Local Interpretation and LIME

David S. Rosenberg

NYU: CDS

April 28, 2021

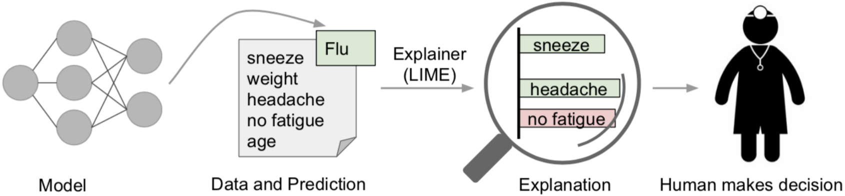
Contents

Introduction

Local Feature Importance

- I want to know ***why*** a particular prediction was made.
- Don't care that x_1(say age) is generally important to the prediction function.
- I want to know why $f(x) = \text{DENY LOAN}$.
- e.g. Perhaps when $x_2 = \text{Bad Credit}$, x_1 is irrelevant.
- If my loan was denied, I may also want to know
  - what can I change to improve my score, according to your function f?
- The methods we talk about today are about local feature importance
  - But they can be aggregated back to global importance scores

Idea of local interpretability



- Model makes prediction for diagnosis given observations.
- "Explanation" highlights symptoms that provide most evidence for and against the prediction.
- Helps human decide whether to trust prediction.
- By looking at many such explanations, can we build trust in the model?

Figure 1 from [RSG16b].

What does this buy us?

- Helps detect leakage (i.e. information leakage from label to input)
- Helps identify domain shift (e.g. feature useful in training that won't be useful in deployment)
- Could help inform model selection
  - e.g. two models perform equally well on validation metrics
  - one model uses features that are more intuitive to us

LIME

"Why Should I Trust You?" Explaining the Predictions of Any Classifier

Marco Tulio Ribeiro
University of Washington
Seattle, WA 98105, USA
marcotcr@cs.uw.edu

Sameer Singh
University of Washington
Seattle, WA 98105, USA
sameer@cs.uw.edu

Carlos Guestrin
University of Washington
Seattle, WA 98105, USA
guestrin@cs.uw.edu

- This is the paper that introduced LIME [RSG16b].
- A popular **local** interpretation method.
- In other words, explain the prediction for a particular instance $x \in \mathcal{X}$.
- The paper is actually a bit difficult to understand, but we'll unpack it here.

Will the real LIME please stand up?

- There was the original LIME paper [RSG16b].
  - Presented a general approach for local interpretation...
  - But only really fleshed out the method for image and text inputs.
  - My thought: when it works, seems useful!
- The associated GitHub repo later added support for tabular data.
  - My thought: not as natural or compelling as for images / text
- Explanatory blogs and books (e.g. [Mol19, Tha21])
  - often introduce LIME for the tabular case, and give reasonable critique,
  - but not obvious the same critiques are as relevant for images and text

LIME: basic idea

- Given: prediction function f
- Given: instance $x \in \mathcal{X}$ to be explained
- Build "interpretable" / "white box" model $\hat{f}$ that approximates f near x.
- Explain prediction $f(x)$ by interpreting $\hat{f}$.
- How this is implemented varies by input domain.

What's interpretable, really?

- Some standard "white box" / interpretable models are
  - linear models
  - tree models
  - generalized additive models
- But for these to be interpretable, the features must be interpretable.
  - No word embeddings.
  - No pixels.
  - No complicated derived features or principal components.
  - etc...

Interpretable data representations

- [RSG16b] presents the idea of
  - "interpretable data representations"
- We'll also call them interpretable features.
- For interpretable features, they recommend using binary features.
- If $x \in \mathbb{R}^d$ is the original representation of some instance,
  - we'll use $x' \in \{0,1\}^{d'}$ as the **interpretable representation**.
- This interpretable representation is only intended to represent examples
  - that are "local" to x in some sense.
  - Or are "perturbations" of x.
- In particular, the instance x is usually represented as $(1,\ldots,1)^{d'}$
  - i.e. all 1's.

An explanation for $f(x)$ at x

- [RSG16b] defines an **explanation** as
  - a model $g \in G$, where G is a class of interpretable models.
- Important: the explanation model g is **specific to** an instance x.
- The domain of g is $\{0,1\}^{d'}$.
- That is, g acts on the presence or absence of "interpretable components."
- Let's jump to an example...

Example: LIME for vision

LIME: for vision

- Suppose we have a trained model that predicts $\mathbb{P}(\text{frog})$.
- We want to understand the prediction for the following:

Original Image

Interpretable Components

Image from [RSG16a].

