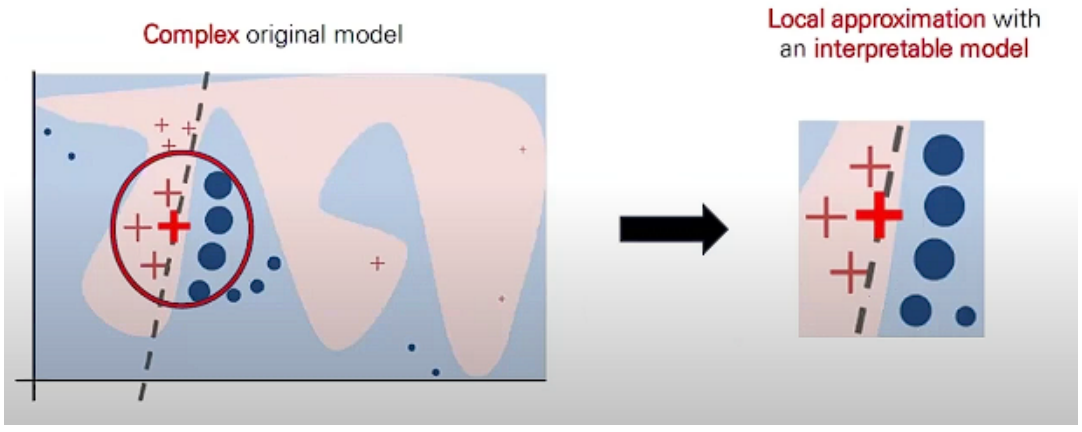
- 다양한 Local Explanation Methods들을 두 가지로 분류하여 살펴보자.
  - Model-Agnostic Approaches
    - 모든 모델에 적용할 수 있음
    - Ex) LIME, SHAP
  - Model-Specific Approaches
    - 특정 타입의 모델에만 적용할 수 있음
    - Ex) LRP, IG

2-1. Model-Agnostic Approaches - LIME

Main idea

- LIME의 main idea는 설명하고자 하는 input point들을 local하게 approximate하는 해석가능한 모델을 찾는 것이다.
- 해석 가능한 모델(interpretable model) : 그 자체만으로 설명이 되는 모델
  - Ex) Linear model, decision tree, ...
- 보통은 Linear model을 interpretable model로 선정하고, 이 모델을 통해서 설명을 제공하는 것이 LIME이다.

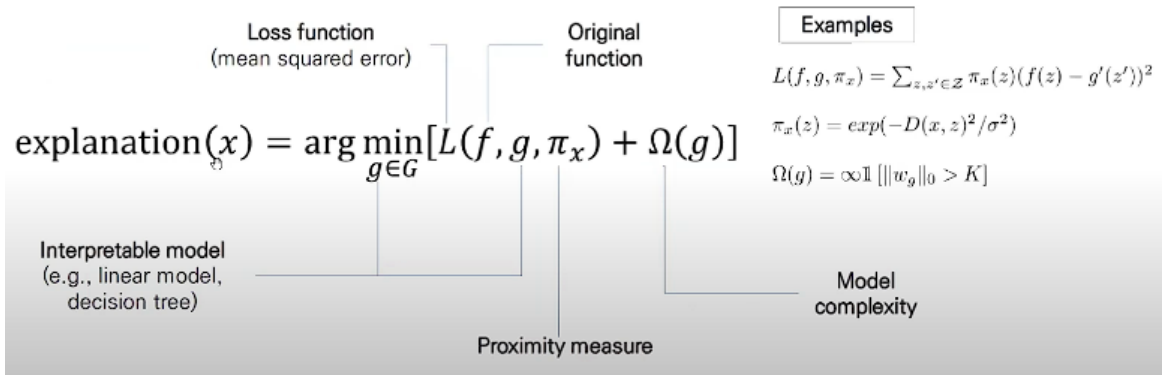
- 아래의 그림을 살펴보자.
  - 복잡한 원래 모델의 decision boundary는 왼쪽과 같이 상당히 복잡할 것이다.
  - 여기서 특정 point의 근방에서 국소적으로(=local하게) 근사를 잘 할 수 있는 linear model을 찾아서 설명을 제공하는 것이다.



Mathematical formulation

- x : 설명이 필요한 input x
- explanation(x) : x에 대한 prediction의 설명
- f : 원래 사용한 original function
- g : 찾아야 할 interpretable model
- π_x : proximity measure, local 하게 근사해야 하므로 x point와 가까울수록 가중치를 높게 주고, 멀수록 낮게 주는 것
- $\Omega(g)$: g를 f에 너무 fitting하다 보면 오히려 너무 복잡해질 가능성이 있으므로 모델의 복잡도를 penalize하는 term이다

- 기본적으로 Loss function은 MSE를 사용한다.
  - $L(f, g, \pi_x) + \Omega(g)$를 최소화하는 모델 g를 찾아야 한다.



Overall process

1. Select your instance of interest

2. Generates a dataset by perturbation
3. Weight the perturbed samples
4. Find an interpretable model that fits the generated dataset

Example

classification of YouTube comments as spam(1) or normal(0)

1. select a sentence for which you want to explain the prediction

	CONTENT	CLASS
267	PSY is a good guy	0
173	For Christmas Song visit my channel! ;)	1

2. Perturb the sentence : 문장에 있는 단어를 랜덤하게 몇 개를 없앤다.

For	Christmas	Song	visit	my	channel!	;)	prob	weight
1	0	1	1	0	0	1	0.17	0.57
0	1	1	1	1	0	1	0.17	0.71
1	0	0	1	1	1	1	0.99	0.71
1	0	1	1	1	1	1	0.99	0.86
0	1	1	1	0	0	1	0.17	0.57

→ Christmas Song visit ;)

3. Get the predicted probability of spam for each perturbed sentence
4. Weight the perturbed sentences
    - 이 예시에서는 Weight = 1 - fraction of removed tokens 이렇게 weight를 정의하는데, 지워진 부분이 많을 수록 원래 문장 즉, input과 많이 달라졌다고 볼 수 있으므로 많이 다를수록 weight를 작게 주는 방식으로 사용하는 것이다.

