<https://compstat-lmu.github.io/iml_methods_limitations/lime-neighbor.html>

12.1

A LIME explanation could be retrieved by the following algorithm:

1. Get instance xx out of the data space for which we desire an explanation for its predicted target value.
2. Perturb your dataset XX and receive a perturbed data set ZZ of increased size.
3. Retrieve predictions ^yZy^Z for ZZ using the black box model ff.
4. Weight ZZ w.r.t. the proximity/neighborhood to xx.
5. Train an explainable weighted model gg on ZZ and the associated predictions ^yZy^Z

12.2

To better assess these open questions it may be helpful to study the mathematical definition of LIMELIME. The explanation for a datapoint xx, which we aim to interpret, can be represented by the following formula:

explanation(x)=argmingϵGL(f,g,πx)+Ω(g)explanation(x)=argmingϵGL(f,g,πx)+Ω(g)

Let's decompose this compact, yet precise definition:

xx can be an instance that is entirely new to us as long as it can be represented in the same way as the training data of the black box model. The final explanation for xx results from the maximisation of the loss-like fidelity term L(f,g,πx)L(f,g,πx) and a complexity term Ω(g)Ω(g). ff refers to the black box model we want to explain and gg to the explainer. GG represents the complete hypothesis space of a given interpretable learner. The explanation has to deal with two trade-off terms when minimizing: The first term L(f,g,πx)L(f,g,πx) is responsible to deliver the optimal fit of gg to the model ff while a low *loss* is desirable indicating high (local) *fidelity*. The optimal fit is only found with respect to a proximity measure πx(z)πx(z) in the neighborhood of xx.

13

As described in the previous chapter, LIME aims to create local surrogate models -- one for each observation to be explained. These local models operate in the proximity or neighborhood of the instance to be explained. They are fitted based on weights which indicate their proximity to the observation to be explained. The weights are typically determined using kernels that transform the proximity measure.

13.1

When obtaining explanations with LIME, the neighborhood of an observation is determined when fitting the model by applying weights to the data. These weights are chosen w.r.t. the proximity to the observation to be explained. However, there is no natural law stating that local models have to be found this way. Alternatively, Craven and Shavlik ([1996](https://compstat-lmu.github.io/iml_methods_limitations/lime-neighbor.html#ref-craven1996)) show that increasing the density of observations around the instance of interest is very helpful to achieve locally fidele models. Hence, locality could be obtained in many more different ways than weighting observations combined with global sampling as it is in LIME. After sampling, the data points are weighted w.r.t. their proximity to the observation to be explained. One possible alternative to this procedure might be to combine steps 2 (sampling) and 4 (weighting) of the LIME algorithm[2](https://compstat-lmu.github.io/iml_methods_limitations/lime-neighbor.html#fn2) to a local sampling. This way we would increase the density around the instance already by proper sampling. In fact, Laugel et al. ([2018](https://compstat-lmu.github.io/iml_methods_limitations/lime-neighbor.html#ref-laugel2018defining)) claim that this way should be preferred over the way LIME samples. In this chapter, however, we focus on the explicit implementation of LIME and analyze how the weighting strategy *ceteris paribus* affects surrogate model accuracy and stability.

When working with LIME, the weighting of instances is performed using a kernel function over the distances of all other observations to the observation of interest. This leaves us *arbitrary* (in fact, they may not be *that* arbitrary) choices on two parameters: the distance and the kernel function. Typical distance functions applicable to statistical data analysis are based on the L0, L1 and L2 norms. For numerical features, one tends to use either Manhattan distance (L1) or Euclidean distance (L2). For categorical features, one would classically apply Hamming distance (L0). For mixed data (data with both categorical and numerical features), one usually combines distances for numerical and categorical features. So does Gower's distance (Gower ([1971](https://compstat-lmu.github.io/iml_methods_limitations/lime-neighbor.html#ref-gower1971general))) or the distance proposed by Huang ([1998](https://compstat-lmu.github.io/iml_methods_limitations/lime-neighbor.html#ref-huang1998kproto)):