LIME: Main idea

- Which part of the image / "interpretable component" was most important for prediction?
- One strategy: cover up various parts of the image, and see effect on $\mathbb{P}(\text{frog})$.
  - cover up = replace with solid background color
- LIME strategy
  - Use 0/1 indicator vector to indicate which parts of image are "on"
  - Train [weighted, sparse] linear regression to predict the [logit of] probability
  - feature with largest positive weight is the "explanation"

LIME: For vision

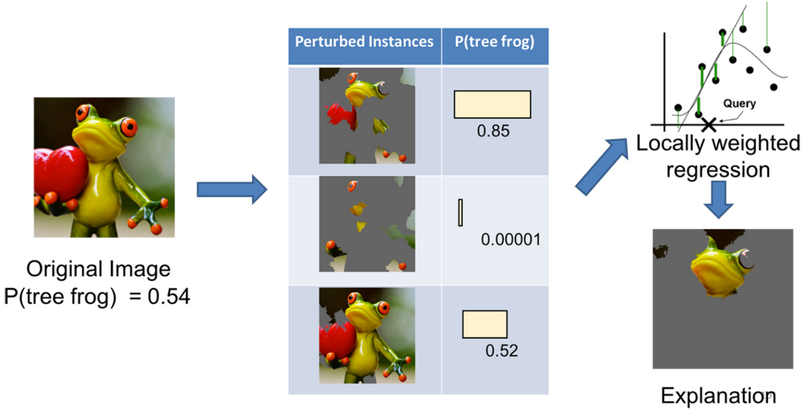


Image from [RSG16a].

LIME: For vision

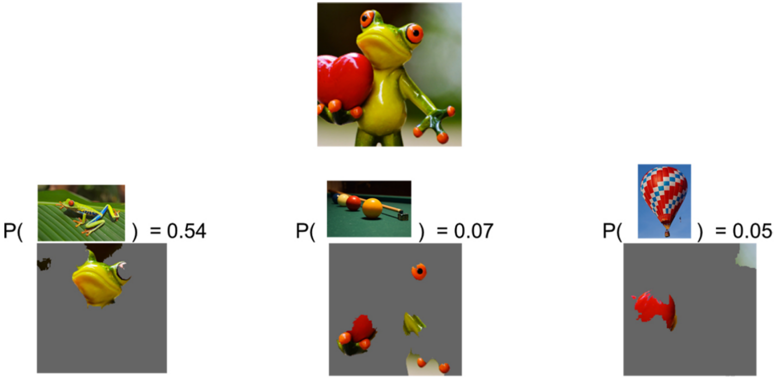
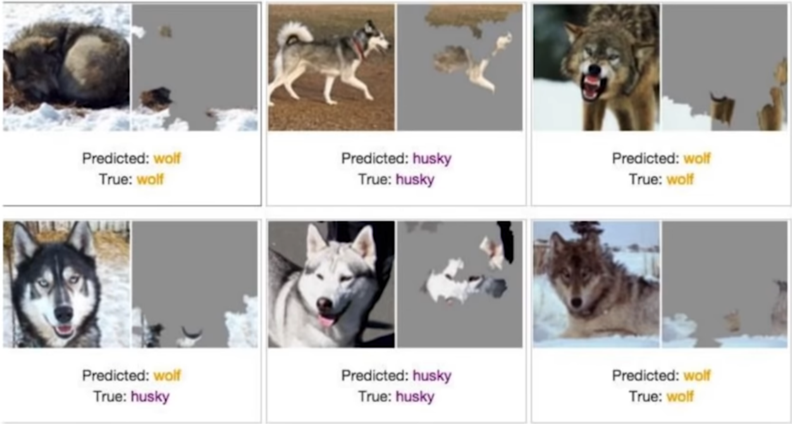


Figure 6. Explanation for a prediction from Inception. The top three predicted classes are "tree frog," "pool table," and "balloon." Sources: Marco Tulio Ribeiro, Pixabay (frog, billiards, hot air balloon).

Image from [RSG16a].

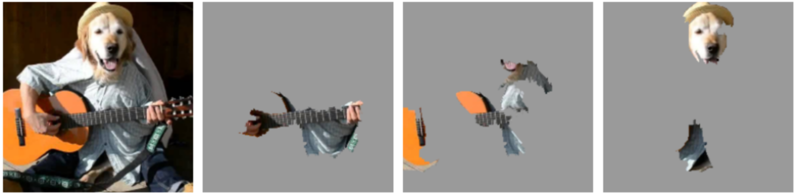
LIME: For finding generalization issues



Husky vs wolf? or "snow detector" attribution.

From [Kul17] slide 32.

Interpretation for dog playing guitar



(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Figure 4: Explaining an image classification prediction made by Google's Inception neural network. The top 3 classes predicted are "Electric Guitar" ($p = 0.32$), "Acoustic guitar" ($p = 0.24$) and "Labrador" ($p = 0.21$)

Figure 4 from [RSG16b].

