Local Surrogate (LIME)

Local surrogate models are interpretable models that are used to explain individual predictions of black box machine learning models. Local interpretable model-agnostic explanations (LIME)[79] is a paper in which the authors propose a concrete implementation of local surrogate models. Surrogate models are trained to approximate the predictions of the underlying black box model. Instead of training a global surrogate model, LIME focuses on training local surrogate models to explain individual predictions.

The idea is quite intuitive. First, forget about the training data and imagine you only have the black box model where you can input data points and get the predictions of the model. You can probe the box as often as you want. Your goal is to understand why the machine learning model made a certain prediction. LIME tests what happens to the predictions when you give variations of your data into the machine learning model. LIME generates a new dataset consisting of permuted samples and the corresponding predictions of the black box model. On this new dataset LIME then trains an interpretable model, which is weighted by the proximity of the sampled instances to the instance of interest. The interpretable model can be anything from the interpretable models chapter, for example Lasso or a decision tree. The learned model should be a good approximation of the machine learning model predictions locally, but it does not have to be a good global approximation. This kind of accuracy is also called local fidelity.

Mathematically, local surrogate models with interpretability constraint can be expressed as follows:

$$\text{explanation}(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g)$$

The explanation model for instance x is the model g (e.g. linear regression model) that minimizes loss L (e.g. mean squared error), which measures how close the explanation is to the prediction of the original model f (e.g. an xgboost model), while the model complexity $\Omega(g)$ is kept low (e.g. prefer fewer features). G is the family of possible explanations, for example all possible linear regression models. The proximity measure π_x defines how large the neighborhood around instance x is that we consider for the explanation. In practice, LIME only optimizes the loss part. The user has to determine the complexity, e.g. by selecting the maximum number of features that the linear regression model may use.

The recipe for training local surrogate models:

- Select your instance of interest for which you want to have an explanation of its black box prediction.
- Perturb your dataset and get the black box predictions for these new points.
- Weight the new samples according to their proximity to the instance of interest.
- Train a weighted, interpretable model on the dataset with the variations.

[79]Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. "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. ACM (2016).

- Explain the prediction by interpreting the local model.

In the current implementations in R[80] and Python[81], for example, linear regression can be chosen as interpretable surrogate model. In advance, you have to select K, the number of features you want to have in your interpretable model. The lower K, the easier it is to interpret the model. A higher K potentially produces models with higher fidelity. There are several methods for training models with exactly K features. A good choice is Lasso. A Lasso model with a high regularization parameter λ yields a model without any feature. By retraining the Lasso models with slowly decreasing λ, one after the other, the features get weight estimates that differ from zero. If there are K features in the model, you have reached the desired number of features. Other strategies are forward or backward selection of features. This means you either start with the full model (= containing all features) or with a model with only the intercept and then test which feature would bring the biggest improvement when added or removed, until a model with K features is reached.