For	Christmas	Song	visit	my	channel!	;)	prob	weight
1	0	1	1	0	0	1	0.17	0.57
0	1	1	1	1	0	1	0.17	0.71
1	0	0	1	1	1	1	0.99	0.71
1	0	1	1	1	1	1	0.99	0.86
0	1	1	1	0	0	1	0.17	0.57

5. Train an interpretable model with the perturbed data

case	label_prob	feature	feature_weight
1	0.1701170	is	0.000000
1	0.1701170	good	0.000000
1	0.1701170	a	0.000000
2	0.9939024	channel!	6.180747
2	0.9939024	;)	0.000000
2	0.9939024	visit	0.000000

LIME Pros & Cons

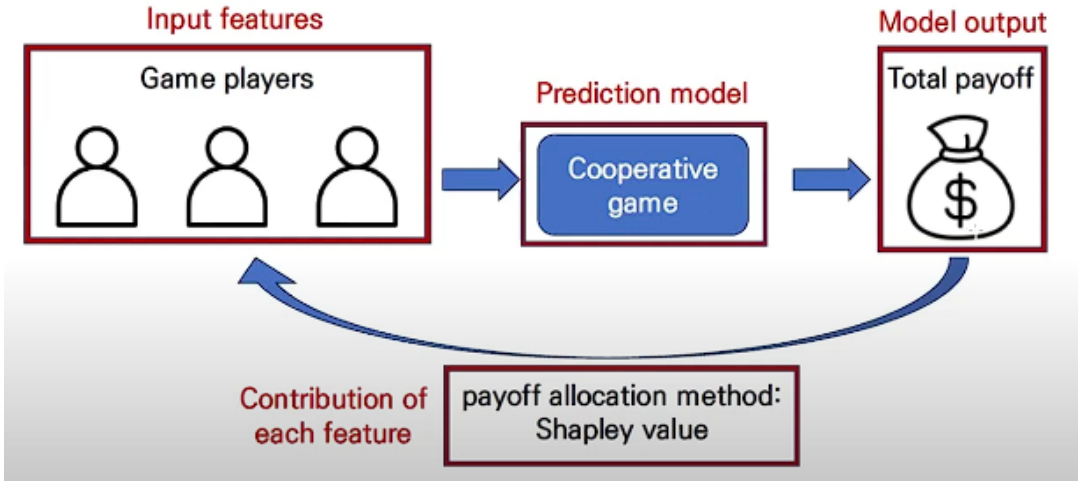
1. Pros
    - 어떤 모델에도 적용할 수 있다.
    - 사용하기 쉽다.

2. Cons
    - proximity measure를 어떻게 정의해야 할 지가 어렵다.
    - perturbation은 out of distribution data가 나올 수도 있다.

2-2. Model-Agnostic Approaches - Shapley Value

Shapley Value

- Shapley Value는 협력적 게임 이론에서 나온 개념이다.
- 게임 플레이어들이 서로 협력을 통해 게임을 참여하여 상금을 얻었다고 했을 때, 각 플레이어의 기여도에 따라 상금을 공정하게 분배하는 방식이다.
- 이를 머신러닝 모델에 적용해보자면,
    - Input features = Game players
    - Prediction model = Cooperative game
    - Model output = Total payoff
    - payoff allocation method = Contribution of each feature



- 그렇다면 '공정'하게 배분할 때 '공정함(=fairness)'를 어떻게 정의할 것인가?
- 4 Axioms
    - Linearity
    - Dummy
    - Symmetry
    - Efficiency
- 위의 4가지 조건을 만족하는 값은 Shapley value밖에 없다.

4 Axioms

- $\phi_i(v)$: Value(payoff allocation) of player i in the game v
- $v(\emptyset)$ = 0
- N : The entire set of players

1. **Linearity**

- ϕ is linear with respect to the game v

$$\phi_i(v_1) + \phi_i(v_2) = \phi_i(v_1 + v_2)$$
$$\phi_i(cv) = c\phi_i(v)$$

2. **Dummy(Null player)**

- A player that does not contribute to the output get the zero value

$$\forall S \subseteq N \setminus i : v(S \cup i) = v(S) \quad \Rightarrow \quad \phi_i(v) = 0$$

3. **Symmetry**

- if player i and j contribute equally to all subsets S, they get the same value

$$\forall S \subseteq N \setminus \{i, j\} : v(S \cup i) = v(S \cup j) \quad \Rightarrow \quad \phi_i(v) = \phi_j(v)$$

4. **Efficiency**

- The values add up to the total payoff

$$\sum_{i \in N} \phi_i(v) = v(N)$$

Formula

유일하게 4가지 조건을 만족하는 Shapley value의 formula는 아래와 같다.

이 식의 의미는 marginal contribution의 weighted average이다.

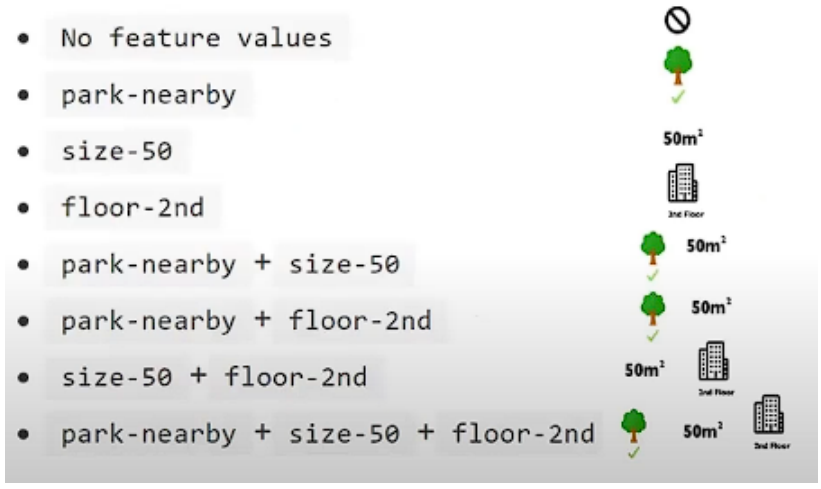
$$\phi_i(v) = \underbrace{\sum_{S \subseteq \{1,2,\ldots,p\} \setminus \{i\}} \frac{|S|!(p-|S|-1)!}{p!}}_{\text{(weighted) average}} \underbrace{(v(S \cup \{i\}) - v(S))}_{\text{marginal contribution}}$$

예시를 통해 알아보자

- 아래의 예시는 아파트 가격이다.
  - input feature로는 공원이 근처에 있는지, 50평, 고양이를 키울 수 있는지 이렇게 3가지가 있다.
  - 이 input feature들을 통해 아파트 가격을 예측한다.
  - 나머지 feature들은 동일하고, 고양이를 키울 수 있는 곳과 없는 곳으로 구분되는 두 곳의 가격 차이가 존재하는 것을 알 수 있다.
  - 이것이 바로 cat-banned feature의 marginal contribution이다.