LIME (generic paper version)

The interpretable representation

- Generate **interpretable representation**.
- That is, create a mapping $h_x : \{0,1\}^{d'} \to \mathbb{R}^d$ such that
  - $h_x((1,1,\ldots,1)) = x$.
  - for any $x' \in \{0,1\}^{d'}$, $h_x(x')$ is some "perturbation" of x.
- For example
  - $h(x')$ is an image with regions blocked out.
  - $h(x')$ is a sentence with words eliminated.

Sample perturbations

- In [RSG16b] they say they sample as follows:
  - Sample k uniformly at random from $1, \ldots, d'$.
  - Then randomly choose k interpretable components to keep.
  - So if we choose $k = 3$ and $d' = 5$, we may end up with

$$x' = (1, 1, 0, 1, 0).$$

- Though in the LIME codebase,
  - they just keep or remove each component with equal probability,
  - which is quite different.
- In either case, $h_x(x') \in \mathbb{R}^d$ will be some perturbed version of x in the original space.

Proximity measure

- Introduce proximity measure $\pi_x(z)$.
- Measure how "similar" z and x are.
- Both z and x are in the original feature space $\mathbb{R}^d$.
- For perturbed sample $x' \in \{0,1\}^{d'}$,
  - we create its image in feature space $h(x') \in \mathbb{R}^d$
  - then compute its "similarity" to x with $\pi_x(h(x'))$.
- In [RSG16b] they use a standard RBF kernel

$$\pi_x(z) = \exp\left(-D(x,z)^2/\sigma^2\right),$$

with any distance function D and $\sigma > 0$.
  - e.g. D can be Euclidean distance or cosine distance.

Fidelity measure

- The idea is for g to be a good approximation of f around x.
- How can we assess that?
- Let $\mathcal{D}'_x \subset \{0,1\}^{d'}$ be a set of perturbations of x.
- They use a square-loss fidelity measure:

$$\mathcal{L}(f,g,\pi_x,\mathcal{D}'_x) = \sum_{x' \in \mathcal{D}'_x} \pi_x(h(x')) \left(f(h(x')) - g(x')\right)^2.$$

- In words: g's predictions on $\mathcal{D}'_x$ are close to
  - f's predictions on the feature-space versions of elements of $\mathcal{D}'_x$,
  - weighted by the similarity between the perturbations and x.

Fitting g

- They take G to be the class of linear models

$$g(x') = w_g^T x'$$

  for $x' \in \{0,1\}^{d'}$ and $w_g \in \mathbb{R}^{d'}$.
- Create perturbation set $\mathcal{D}'_x \subset \{0,1\}^{d'}$.
- For each $x' \in \mathcal{D}'_x$ we get a training example $(x', f(h(x')))$ with weight $\pi_x(h(x'))$.
- Fit g with this weighted training data so that w_g is sparse.
  - e.g. use Lasso to get at most K nonzero components of w_g.
- Resulting weights w_g can be "interpreted".

LIME: tabular version (from code)

Tabular case

- [RSG16b] doesn't discuss the tabular case.
- But the LIME codebase supports it.
- What changes?
  - perturbations
  - the interpretable representation
- Perturbations are done separately for each column/feature.
- At a high level, we resample each feature according to
  - some approximation of its marginal distribution.

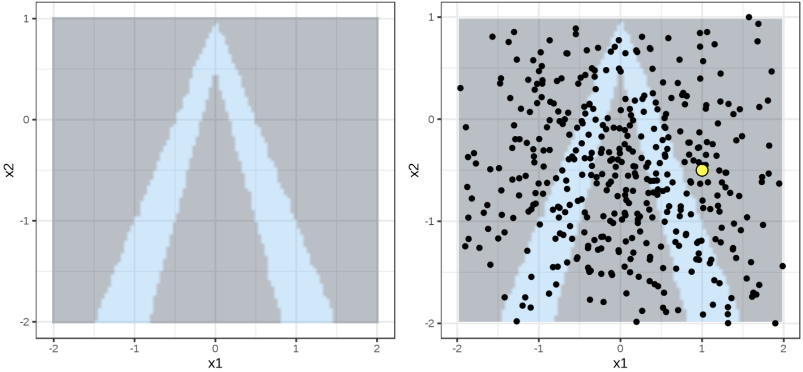
Tabular case: perturbations for categorical variables

- For a categorical column,
    - we get the relative frequencies of the categories in a training set.
- Column is perturbed by resampling according to those frequencies.
- The "interpretable representation" is a binary indicator
    - 1 if resampled column is same as original value of point we're trying to interpret
    - 0 otherwise

Tabular case: perturbations for continuous variables

- For a continuous column,
  - we compute the mean μ and SD σ of the column.
- We perturb by sampling a Gaussian either from $\mathcal{N}\left(\mu, \sigma^2\right)$.
  - This doesn't actually feel like a perturbation, since it has nothing to do with x.
- OR we sample from from $\mathcal{N}\left(x, \sigma^2\right)$,
  - where x is the value of the variable for the instance of point we're interpreting,
  - which feels more like a perturbation.
- If the variable is sparse, we only perturb the nonzero values.