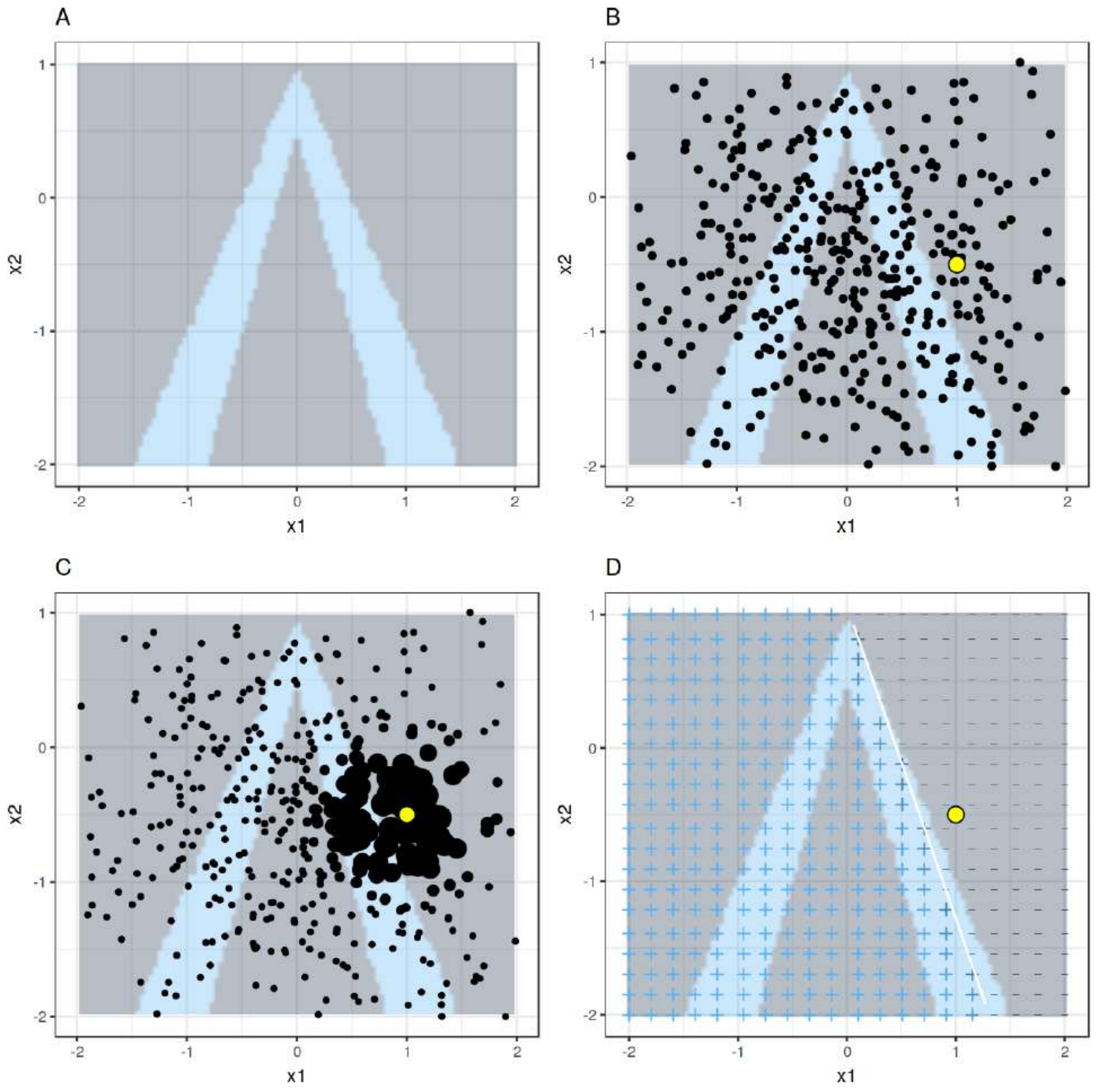
How do you get the variations of the data? This depends on the type of data, which can be either text, image or tabular data. For text and images, the solution is to turn single words or super-pixels on or off. In the case of tabular data, LIME creates new samples by perturbing each feature individually, drawing from a normal distribution with mean and standard deviation taken from the feature.

LIME for Tabular Data

Tabular data is data that comes in tables, with each row representing an instance and each column a feature. LIME samples are not taken around the instance of interest, but from the training data's mass center, which is problematic. But it increases the probability that the result for some of the sample points predictions differ from the data point of interest and that LIME can learn at least some explanation.