- 하지만, marginal contribution은 어떤 feature가 존재하는지 여부에 따라 계속해서 달라진다.
- 그렇기 때문에 모든 케이스를 모두 고려하는 것이 Shapley value의 핵심이다.
- 아래와 같이 모든 케이스의 marginal contribution을 계산하고, 가중치를 주어 모두 합한 값이 Shapely value가 되는 것이다.

→ **모든 상황에서의 기여도를 모두 고려한 분배방식**



Problems when applied to machine learning models

하지만 Shapley value를 머신러닝 모델에 적용할 때 크게 두 가지 문제가 있다.

1. **Computationally intractable**
   - 모든 부분 집합에 대하여 계산해야 하므로, feature수가 많으면 부분 집합의 수가 너무 많기 때문에 정확한 계산이 어렵다.

2. Input feature absent
   - 학습하고자 하는 모델이 3개의 input features가 필요한데, marginal contribution을 계산할 때 input feature의 수가 1개가 되거나 2개가 되어버리면 학습할 수 없게 되어버림.

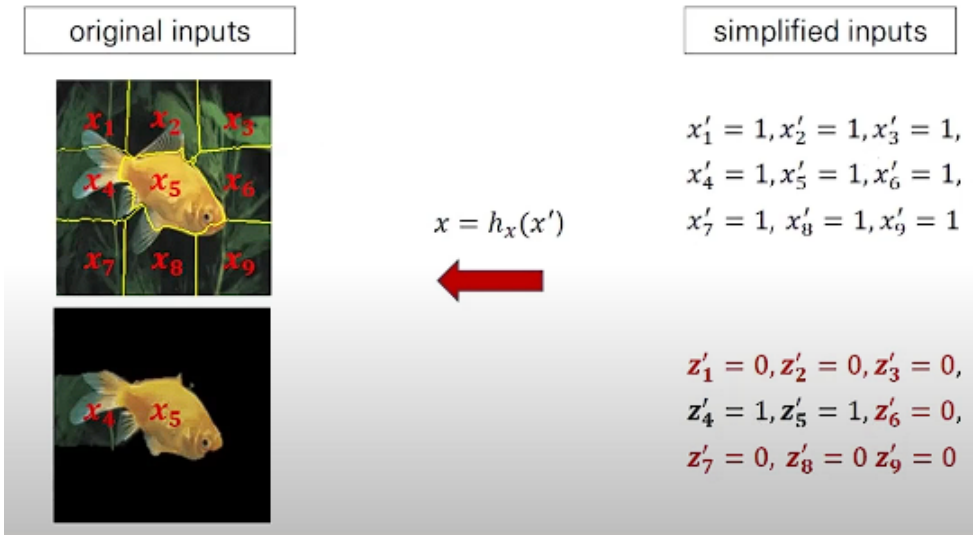
2-3. Model-Agnostic Approaches - SHapley Additive exPlanations(SHAP)

SHAP은 간단히 설명하자면 linear model인데 계수가 Shapley value인 것이다.

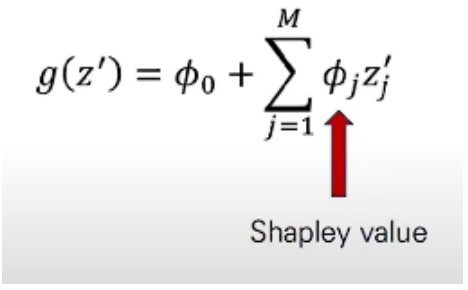
여기서 simplified input domain이라는 개념이 나온다.

Simplified inputs

- 어떤 feature가 있는지 없는지에 따라서 1 or 0을 부여하는 binary variable이다.
- x'과 같이 프라임 표기를 사용한다.
- 어떤 feature가 있는지 없는 지를 직관적으로 알 수 있기 때문에 simplified input domain을 사용한다.



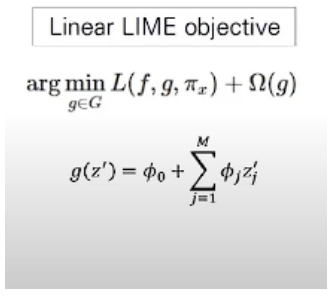
- 아래와 같은 SHAP 식에서 simplified input domain이므로 z_j'은 1 or 0의 값을 갖는다.
- 이 값이 1이라면 ϕ_j만 남고, 이는 Shapley value값이다.
- 즉, SHAP는 Shapley value를 기여도로 보는 방법이라는 것을 알 수 있다.

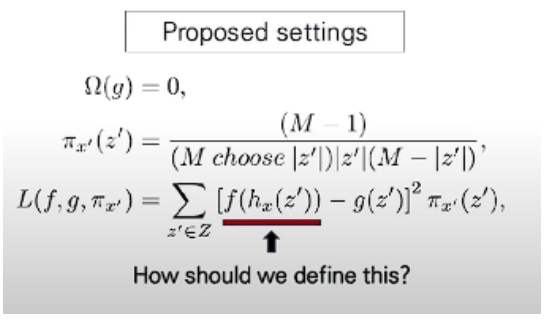


그렇다면 SHAP는 위에서 봤던 Shapley value의 2가지 문제점을 어떻게 해결했을까?

KernelSHAP - Shapley value의 계산 복잡성 문제 해결

- KernelSHAP은 LIME의 로컬 선형 근사 아이디어를 Shapley value값 계산에 적용하여 설계된 것으로, LIME의 효율성과 Shapley value의 이론적 공정성을 결합한 방법이다.
- 왼쪽의 LIME의 목적함수 구조를 가져와서 오른쪽의 Proposed setting을 통해 계산량을 줄이면서 Shapley value를 근사한 것이다.
- Proposed settings을 살펴보자.
  - $\Omega(g) = 0$: 정규화 항을 제거하여 최적화 문제를 단순화
  - $\pi_x(z')$: 각 데이터 샘플 z'에 가중치를 부여하여 Shapley 값을 계산
  - $L(f, g, \pi_x)$: 목적 함수는 LIME과 동일한 구조를 유지하면서, Shapley 값을 계산하기 위해 각 데이터 샘플의 가중치를 $\pi_x(z)$로 대체





- KernelSHAP이 LIME에서 가져온 핵심 개념
  1. 로컬 선형 근사
  2. 샘플링 기반 접근
  3. 손실 최소화

- KernelSHAP의 독창성

1. Shapley value 기반 가중치 설계
2. 정규화 항 제거
3. 특성 상호작용 고려

→ 즉, KernelSHAP는 LIME의 샘플링 기반 접근법을 가져와서 모든 부분집합을 탐색하지 않고 일부만 샘플링하여 계산함으로써 계산의 복잡성 문제를 극복하였다.