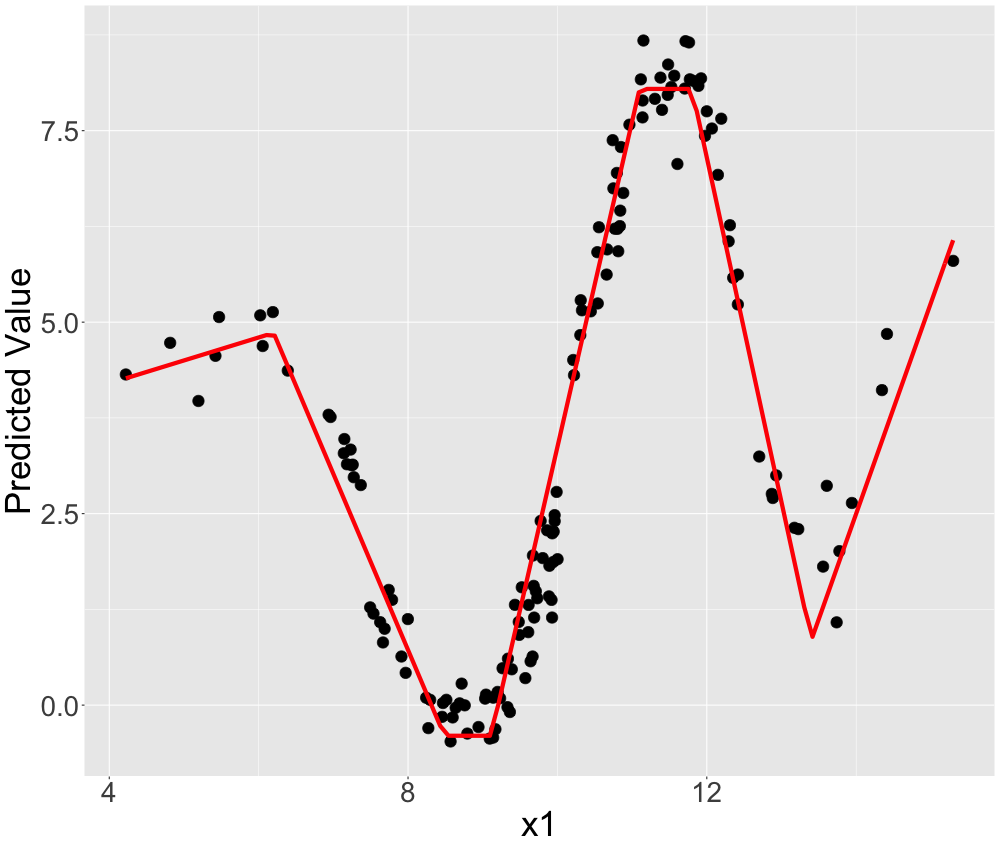
dH(xi,xj)=deuc(xi,xj)+λdham(xi,xj)dH(xi,xj)=deuc(xi,xj)+λdham(xi,xj)

with deucdeuc referring to the Euclidean distance and dhamdham to the Hamming distance. deucdeuc is only computed for numerical and dhamdham only for categorical ones. λλ steers the importance of categorical features relative to numerical ones. Huang ([1998](https://compstat-lmu.github.io/iml_methods_limitations/lime-neighbor.html#ref-huang1998kproto)) recommends setting λλ equal to the average standard deviation of the numerical features. For scaled numerical features (standard deviation is one) this metric is equivalent to the Euclidean distance. It is important to note that despite these existing measures it may be challenging to properly determine distances for mixed data. For text data, M. T. Ribeiro, Singh, and Guestrin ([2016b](https://compstat-lmu.github.io/iml_methods_limitations/lime-neighbor.html#ref-ribeiro2016should)) recommend using cosine distance and Euclidean distance for images.

