BayLIME: Bayesian local interpretable model-agnostic explanations

Xingyu Zhao, Wei Huang, Xiaowei Huang, Valentin Robu, David Flynn, 2021

Motivation: take LIME algorithm and improve it, specifically:

- explanation consistency (across different runs)
- robustness to hyperparameters (kernel settings)
- fidelity of explanations

LIME recap

LIME Algorithm

Explanations can be calculated with a following instructions.

1. Let $x' = h(x)$ be a version of x in the interpretable data space
2. for i in 1…N {
3. z'[i] = `sample_around`(x')
4. y'[i] = f(z'[i])
5. w'[i] = `similarity`(x', z'[i])
6. }
7. return `K-LASSO`(y', x', w')

where

- x – an observation to be explained
- N – sample size needed to fit a glass-box model
- K – complexity, the maximum number of variables in the glass-box model
- `similarity` – a distance function in the original data space
- `K-LASSO` – a weighted LASSO linear-regression model that selects K variables
- w' – weights that measure of the similarity between original observation x and new artificially

source: https://github.com/mim-uw/eXplainableMachineLearning-2023/blob/main/Lectures/03_lime.html#/lime-algorithm

LIME problem

Explanations may be inconsistent and imprecise for small sample size n.
Computation time is linear w.r.t. n.

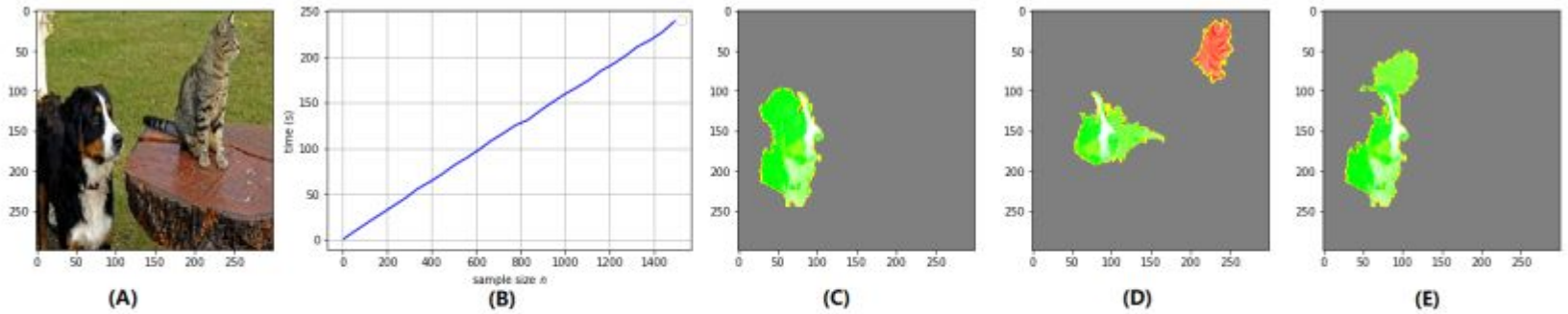
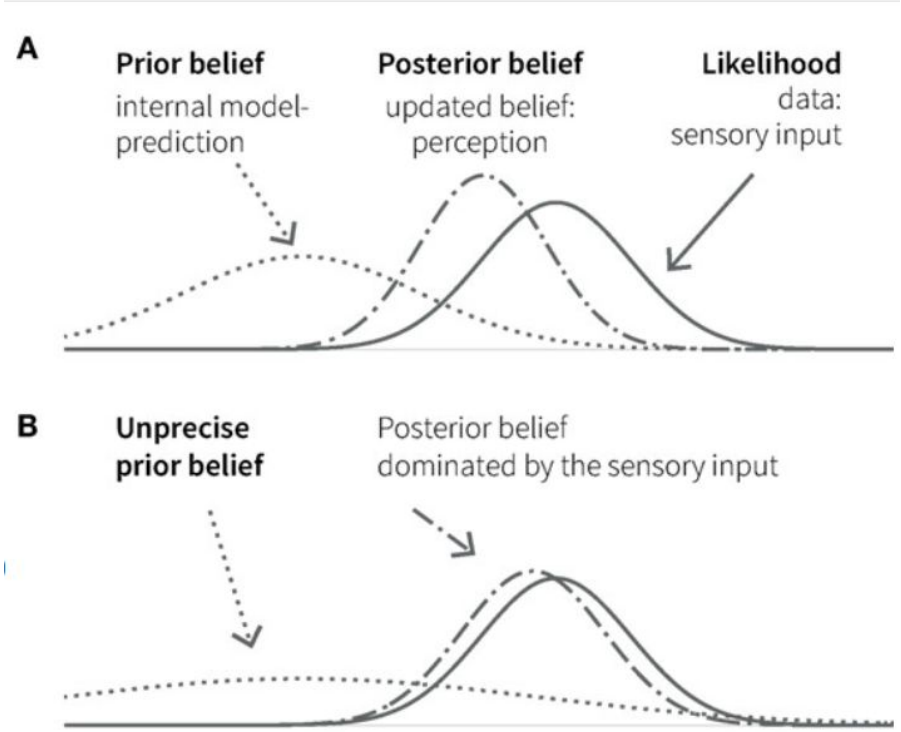