→ 또한, Shapley value의 특성 조합별 기여도를 근사하기 위해 커널 가중치를 설계하여 Shapley value를 계산할 때 특성 조합의 중요성을 반영했다.

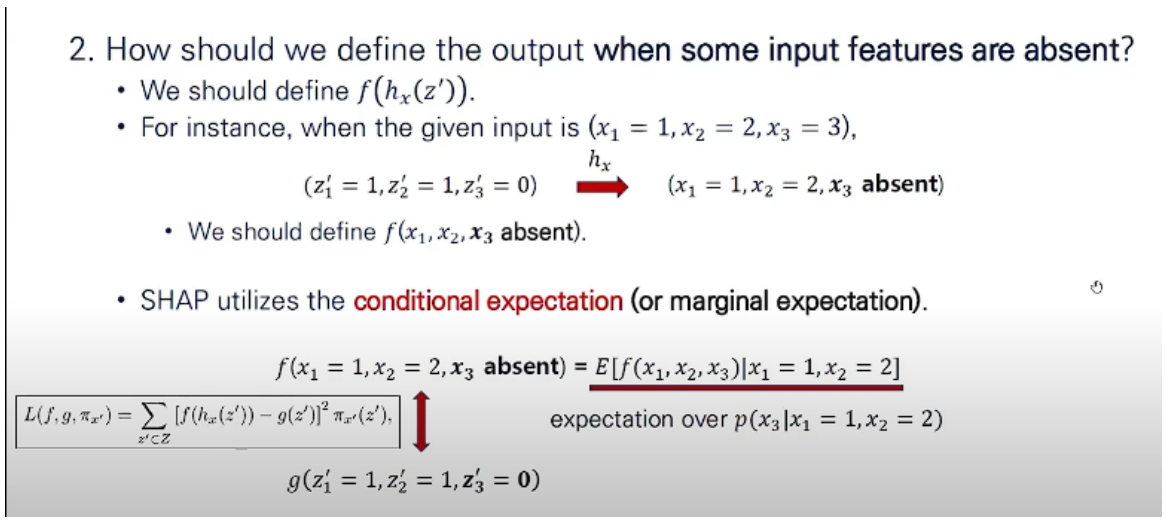
→ 그렇게 샘플링된 부분집합과 커널 가중치를 사용하여 '가중 선형 회귀'를 수행함으로써 Shapley value를 근사한다.

Marginal expectation - input feature absent 문제 해결

- SHAP는 결측된 특성을 처리하기 위해 '조건부 기대값 'or '주변 기대값'을 사용한다.

$f(x_1 = 1, x_2 = 2, x_3\ absent) = E[f(x_1, x_2, x_3)|x_1 = 1, x_2 = 2]$

- 결측된 특성 x_3의 값에 대해, 나머지 활성된 특성 x_1, x_2의 정보를 기반으로 조건부 기대값을 계산한다.
- 여기서 조건부 기대값은 결측된 특성 x_3가 확률 분포에 따라 가질 수 있는 모든 값을 고려한 모델 출력의 평균값이다.



Result of SHAP

- SHAP의 결과는 특정 예측값 $f(x)$이 어떻게 도출되었는지, 각 특성이 평균 예측값 $E[f(x)]$에 비해 얼마나 기여했는지를 나타낸다.
  - $E[f(x)]$: 평균 예측값(Baseline), 모든 입력 데이터에서 모델이 평균적으로 예측하는 값
  - $f(x)$: 최종 예측값, 특정 데이터 포인트에 대해 모델이 계산한 최종 출력값
  - SHAP 값 : 각 특성이 평균 예측값에서 최종 예측값으로 이동하는 데 기여한 정도를 나타낸다.

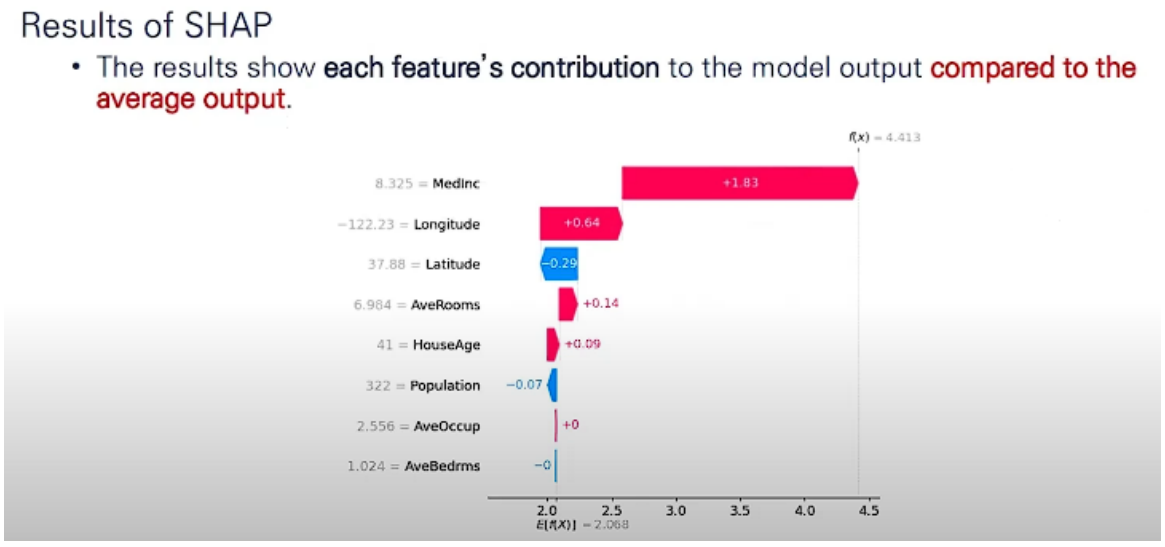
[SHAP값 계산 과정]

- Baseline (E[f(x)]=2.068)에서 시작.
- 각 특성의 SHAP 값을 순차적으로 더하거나 빼면서 최종 예측값 (f(x)=4.413)에 도달한다.

[특성 별 기여도]

1. **MedInc (Median Income)**:
   - +1.83+1.83+1.83: 가장 큰 기여도. 예측값을 1.83만큼 증가시킴.
   - 이는 소득 수준이 예측값에 매우 긍정적인 영향을 미친다는 것을 의미함.
2. **Longitude (경도)**:
   - +0.64+0.64+0.64: 경도 값이 예측값을 증가시키는 데 기여함.

