NOTES ON LOESS IMPLEMENTATION

ANDREW BLANDINO

1. Introduction

- Loess is a form of *non-parametric* regression i.e., assume regression model has the form
$$y = f(x) + \varepsilon$$
for some function $f(\cdot)$ (may be continuous, differentiable etc.).
- Easy example of *parametric* regression is linear regression
$$y = a + bx + \varepsilon$$
which assumes $f(x) = a + bx$. This assumption is often untenable on a global scale.
- Loess regression fits to non-linear functions, f, by using *local* linear regression
$$y = a(x) + b(x)x + \varepsilon$$
where coefficients $a(x), b(x)$ depend on x (hence, local).
- The challenge is how best to estimate $a(x)$ and $b(x)$ and how 'local' should the process be?

1.1. **Loess algorithm.**

The following Loess algorithm is from the original formulation from [Cleveland et al., 1992]. Let $(x_1, y_1), \ldots, (x_n, y_n)$ denote the dataset. Then the following hyper-parameters are needed

- Span: determines the length of the sliding window used to define a local fit. Let span $\in (0, 1)$ and take the rounded-up integer of span $\cdot\, n$, that is $s := \lceil \text{span} \cdot n \rceil$. This integer s will be used to determine the window length in the following way: for any data point x_0, calculate the distances of x_0 to any data point x_i given by
$$\Delta_i(x_0) = |x_0 - x_i|.$$
With these distances, we can then sort them as an increasing sequence
$$\Delta_{(1)}(x_0) \le \Delta_{(2)}(x_0) \le \cdots \le \Delta_{(n)}(x_0).$$

Date: March 31, 2022.

We choose the window length to be the sth smallest pairwise distance, $\Delta_{(s)}(x_0)$. That is, the window length $\lambda(x_0)$ is

$$\lambda(x_0) = \Delta_{(s)}(x_0).$$

(Note that the window length depends on span through the nearest integer s.) If span > 1, we instead define

$$\lambda(x_0) = \Delta_{(n)}(x_0) \cdot \text{span}^{1/d}$$

where d is the degree of local polynomial being fit (see below).

- Weight/window function: used to weigh observations, with closer observations obtaining more weight. The classic Loess algorithm uses the *tri-cube* weight function given by

$$W(x,t) = \begin{cases} (1 - |x|^3/t^3)^3 & 0 \le |x| \le t, \\ 0 & |x| > t. \end{cases}$$

With a window length λ, then the function $W(x - x_0, \lambda)$ will give maximum weight for $x = x_0$, that is $W(0, \lambda) = 1$, while giving less and less weight as x moves further away from x_0. The weight becomes 0 once x has a distance of λ or more from x_0. Note, other weight functions are possible as long as they satisfy the following conditions:

Symmetry: $W(x,t) = W(-x,t)$ (weighting only depends on distance, not direction).

Non-negativity: $W(x,t) \ge 0$ for all x and $t > 0$.

Monotonicity: for each $t > 0$, $W(x,t)$ is decreasing for all $x > 0$ (distant points have less weight than closer points).

Bounded: $W(x,t) \le K$ for all (x,t) for some fixed $K > 0$. (Typically, we would want $K = 1$).

- Degree of local fit: the Loess procedure fits a local linear regression in x and hence can allow for polynomial fits. The typical choices are

linear:

$$\sum_{i=1}^{n} W(x - x_i, \lambda(x)) \times (y_i - \beta_0(x) - \beta_1(x)x_i)^2$$

quadratic:

$$\sum_{i=1}^{n} W(x - x_i, \lambda(x)) \times (y_i - \beta_0(x) - \beta_1(x)x_i - \beta_2(x)x_i^2)^2$$

With the following parameters specified then the Loess fit can be fit for any point x by solving

$$\min_{\beta_0,\beta_1,\beta_2} \sum_{i=1}^{n} W(x - x_i, \lambda(x)) \times (y_i - \beta_0(x) - \beta_1(x)x_i - \beta_2(x)x_i^2)^2$$

using weighted least squares and obtain $\hat{\beta}_0(x), \hat{\beta}_1(x), \hat{\beta}_2(x)$ for the given x. Then the Loess prediction is given by

$$\hat{y} = \hat{\beta}_0(x) + \hat{\beta}_1(x)x + \hat{\beta}_2(x)x^2.$$

2. Selection of Fitting Parameters

The most important fitting parameter is the **span**, which acts as a 'smoothing parameter'. The choice of the span is important to avoid two extreme problems:

under-smoothing: The span is *too small*, which results in extremely wild and jagged fits. The result is typically a poor representation of the underlying process (*principal of parsimony*).

over-smoothing: The span is *too large*, which results in 'too smooth' of a fit that does not capture the trend in the data. This may capture a global trend well, but loses the fine details for portions of the data.