Figure 1: (A) The instance with prediction "Bernese mountain dog" under explanation; (B) LIME computational time as a function of perturbed sample size n; (C)-(E) Three repeated and noticeably *inconsistent* LIME explanations when $n = 100$, showing the top 4 features contributing toward and against the prediction (shaded green and red respectively).

BayLIME main idea



https://www.researchgate.net/figure/Principles-of-Bayesian-inference-A-A-prior-belief-knowledge-expectation-or_fig2_304033579

Let’s try to mirror Bayesian inference improvement upon the frequentist approach and use some belief on how more or less the explanations should look like (using some other, fast algorithm).

Bayesian details

$$Pr(y_i \mid \boldsymbol{\beta}, \boldsymbol{x_i}, \alpha) = \mathcal{N}(y_i \mid \boldsymbol{x_i}\boldsymbol{\beta}, \alpha^{-1}) \quad (3)$$

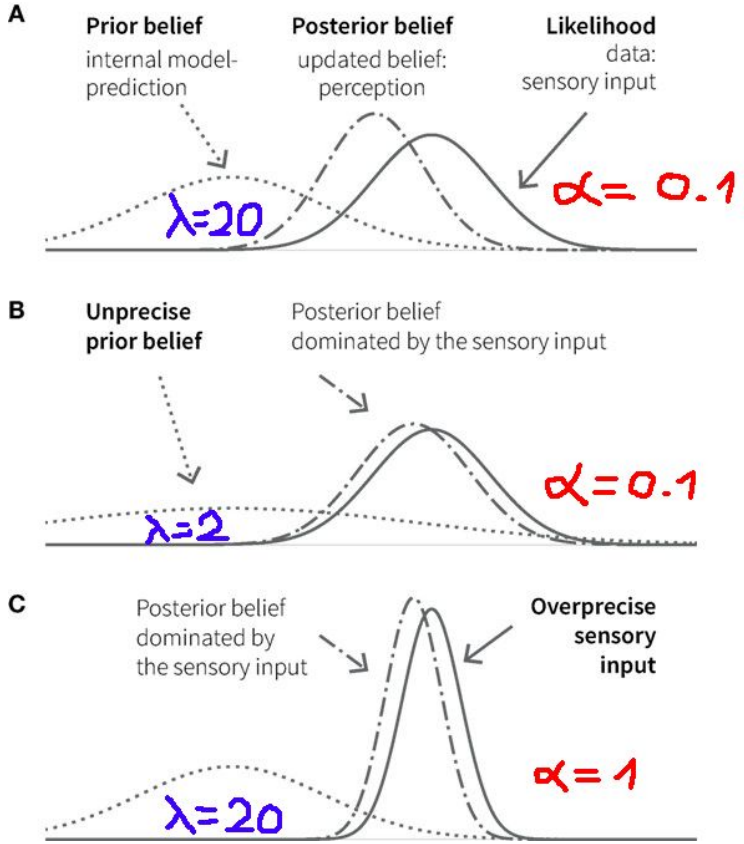
where $\boldsymbol{\mu_0}$ and $\boldsymbol{S_0}$ are the prior mean vector and covariance matrix, respectively. Then, thanks to the conjugacy, the posterior $\boldsymbol{\beta}$ is also a Gaussian:

$$\begin{aligned} Pr(\boldsymbol{\beta} \mid \boldsymbol{y}, \boldsymbol{X}, \alpha, \boldsymbol{\mu_0}, \boldsymbol{S_0}) &\propto Pr(\boldsymbol{y} \mid \boldsymbol{\beta}, \boldsymbol{X}, \alpha) Pr(\boldsymbol{\beta} \mid \boldsymbol{\mu_0}, \boldsymbol{S_0}) \\ &= \mathcal{N}(\boldsymbol{\beta} \mid \boldsymbol{\mu_n}, \boldsymbol{S_n}) \quad (6) \end{aligned}$$

with the mean vector $\boldsymbol{\mu_n}$ and covariance matrix $\boldsymbol{S_n}$:

$$\boldsymbol{\mu_n} = \boldsymbol{S_n}(\boldsymbol{S_0}^{-1}\boldsymbol{\mu_0} + \alpha \boldsymbol{X}^T \boldsymbol{y}), \;\; \boldsymbol{S_n}^{-1} = \boldsymbol{S_0}^{-1} + \alpha \boldsymbol{X}^T \boldsymbol{X} \quad (7)$$

For simplicity and without loss of generality, we assume the Gaussian prior is governed by a single precision parameter λ, i.e. $\boldsymbol{S_0} = \lambda^{-1}\boldsymbol{I}_m$ where $\boldsymbol{I}_m$ is a $m \times m$ identity matrix.



Inference interpretation

To see the above insight clearer, we present the special case of a single feature instance ($m = 1$) with a simplified kernel function that returns a constant weight w_c (i.e. $w_i = w_c, \forall i = 1, \ldots, n$), then μ_n ($\boldsymbol{\mu_n}$ with 1 feature) becomes:

$$\frac{\lambda}{\lambda + \alpha w_c \sum_{i=1}^{n} x_i^2} \mu_0 + \frac{\alpha w_c \sum_{i=1}^{n} x_i^2}{\lambda + \alpha w_c \sum_{i=1}^{n} x_i^2} \beta_{MLE} \quad (10)$$

For numerical features of a tabular dataset, x_i are random samples from a standard Gaussian $\mathcal{N}(0, 1)$. Thus, say, for numerical features from tabular data, $\sum_{i=1}^{n} x_i^2 \approx n(1 + 0^2) = n$ via Eq. (11). Then, Eq. (10) can be simplified as:

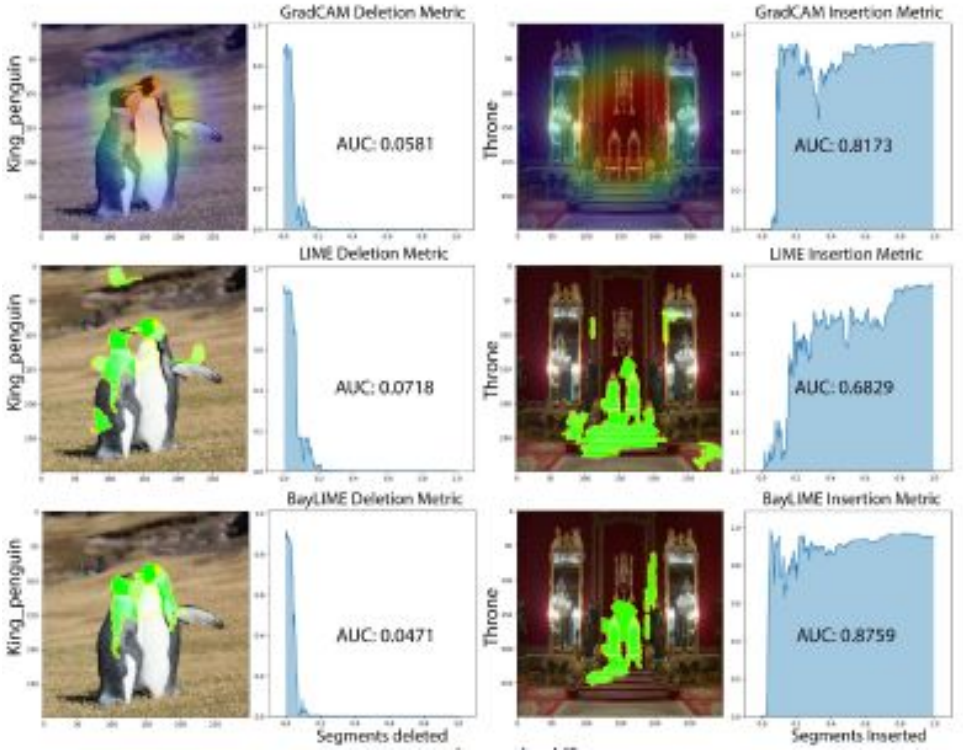
$$\frac{\lambda}{\lambda + \alpha w_c n} \mu_0 + \frac{\alpha w_c n}{\lambda + \alpha w_c n} \beta_{MLE} \quad (12)$$

With a couple of assumptions we can end up with a posterior which is a hyper-parameter and sample size weighted average of prior explanation and data-originated one which is nice to have to grasp a feeling of this mechanism.

How to obtain prior knowledge?

Short and correct answer is anyhow. The only requirement is that you need to extract your LIME-specific feature importances from obtained explanation.

here GradCAM is used as a prior for BayLIME - - - ->



Explanation consistency evaluation

Create k explanations and quantify their agreement. Technically one can use adapted Kendall's W metric which for a set of feature ratings outputs their agreement (from 0 to 1).

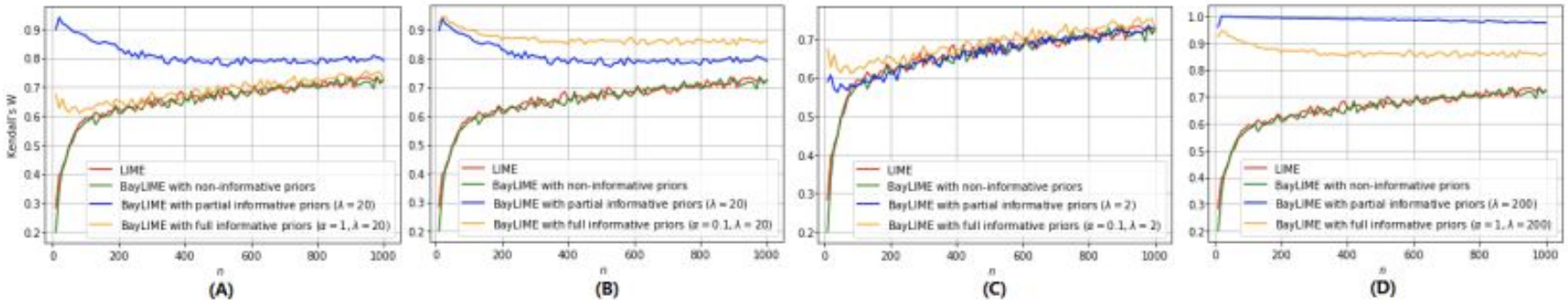


Figure 2: Kendall's W in $k = 200$ repeated explanations by LIME/BayLIME on random Boston house-price instances. Each set shows an illustrative combination of α and λ. Same patterns are observed on images, cf. Appendix B for more.