3. **Latitude (위도)**:
    - −0.290.29−0.29: 위도가 예측값을 감소시키는 데 기여함.
4. **AveRooms (평균 방 개수)**:
    - +0.14+0.14+0.14: 약간의 기여도를 통해 예측값을 증가시킴.
5. **HouseAge (주택 연령)**:
    - +0.09+0.09+0.09: 예측값에 긍정적인 영향을 줌.
6. **Population (인구)**:
    - −0.070.07−0.07: 인구가 예측값을 약간 감소시킴.
7. **AveOccup (평균 거주 인구)** 및 **AveBedrms (평균 침실 개수)**:
    - 이 두 특성은 예측값에 거의 영향을 미치지 않았음



SHAP Pros & Cons

1. Pros
    - Solid theoretical foundation
    - SHAP connects LIME and Shapley values

2. Cons
    - KernelSHAP is slow
    - KernelSHAP ignores feature dependence

3. Model-Specific Approaches

3-1. Layer-wise Relevance Propagation(LRP)

Main idea

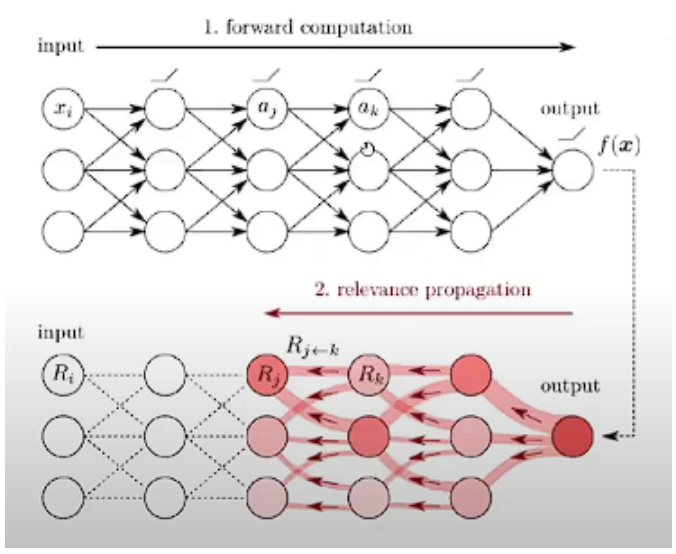
- LRP는 Deep Neural Network의 계층 구조를 활용한다.
- 모델의 출력인 예측값을 각 뉴런과 연결된 가중치와 활성화 값을 바탕으로 입력 데이터로 다시 분배하는 방식이다.

- Relevance Score
    - 각 뉴런이 모델의 최종 출력에 얼마나 기여했는지를 나타내는 점수이다.
    - 즉, 특정 입력 데이터가 모델 출력값에 미치는 영향을 수치로 표현한 것이다.

- Sequential Redistribution(순차적 분배)
  - 모델의 출력값을 모델의 네트워크의 역방향으로 전파하면서, 각 뉴런에 기여도를 순차적으로 할당한다.
  - 각 뉴런의 Relevance Score는 그 뉴런과 연결된 가중치 및 입력 신호를 기준으로 계산된다.

[Relevance Propagation 동작 원리]

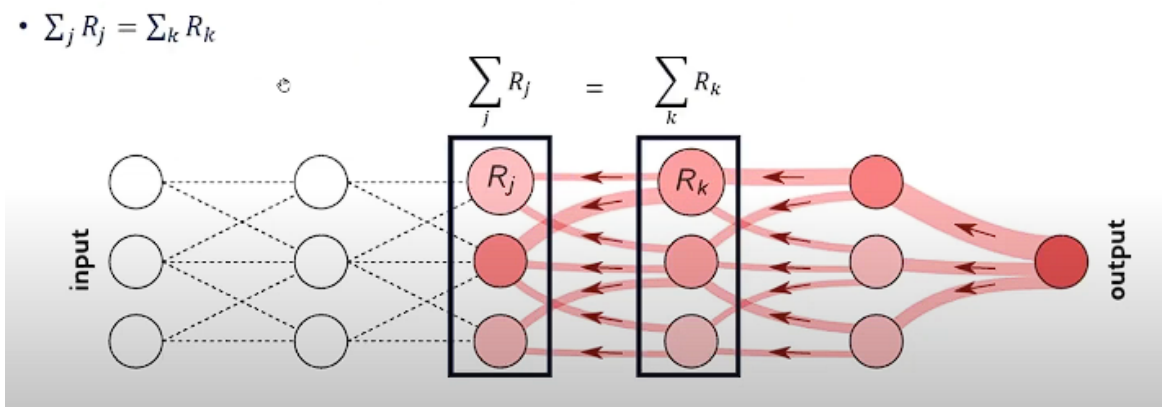
1. 출력값에서 시작 : 모델의 예측 결과(출력값)를 초기 점수로 설정한다.
2. 계층을 따라 역전파 : 출력값에서 입력값으로 역방향으로 이동하며, 각 뉴런에 대한 관련성을 계산하여 이전 계층으로 전파한다.
3. 관련성 점수 계산 : 각 뉴런의 관련성 점수는 해당 뉴런이 모델 출력에 기여한 비율에 따라 할당된다.
4. 입력값으로 분배 : 최종적으로 모든 관련성 점수는 입력값에 분배되어, 모델이 특정 입력값에 대해 왜 특정 예측을 했는지를 설명한다.



Conservation Property

- LRP의 중요한 속성인 보존 속성은 관련성 점수의 합이 모든 계층에서 동일하게 유지되는 것을 의미한다.
- 관련성 점수의 총합이 모든 계층에서 동일하게 유지된다는 것은 모델 출력값(예측값)을 각 계층에 걸쳐 입력 특성으로 정확히 다시 분배할 수 있다는 것을 보장한다.
- 아래의 수식을 살펴보자.

  $\sum_j R_j = \sum_k R_k$
  - R_j : 특정 계층에서 뉴런 j의 관련성 점수
  - R_k : 다음 계층에서 뉴런 k의 관련성 점수
  - 즉, 한 계층에서 모든 뉴런의 관련성 점수를 합한 값은 다음 계층에서도 동일하다.