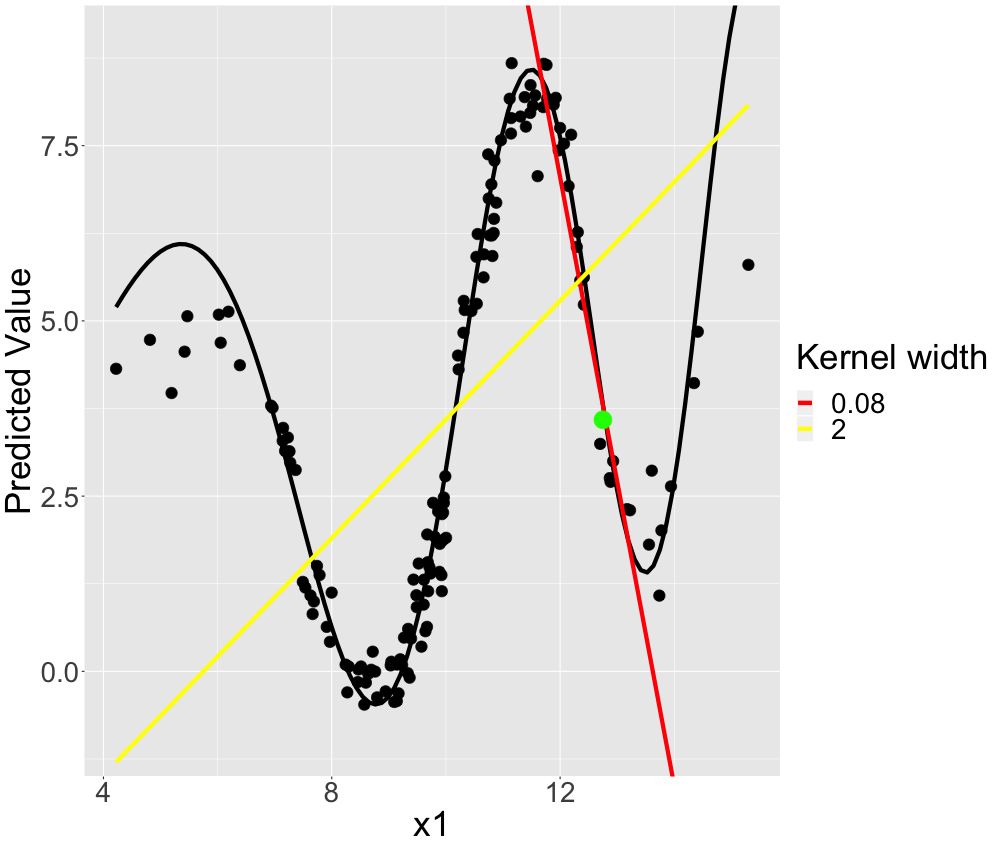
For the kernel function itself, there are two parameters to be set. First of all, the type of kernel. Second, the kernel width. By default, the R implementation uses an exponential kernel where the kernel width equals the square root of the number of features.

The choice of the distance measure seems least arbitrary. Furthermore, the choice of the kernel function is not expected to have the most crucial impact on the neighborhood definition. Thus, we focus on the **kernel width** in our experimental study.

13.2



LIME with best kernel width



LIME with not great kernel widths

13.3

As before, we observe that a too-small kernel width (<0.1<0.1) produces non-meaningful coefficients. On the other hand, for large kernel widths (>0.7>0.7) the true coefficient is not approximated, but rather the global (average) linear coefficient: For x1x1 a large kernel width results in a linear model that averages the local slopes.

The more complex the true associations become, the more we observe this trend of decreasing solution intervals. It seems as if the more complex the predictive surface is, the harder it is for LIME to even find a good local model.

For globally non-linear associations, we also find that we prefer a small kernel which however also produces stable coefficients.

A (too) small kernel width usually worsens coefficient stability whilst a too large kernel width fits a global surrogate model. An optimal kernel size should balance these effects. We may formulate the problem as a minimization problem w.r.t. the kernel size. However, the minimization needs to consider the constraint that coefficients need to be stable.

13.4

Leaving the controlled environment may make things more difficult. Relevant challenges include:

1. High-dimensional data may be an issue for the computation of the kernel width. LIME computes dissimilarities. It is well-known that (some) dissimilarities get increasingly less meaningful as the feature space expands. This is one consequence of the curse of dimensionality.
2. Computing some dissimilarities (e.g. Manhattan or Euclidean) also comes with the problem that the cardinality of the features mainly steers this measure. Thus, LIME should always apply scaling.
3. When working with real data sets with many features, we typically want a sparse explanation. To achieve this, we should let LIME perform feature selection.

13.4

Summarizing, we observe both effects being described in the literature also for our real data example: instability (Alvarez-Melis and Jaakkola ([2018](https://compstat-lmu.github.io/iml_methods_limitations/lime-neighbor.html#ref-alvarez2018robustness))) for small kernel widths and global surrogates (Laugel et al. ([2018](https://compstat-lmu.github.io/iml_methods_limitations/lime-neighbor.html#ref-laugel2018defining))) for large ones.

We think that the global sampling of LIME is responsible for many of the pitfalls identified. Hence, we propose that LIME should be altered in the way proposed by Laugel et al. ([2018](https://compstat-lmu.github.io/iml_methods_limitations/lime-neighbor.html#ref-laugel2018defining)) to LIME-K. Local sampling should replace global sampling to better control for the locality.

Domains where LIME has been applied successfully include image data and text data. Within these domains, LIME works differently from tabular data. For example, LIME's sampling for text data is already very local. It only creates perturbations based on the instance to be explained.