Robustness evaluation

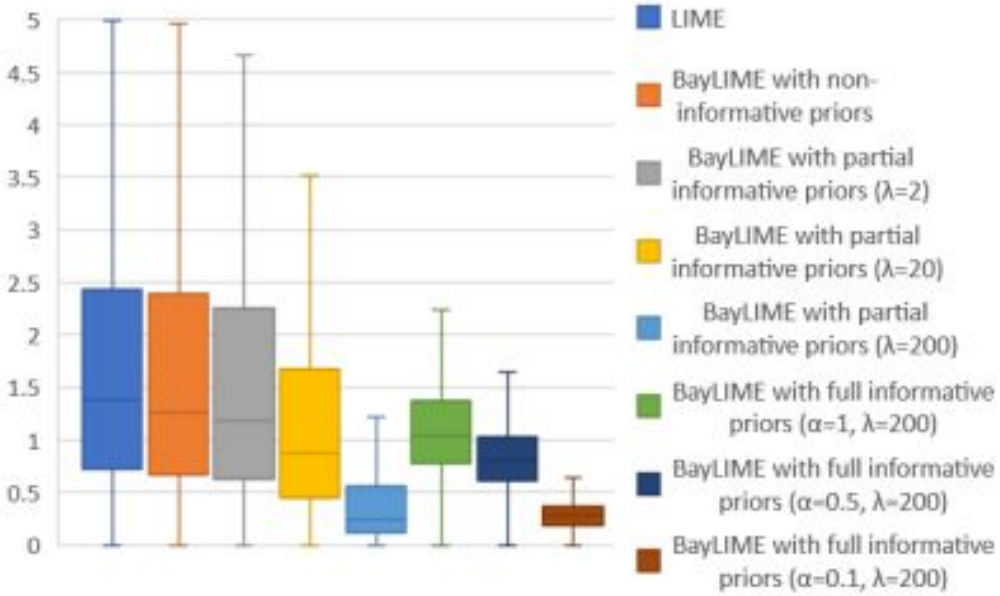
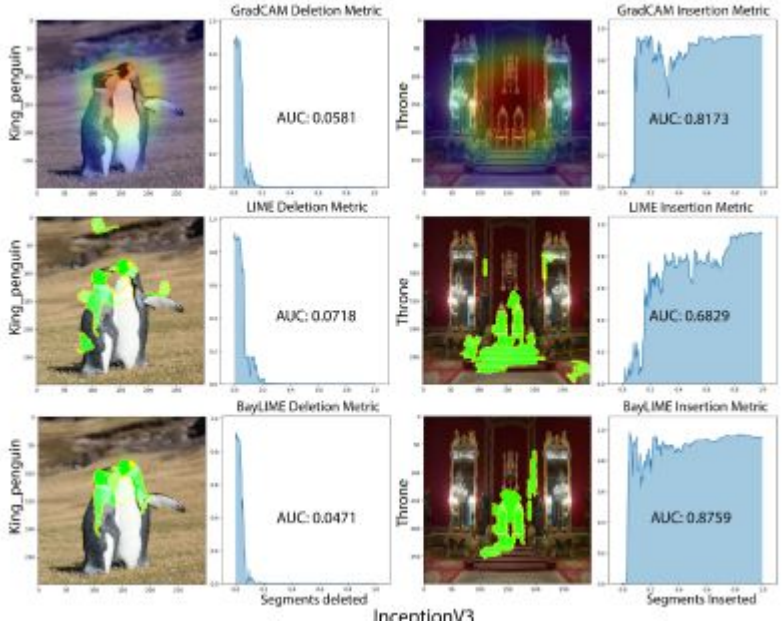


Figure 3: The general (un-)robustness of eight AI explainers to kernel settings (box-and-whisker plots without outliers).

Sample reasonable* kernel widths l1, l2, compute distance in explanations obtained using these divided by ||l1 - l2||. Having that report the median of this set as a kernel robustness loss of a model.