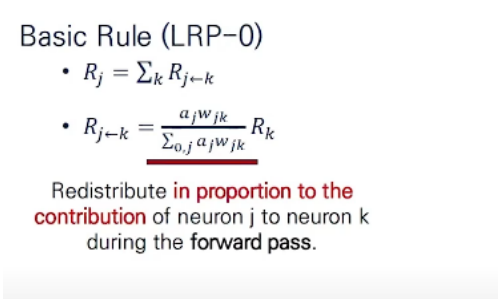
Rules for redistribution

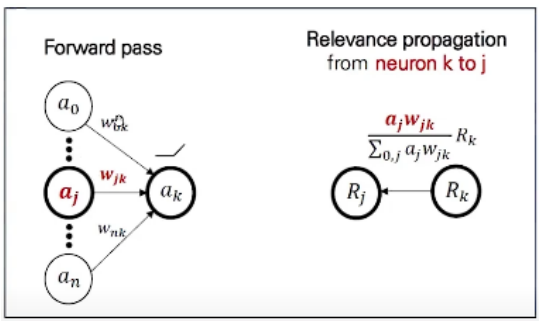
- 분배하는 방식은 굉장히 다양하다.
- 이번에는 Basic, Epsilon, Gamma Rule 이렇게 3가지를 살펴보자.

1. Basic Rule(LRP-0)

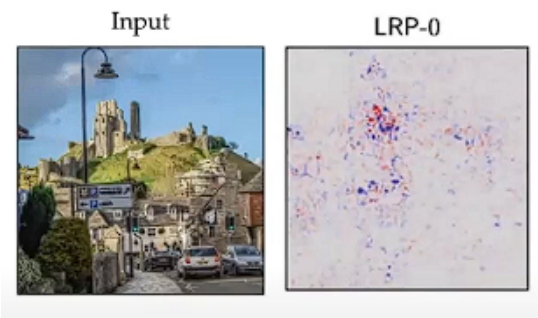
- LRP-0은 관련성 점수를 출력 뉴런에서 입력 뉴런으로 전파하는 방식을 정의한다.
- 아래의 왼쪽 그림을 살펴보자.
  - 첫 번째 수식은, 뉴런 j의 관련성 점수는 다음 계층의 모든 뉴런 k에서 전파된 관련성 점수의 합이라는 것을 보여준다.
  - 두 번째 수식은, 뉴런 k에서 j로 전파되는 관련성 점수는 뉴런 j가 k에 기여한 정도에 비례한다는 것을 보여준다.

- 아래의 오른쪽 그림은 각 뉴런의 활성화 값과 가중치를 사용하여 이전 뉴런이 다음 계층의 뉴런에 기여한 만큼 관련성 점수를 전파한다는 것을 보여준다.



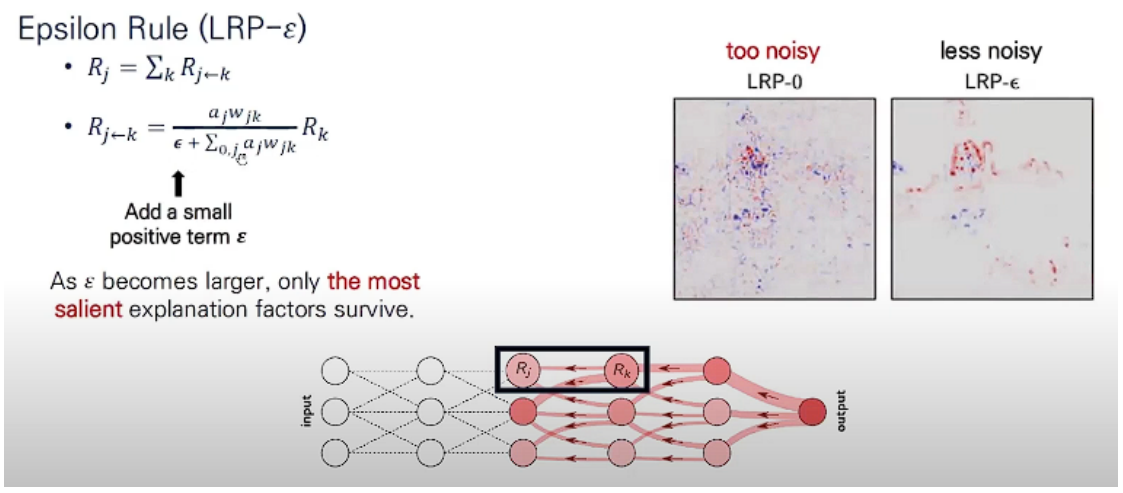


- 즉, LRP-0의 핵심은 이전 뉴런이 다음 계층의 뉴런에 기여한 '비율'에 따라 분배된다는 것이다.
- 오른쪽 사진은 input으로 castle class를 넣었을 때 LRP-0의 결과인데, 많이 noisy한 것을 알 수 있다.



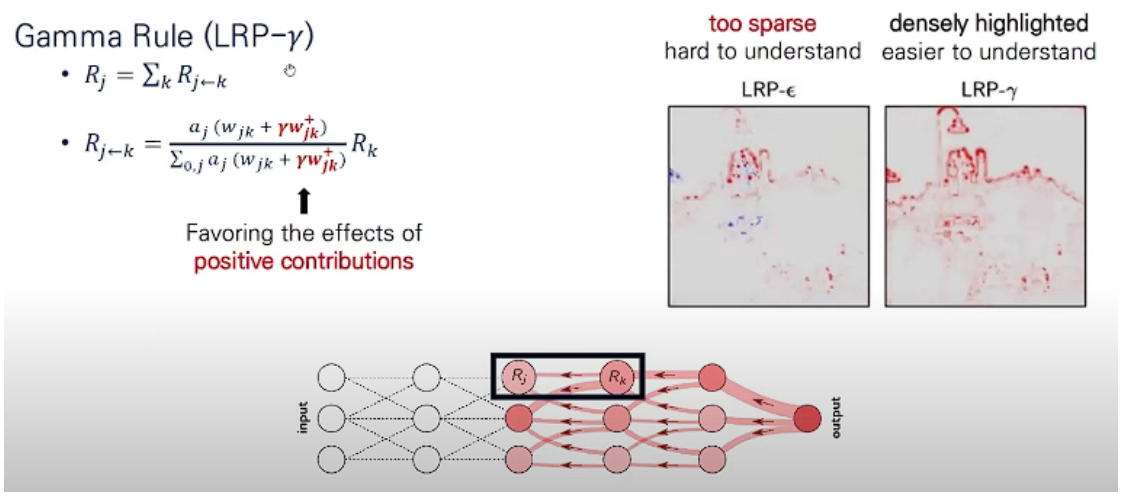
2. Epsilon Rule(LRP-ε)

