Saliency maps are popular for vision

Given model $f: \mathbb{R}^n \rightarrow \mathbb{R}$ and input $x$ in $\mathbb{R}^n$:

- $\alpha = \text{AttributionMethod}(f, x)$ in $\mathbb{R}^n$

Simple "interpretation": if $\alpha_i$ is big, then $x_i$ is important!

- Useful when the end-user may not be ML specialists

... but what does "important" mean exactly?
Feature attributions are "obvious" for simple models

Consider a linear model

\[ f(x) = c_0 + c_1x_1 + \cdots + c_nx_n \]

Clearly if the larger some \( c_i \), the more \( x_i \) will contribute to the score

- A natural feature attribution: \( \alpha_i = c_i \) (alternatively, \( \alpha_i = \text{abs}(c_i) \))

... but what about for a quadratic model?

\[ f(x) = c + b^\top x + x^\top Ax = c + \sum_{i=1}^{n} b_i x_i + \sum_{1 \leq i,j \leq n} A_{ij} x_i x_j \]

It's less clear what score each feature \( x_i \) should get
"Fundamental dilemma" of feature attributions

Pro: feature attributions are "nice"

- Easy to understand: number big = feature important
- There's a lot of attribution methods

Cons:

- What does "important" mean?
- There's a lot of attribution methods
  - "This feature has Shapley value XXX", okay, so what?
Idea: maybe we can measure the "quality" of FAs

If a feature is "important", then it should satisfy some properties.

- ... but what are these properties, and can we quantify them?

There is substantial work on developing metrics for feature attributions

- There's a lot, we'll talk about a few
Subtractive metrics

"If some feature is important, then removing it should decrease the score"

"Dog" (97%)

"Dog" (50%)

*I made up these numbers*
Additive metrics

"If a feature is important, then inserting it should increase the score"

"Dog" (<1%)

"Dog" (40%)
Example of other metrics

Perturbation:
- How sensitive is your metric to perturbations?

Compactness:
- Is your explanation too "big"? (e.g., for feature selection)

Connectedness:
- Are two candidate explanations "connected" in some sense?

What mathematical properties should we expect?

Given a model, an input, an explanation method, and some metric ...

... what formal (i.e., mathematical) properties should these things satisfy?

- e.g., "does the top-k% of features from this method guarantee a score decrease of q% wrt some metric, model class, and input family?"

In general? Hard to prove such statements

- Neural networks are magic
What can we do from here?

Our work: under some conditions, one CAN guarantee formal properties

Special case: binary-valued feature attributions (i.e., feature selection)

Input $x \in \mathbb{R}^n$

Attr $\alpha \in \{0,1\}^n$
I have an attribution, but how do I "evaluate" it?

In vision: we can use the original model

Classifier f

"Dog" (96%)

Classifier f

??? (ideally: "Dog")
What do we NOT want to happen?

"Dog"

"Dog"

"Cat"

This is usually **NOT** desirable

Intuition: the original feature selection you gave me is not "convincing"

*I made up this example, but it can happen. Trust me, bro!*
Stability as a "desirable" property

Selected by your favorite attribution method

Stability: any superset of features induces the same prediction

\[ f(x^\circ \alpha) \equiv f(x^\circ \alpha') \text{ for all } \alpha \leq \alpha', \text{ where } \alpha = \text{BinaryAttribution}(f,x) \]
How can we achieve/guarantee stability!

You probably can't! (For Real Models™)

- There's $O(2^n)$ different $\alpha' \geq \alpha$ binary vectors to check

But we can maybe go for local approximations (Incremental Stability)

Given explanation

"Dog"

+ a few additional features

"Dog"
The plan

1. Incremental stability via Lipschitz smoothness

2. Achieve incremental stability with multiplicative smoothing (MUS)

3. We can check if $f$ is incrementally stable at some $x$ in $O(1)$ time.
Step 1: incremental stability

\[ f(\mathbf{x}) \equiv f(\mathbf{x} + \Delta) \quad \text{for all small } \Delta \]

Sufficient condition: Lipschitz wrt masking of features
- L1 norm on binary vectors = number of differences

\[ f(\mathbf{x}) - f(\mathbf{y}) \leq \lambda \|\mathbf{x} - \mathbf{y}\|_1 \quad \text{for all } \mathbf{x}, \mathbf{y} \]

Definition (Lipschitz wrt Feature Maskings). The function \( f: \mathbb{R}^n \rightarrow [0,1] \) is \( \lambda \)-Lipschitz wrt the masking of features at \( x \) in \( \mathbb{R}^n \) if:

\[ f(x^\circ \alpha) - f(x^\circ \alpha') \leq \lambda \|\alpha - \alpha'\|_1 \quad \text{for all } \alpha, \alpha' \text{ in } \{0,1\}^n \]
Step 2: Multiplicative Smoothing (MuS)

"base classifier" $h$ \(\xrightarrow{\text{MuS}}\) $\lambda$-Lipschitz-smooth $f$

$$f(x) = \text{MuS}(h, x) = \text{avg}(h(x^{(1)}), ..., h(x^{(N)}))$$

Sample $s^{(1)} ... s^{(N)}$

每个 $s^{(i)}$ \(\sim\text{Bern}(\lambda)\)

Mask $x$

$x^{(1)} ... x^{(N)}$ where each $x^{(i)} = x \odot s^{(i)}$

$h(x^{(1)}) ... h(x^{(N)})$
Step 2: The Math

Recall $f(x) = \text{MuS}(h, x)$ and let

$$g(x, \alpha) = \text{MuS}(h, x \cdot \alpha) = \text{avg}(h(x \cdot \alpha \cdot s^{(1)}), \ldots, h(x \cdot \alpha \cdot s^{(N)}))$$

**Theorem (MuS).** Let $D$ be any distribution on $\{0,1\}^n$ where each coordinate of $s \sim D$ is marginally distributed as $s_i \sim \text{Bern}(\lambda)$ and let

$$g(x, \alpha) = E_{s \sim D} h(x \cdot \alpha \cdot s), \quad \text{for any } h: \mathbb{R}^n \to [0,1],$$

then $g(x, \alpha)$ is $\lambda$-Lipschitz in $\alpha$ wrt the $L^1$ norm for all $x$.