Quality evaluation

Measure the drop/uplift of a proper class probability when removing/adding top features in their importance order.



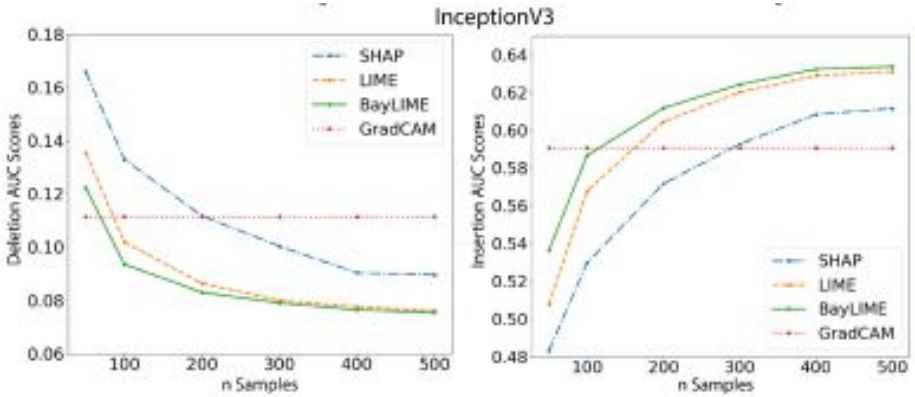
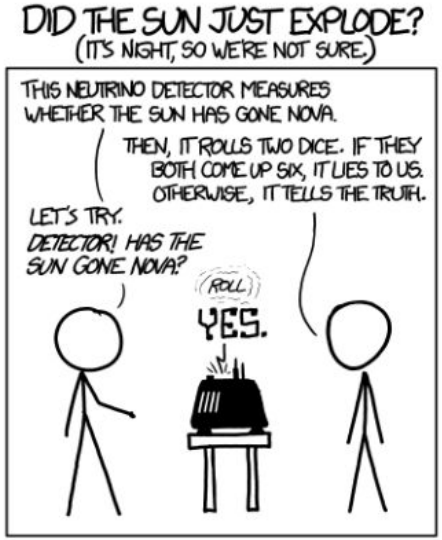
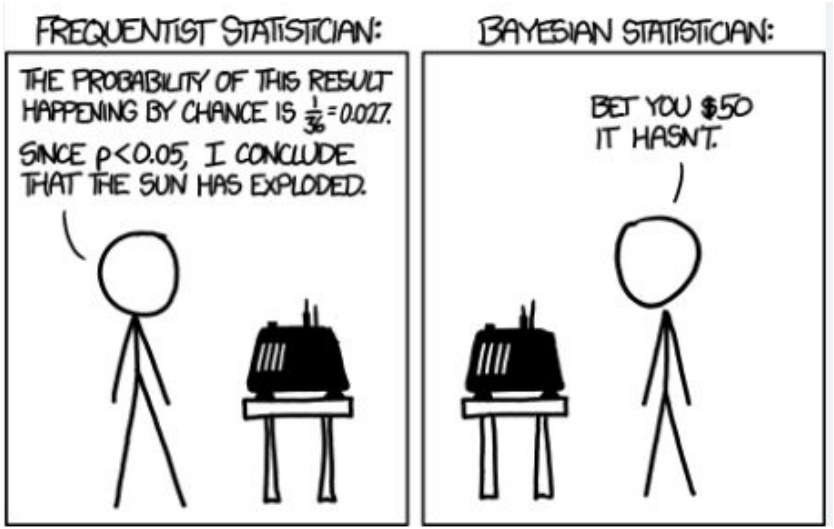


Figure 4: Two sets of examples ($n = 300$), and the average AUC (based on 1000 images per value of n) of the deletion (smaller is better) and insertion (bigger is better) metrics of BayLIME comparing with other XAI methods.

BayLIME key take-away

Use LIME with some prior knowledge to gain on robustness, fidelity and inference speed.





https://xkcd.com/1132/