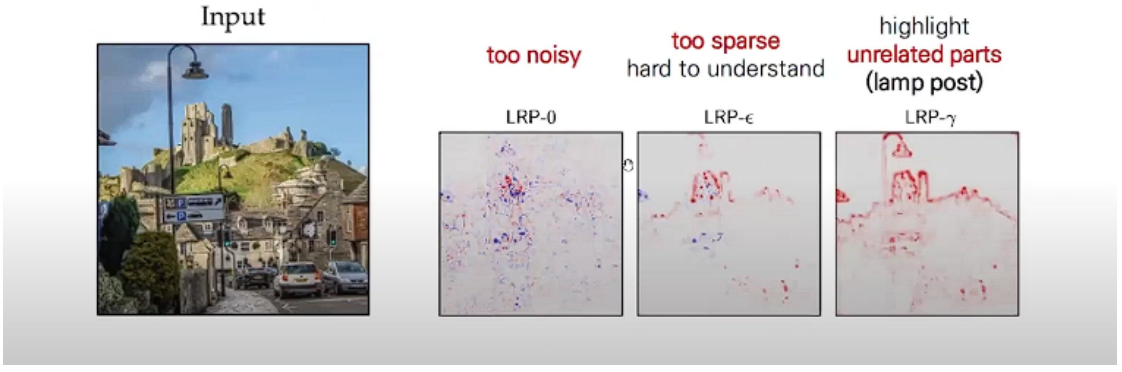
- LRP-ε은 LRP-0의 규칙에서 작은 양수 ε을 추가하여, 관련성 점수 계산을 아래와 같이 정의한다.
- ε은 작은 양의 값을 추가하여 계산과정에서 분모가 0이 되거나, 지나치게 작은 값이 되는 것을 방지하여 노이즈를 제거하고 더 명확한 설명을 제공하는 역할을 한다.
- 즉, 설명에 기여도가 낮은 요소들은 무시되고, 기여도가 큰 요소들이 강조된다.



3. Gamma Rule(LRP-γ)

- LRP-γ는 양의 기여도를 더 강조하는 방법이다.
- 아래 수식을 보면 양의 기여도를 더 강조하기 위해 γ라는 양수 파라미터를 추가한다.
  - 양수 가중치가 더해지므로, 양의 기여도가 더 두드러지게 나타난다.
- 양의 기여도가 강조됨으로써 모델이 특정 입력 데이터에 대해 긍정적으로 작용한 특성을 더 명확히 드러낸다.
- 이전의 LRP-ε에서는 시각화가 너무 희미하거나 이해하기 어려운 경우가 있었는데, LRP-γ는 시각적으로 더 풍부하고 직관적인 설명을 제공한다.





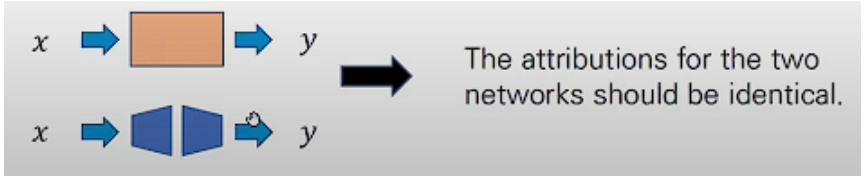
3-2. Integrated Gradients(IG)

- IG의 동기에 대해 알아보자.
- IG는 입력 기여도 계산 방법이 다음 두 가지 공리를 충족해야 한다고 주장한다.

1. **Sensitivity(a)**
   - 만약 하나의 입력 특성을 변경했을 때 모델의 출력값이 변한다면, 해당 특성은 '0이 아닌 기여도'(Non-zero Attribution)를 가져야 한다.
   - 즉, 출력값 변화에 기여한 특성은 반드시 중요하게 간주해야 한다는 것이다.

   Ex)
   - 모델 출력:
     - f(1,1,1)=10 (첫 번째 입력 특성이 1일 때 출력값은 10)
     - f(0,1,1)=1 (첫 번째 입력 특성을 0으로 변경했을 때 출력값은 1)
   - 분석:
     - 첫 번째 입력 특성(값 1 → 0)의 변화가 모델 출력값에 큰 영향을 미쳤다(10→1).
     - 따라서, **첫 번째 특성은 0이 아닌 기여도**를 가져야 한다.

2. **Implementation Invariance**
   - 아래 그림과 같이 두 개의 네트워크가 다른 구조 또는 다른 구현 방식을 갖고 있고, 동일한 입력 x에 대해 동일한 출력 y를 생성할 때, 두 네트워크의 입력 특성에 대한 '기여도'는 동일해야 한다.
   - 즉, 결과가 같다면 기여도 계산도 같아야 한다는 원칙이다.

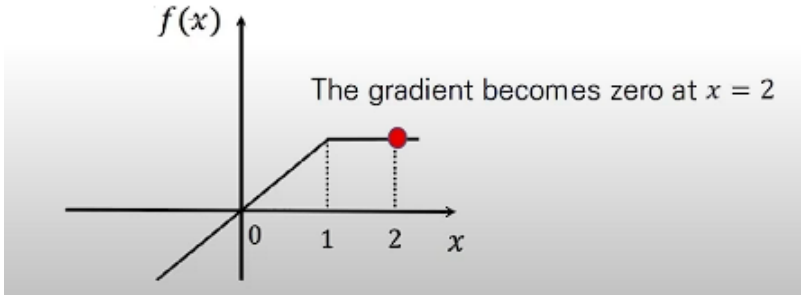


- 이전의 많은 methods들은 두 가지 공리를 동시에 만족하는 것이 없었다.

Ex01)

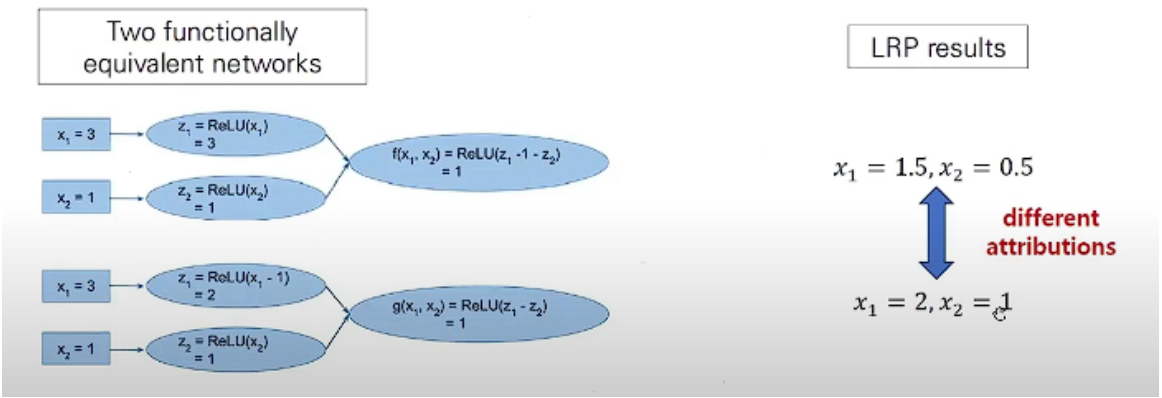
- f(2) = 1, f(0) = 0 → 출력값은 x=2의 영향으로 1만큼 변함
- 그러므로 x=2의 기여도는 0이 아니여야 하는데, gradient값이 0이다.
- 따라서 Sensitivity(a) 공리를 만족하지 못한다!