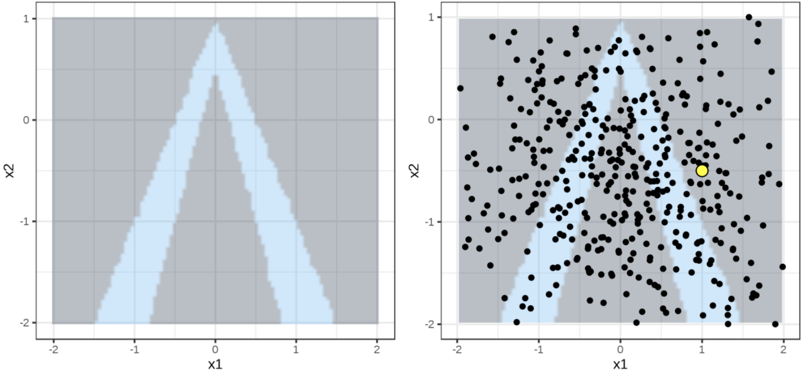
For tabular data (continuous case)



- Left: color indicates prediction of model on input space (x_1, x_2).
- Right: black dots are the "perturbed" versions of the yellow instance

Figure 5.33(A,B) from [Mol19, Sec 5.7.1].

Basic idea of building interpretable model (I)



- Left: color indicates prediction of model on input space (x_1, x_2).
- Right: black dots are a sample from input space; yellow dot is x_0, instance to be explained

Figure 5.33(A,B) from [Mol19, Sec 5.7.1].

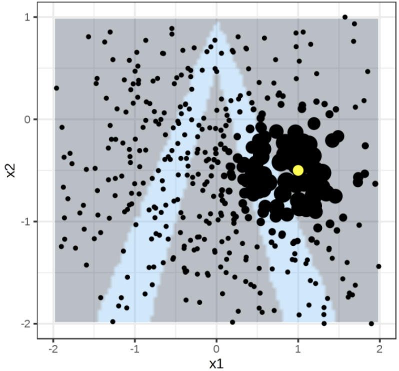
Getting a sample

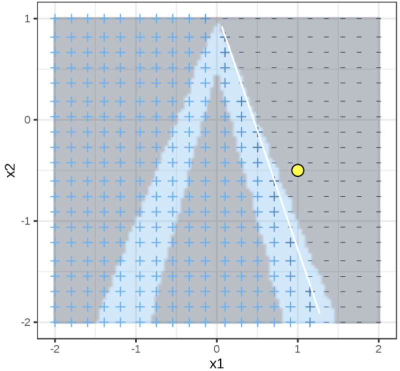
- In LIME, we're only given the prediction function f.
- If we were also given access to the data-generating distribution,
  - perhaps we could lus e

Weight samples by proximity to instance

- Somehow we get a sample from the input space (we'll come back to this).
- We want to build a model that's "local" to the yellow instance of interest.
- Let $\pi_{x_0}(x)$ be a weighting function
  - gives higher weight to points x that are "closer" to x_0.
- [RSG16b] calls π_{x_0} a **proximity measure** to define "locality" around x_0.
- We then do a weighted fit for our interpretable model to the sample points.

Basic idea of building interpretable model (II)





- Left: weights of sample points are shown
- Right: class predictions of linear model fit to weighted points – white line is boundary

Figure 5.33(C,D) from [Mol19, Sec 5.7.1].

References

Resources

- Christoph Molnar's online book, continually updating, is a great source for interpretable machine learning [Mol19]. Though currently (25 April 2021), the explanation for LIME glosses over a lot of details.
- Authors of the LIME paper have a nice blog post [RSG16a]

References I

[Kul17] Kasia Kulma, *Interpretable machine learning using lime framework*, https://www.slideshare.net/0xdata/interpretable-machine-learning-using-lime-framework-kasia-kulma-phd-da 2017, [Online; accessed 15-June-2021].

[Mol19] Christoph Molnar, *Interpretable machine learning*, bookdown.org, 2019, https://christophm.github.io/interpretable-ml-book/.

[RSG16a] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin, *Local interpretable model-agnostic explanations (lime): An introduction*, Aug 2016, https://www.oreilly.com/content/introduction-to-local-interpretable-model-agnostic-explanations-lime/.

References II

[RSG16b] ———, *"why should i trust you?": Explaining the predictions of any classifier*, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (New York, NY, USA), KDD '16, Association for Computing Machinery, 2016, pp. 1135–1144.

[Tha21] Ajay Thampi, *Interpretable ai*, Manning Publications, 2021, https://www.manning.com/books/interpretable-ai.