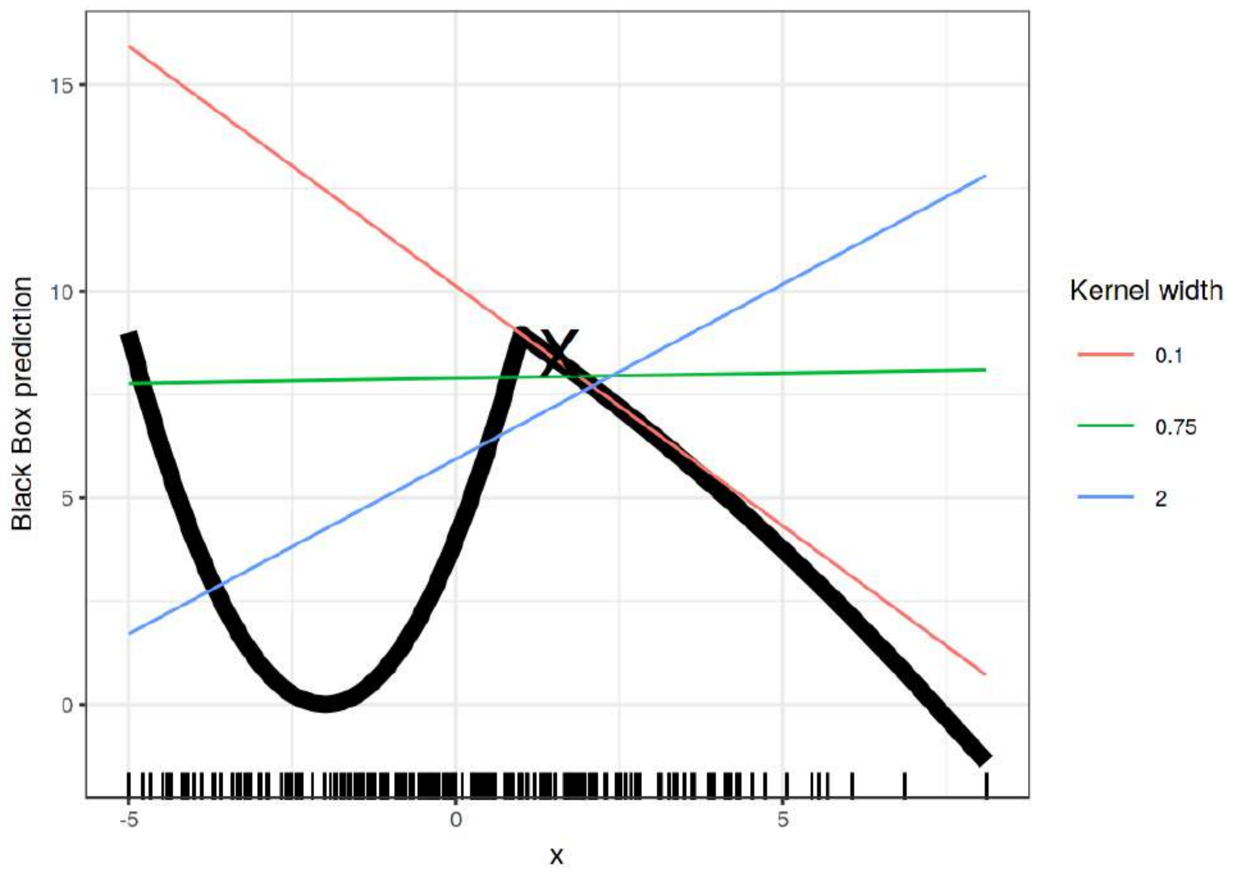
It is best to visually explain how sampling and local model training works:

[80]https://github.com/thomasp85/lime
[81]https://github.com/marcotcr/lime



LIME algorithm for tabular data. A) Random forest predictions given features x1 and x2. Predicted classes: 1 (dark color) or 0 (light color). B) Instance of interest (yellow dot) and data sampled from a normal distribution (black dots). C) Assign higher weight to points near the instance of interest. D) Colors and signs of the grid show the classifications of the locally learned model from the weighted samples. The white line marks the decision boundary (P(class=1) = 0.5).

As always, the devil is in the detail. Defining a meaningful neighborhood around a point is difficult. LIME currently uses an exponential smoothing kernel to define the neighborhood. A smoothing kernel is a function that takes two data instances and returns a proximity measure. The kernel width determines how large the neighborhood is: A small kernel width means that an instance must be very close to influence the local model, a larger kernel width means that instances that are farther away also influence the model. If you look at LIME's Python implementation (file lime/lime_tabular.py)[82] you will see that it uses an exponential smoothing kernel (on the normalized data) and the kernel width is 0.75 times the square root of the number of columns of the training data. It looks like an innocent line of code, but it is like an elephant sitting in your living room next to the good porcelain you got from your grandparents. The big problem is that we do not have a good way to find the best kernel or width. And where does the 0.75 even come from? In certain scenarios, you can easily turn your explanation around by changing the kernel width, as shown in the following figure:



Explanation of the prediction of instance x = 1.6. The predictions of the black box model depending on a single feature is shown as a black line and the distribution of the data is shown with rugs. Three local surrogate models with different kernel widths are computed. The resulting linear regression model depends on the kernel width: Does the feature have a negative, positive or no effect for x = 1.6?

The example shows only one feature. It gets worse in high-dimensional feature spaces. It is also very unclear whether the distance measure should treat all features equally. Is a distance unit for feature x1 identical to one unit for feature x2? Distance measures are quite arbitrary and distances in different dimensions (aka features) might not be comparable at all.