Note: Lipschitz smoothness is a property of functions $D$ need NOT be coordinate-wise independent

- We just requires that each sample's coordinate marginally $\sim \text{Bern}(\lambda)$
- Allows for a deterministic evaluation with $N \ll 2^n$ samples
  - Recall that $\text{Bern}^n(\lambda)$ has $2^n$ unique values
Step 3: provable incremental stability

Suppose $h: \mathbb{R}^n \rightarrow [0,1]^m$ is a classifier
Let $f(x) = \text{MuS}(h, x)$ with parameter $\lambda$
Let $\alpha = \text{BinaryAttribution}(f, x)$, such that $f(x) \equiv f(x \circ \alpha)$

**Theorem (MuS).** Suppose that

$$\text{Class1Prob}(f(x \circ \alpha)) - \text{Class2Prob}(f(x \circ \alpha)) \geq 2\lambda r,$$

then for any $\alpha' \geq \alpha$ with $||\alpha' - \alpha||_1 \leq r$, we have $f(x \circ \alpha') \equiv f(x \circ \alpha)$. 
Basic summary of MuS

Step 1: stability is hard, so we go for incremental stability

- Key idea: Lipschitz constants

Step 2: "randomized" smoothing

- $f(x) = \text{MuS}(h, x) = \text{avg}(h(x \circ s^{(1)}), ..., h(x \circ s^{(N)}))$
- $f(x \circ \alpha) = \text{MuS}(h, x \circ \alpha) = \text{avg}(h(x \circ \alpha \circ s^{(1)}), ..., h(x \circ \alpha \circ s^{(N)}))$

Step 3: Lipschitz constants $\rightarrow$ stability guarantees
Experimental evaluations

E1: how good are the stability guarantees?

- How much incremental stability radius can we achieve?
  - for $x$ in dataset with $\alpha = \text{BinaryAttribution}(f, x)$:
    - $r = \frac{\text{Class1Prob}(f(x\circ\alpha)) - \text{Class2Prob}(f(x\circ\alpha))}{2\lambda}$

E2: what is the cost of smoothing?

- Smoothing inherently requires us to inject noise
- Accuracy degradation of $f(x) = \text{MuS}(h, x)$
E1: radius of incremental stabilities

Base classifier: $h = \text{Vision Transformer}$

Binary Attribution: SHAP (top-25%)

Dataset: $N = 2000$ samples from ImageNet
E2: accuracy penalty of smoothing

Vision Dataset: ImageNet1K (N = 2000 samples)
Language Dataset: TweetEval (N = 2000 samples)
Takeaways

1. We give a way to provably check for incremental stability
2. MuS: randomly drops features to these guarantees
   a. MuS(h, x) = avg(h(x ◦ s^{(1)}), ..., h(x ◦ s^{(N)}))
   b. g(x, α) = f(x ◦ α) = MuS(h, x ◦ α)
   c. g(x, α) is λ-Lipschitz in α wrt the L^1 norm
3. Lipschitz smooth gives lower-bound on the incremental stability radius

Efficient Smoothing

Main challenge: MuS is defined in terms of an expected value

- $\text{Bern}^n(\lambda)$ has $N=2^n$ unique values (too many for the expected value!)
- MuS only requires that each coordinate is $\sim\text{Bern}(\lambda)$
  - Do NOT need coordinate-wise independence
  - Algorithm below: $N = q$ unique values
  - Main idea: use $v$ as a pseudo-RNG seed, with 1-dim "randomness" $s_{\text{base}}$

**Proposition 3.4.** Fix integer $q > 1$ and consider any vector $v \in \{0, 1/q, \ldots, (q-1)/q\}^n$ and scalar $\lambda \in \{1/q, \ldots, q/q\}$. Define $s \sim \mathcal{L}_{qv}(\lambda)$ to be a random vector in $\{0, 1\}^n$ with coordinates given by

$$s_i = \mathbb{I}[t_i \leq \lambda], \quad t_i = v_i + s_{\text{base}} \mod 1, \quad s_{\text{base}} \sim \mathcal{U}(\{1/q, \ldots, q/q\}) - 1/(2q).$$

Then there are $q$ distinct values of $s$ and each coordinate is distributed as $s_i \sim \mathcal{B}(\lambda)$.

**Proof.** First, observe that each of the $q$ distinct values of $s_{\text{base}}$ defines a unique value of $s$ since we have assumed $v$ and $\lambda$ to be fixed. Next, observe that each $t_i$ has $q$ unique values uniformly distributed as $t_i \sim \mathcal{U}(1/q, \ldots, q/q) - 1/(2q)$. Because $\lambda \in \{1/q, \ldots, q/q\}$ we therefore have $\Pr[t_i \leq \lambda] = \lambda$, which implies that $s_i \sim \mathcal{B}(\lambda)$. \qed