$$f(x) = 1 - \text{ReLU}(1 - x)$$



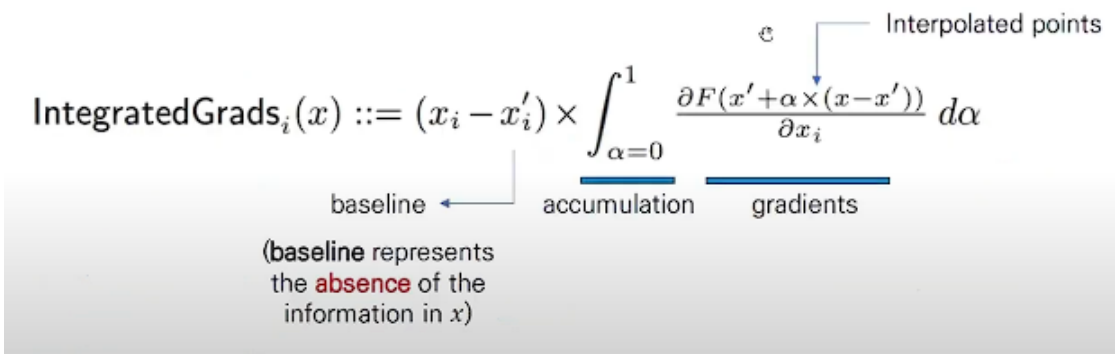
Ex02)

- LRP에서 다른 네트워크 구조를 갖지만, 동일 입력일 때 동일 출력을 생성하는 경우이다.
- 하지만, 다른 기여도가 반환된다. 이는 Implementation Invariance에 위배된다.



Main idea

- IG는 입력 특성이 모델 예측에 얼마나 기여했는지를 계산한다.
- Baseline(특성이 없는 상태)에서 입력값까지의 직선 경로를 따라 gradient를 적분하여 특성의 기여도를 누적한다.
- 이는 모델이 전체 입력 공간에서 어떻게 작동하는지를 반영한다.
- 아래 수식을 살펴보자.
  - x_i : 입력 특성 i
  - $x_i^{baseline}$: 기준값(특성 정보가 없는 상태)
  - F : 모델 함수(예측값을 반환)
  - α : 입력을 Baseline에서 현재 입력값으로 스케일링하는 파라미터

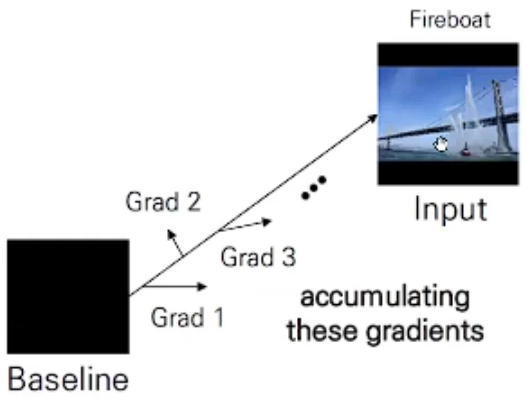


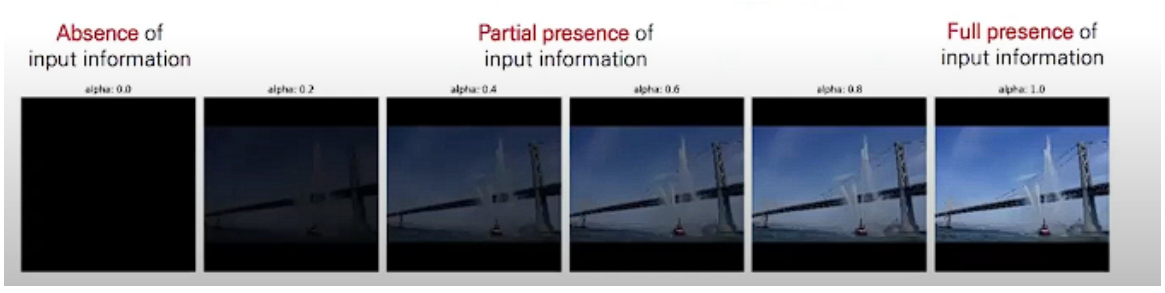
[기여도 계산 과정]

- IG는 입력값의 부분적 기여도(Local Contribution)를 Baseline에서 입력으로 가는 경로를 따라 누적한다.
- Baseline(Ex:완전히 검은 이미지)에서 시작하여 α값(0 → 1)을 변화시키며 각 지점에서의 기울기를 계산한다.

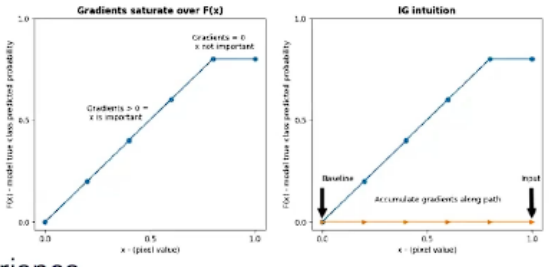
[시각적 예시]

- 아래 그림에서는 Baseline에서 시작하여 실제 input(완전한 이미지)으로 이동한다.
- 이 과정에서 gradient를 누적하며 각 픽셀의 기여도를 계산한다.
- 각 특성(픽셀)의 총 기여도는 이 gradient의 누적값으로 계산된다.



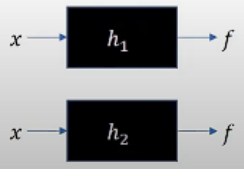


IG satisfies Sensitivity(a).



IG satisfies Implementation Invariance.

- **Chain rule** for gradients ensure Implementation Invariance.



$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial h_1}\frac{\partial h_1}{\partial x} = \frac{\partial f}{\partial h_2}\frac{\partial h_2}{\partial x}$$

Additionally, IG satisfies several other axioms. For more details, please refer to the original paper.