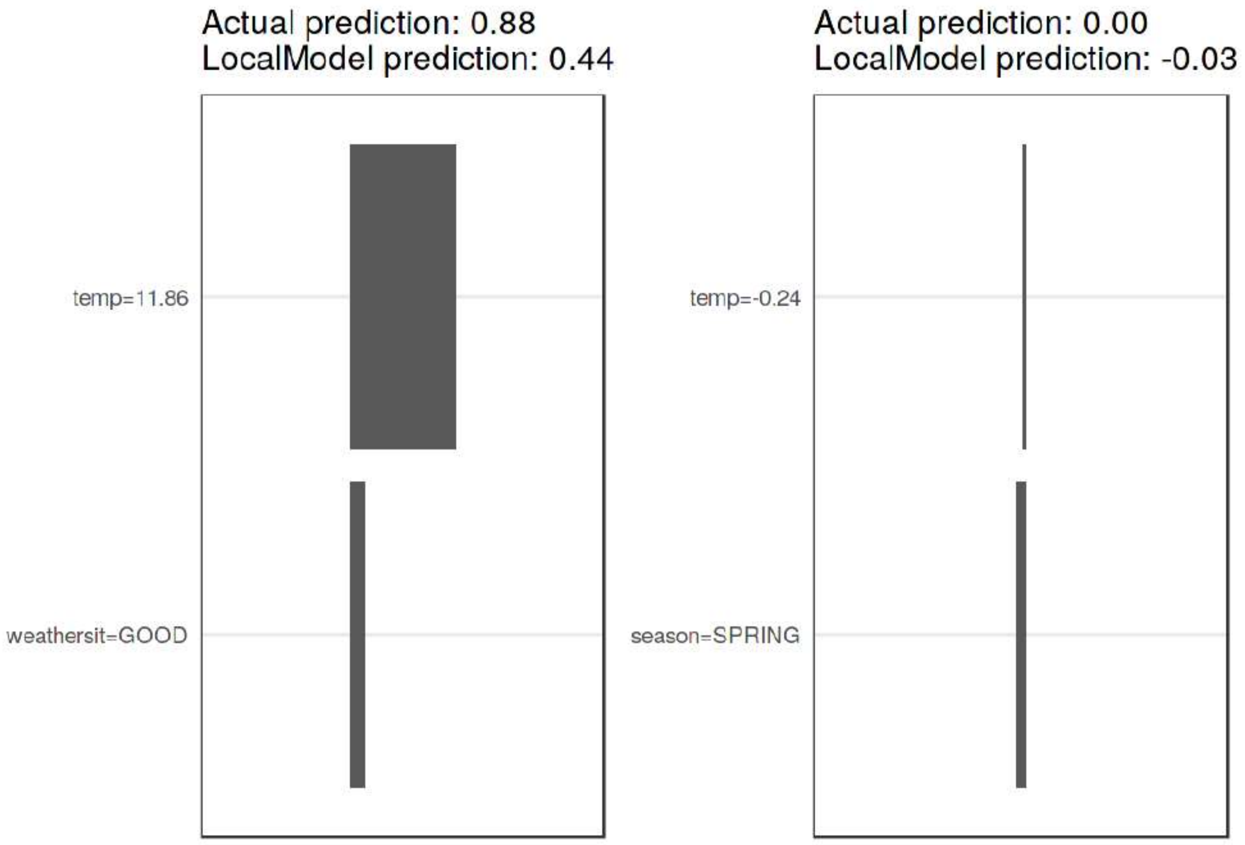
[82]https://github.com/marcotcr/lime/tree/ce2db6f20f47c3330beb107bb17fd25840ca4606

Example

Let us look at a concrete example. We go back to the bike rental data and turn the prediction problem into a classification: After taking into account the trend that the bicycle rental has become more popular over time, we want to know on a certain day whether the number of bicycles rented will be above or below the trend line. You can also interpret "above" as being above the average number of bicycles, but adjusted for the trend.

First we train a random forest with 100 trees on the classification task. On what day will the number of rental bikes be above the trend-free average, based on weather and calendar information?

The explanations are created with 2 features. The results of the sparse local linear models trained for two instances with different predicted classes:



LIME explanations for two instances of the bike rental dataset. Warmer temperature and good weather situation have a positive effect on the prediction. The x-axis shows the feature effect: The weight times the actual feature value.

From the figure it becomes clear that it is easier to interpret categorical features than numerical features. One solution is to categorize the numerical features into bins.