Thus, the ideal goal is to land somewhere in between with the 'just right' amount of smoothing by a *well-chosen* **span** from the data.

In some cases, the amount of smoothing can be chosen reasonably well by eye and some trial and error. But in cases with a high density of points, where the eye can be more easily misled, the need for model fitting criterion becomes important.

2.1. Generalized Cross-Validation.

Cross-validation (CV) is a procedure for estimating the out-of-sample performance of a fitted model using current data. This is accomplished by using a portion of data for fitting the model (training) and a separate portion for evaluating performance (testing/validation). A popular version of this is Leave-one-out CV, or LOOCV, that uses an individual data point as a test data set while the remaining are used for training and repeats this for all data points. The performance is evaluated using the average squared error over all test sets i.e,

$$LOOCV(\textbf{span}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}^{(-i)}(x_i))^2,$$

where $\hat{f}^{(-i)}(x_i)$ is the prediction of y_i using input x_i with the trained model using all data points except x_i. Note, the estimated model $\hat{f}$ depends on the choice of **span**. Minimizing the value of the LOOCV over **span** typically gives the right amount of smoothing.

However, this requires n separate model fits for each data point x_i which can be costly for large data sets and/or complex models. A fast approximation of this given by Generalized Cross-Validation (GCV) which is given by

$$GCV(\textbf{span}) = \frac{1}{n}\sum_{i=1}^{n}\frac{(y_i - \hat{f}(x_i)^2}{(1 - \text{trace}(S)/n)^2},$$

where S is the 'smoothing' matrix for the model fit $\hat{f}$ that satisfies $\hat{y} = Sy$ (note: $\text{trace}(S)$ is the *trace* of the matrix S, which is the sum of the main diagonal elements i.e. $\text{trace}(S) := \sum_{i=1}^{n} S_{ii}$). Since loess is still a linear method, even though it is *local*, there is still an associated S matrix which can be used for this purpose.

2.2. **Selecting span by GCV.** To select for **span** by GCV, the goal is to find the value of **span** that minimizes $GCV(\textbf{span})$. Since the minimization is too difficult to do analytically, in practice we can obtain a close to optimal choice of **span** by a *grid-search.* A simple grid-search is to select a *grid* of candidate **span** points, typically N equally spaced points, t_j, in some interval $[a, b]$ with $a < 1$ and $b = 1$. If $a = t_1 < t_2 < \cdots < t_n = b$ is the chosen grid, then we select **span** by

$$\textbf{span} = \text{argmin}_{t_1,\ldots,t_N}\{GCV(t_j)\}.$$

It is also good practice to plot the points $(t_j, GCV(t_j))$ to see how the GCV curves behaves as a function of t_j. There is some trial-and-error here as well, since the choice of grid-points t_j is subjective, but here's some points to keep in mind:

(1) The lower-bound of the grid should be small enough to show the 'under-smoothing' behavior kick-in, which will show up in the GCV plot with larger errors as a result. But the lower-bound can be too small, which will cause issues with the local weight calculations, but this depends on the density (or sparisty) of the data for the x-axis for what is 'too small'.
(2) The more grid points the better, for finding the 'true optimal value', but this comes at more computational cost. Grid lengths of $N = 100$ to $N = 500$ seem about right for a 'smooth' enough curve for GCV, but this also depends on the grid bounds $[a, b]$.
(3) There is not much utility in selecting the grid upper bound, b, to be larger than $b = 1$. At $b = 1$, the procedure is practically 'global' and gives very similar results to ordinary linear regression. It's not a bad idea to leave $b = 1$ as the upper-bound, since the values of GCV being minimized near $b = 1$ suggest a local fit may be unnecessary.

Example. Let $f(x) = \sin(2\pi x)$, the true regression function, and the data generated according to

$$y_i = f(x_i) + \varepsilon_i = \sin(2\pi x_i) + \varepsilon_i$$

for x_i sampled in $[0, 1]$. A possible realization is shown in Figure 1 ($f(x)$ in black). A plot of the estimated GCV and AICC curves is given for a grid of **span** parameters in Figure 2 and notice the clear minimum at the lower-end. The previouis figure shows the Loess fit using the optimal **span** from GCV along with a 'trial and error' value of 0.5 by inspecting the GCV curve. The 'trial and error' version is typically preferable since the GCV selected minimum shows some signs of overfitting (more wiggly in some regions). This can be argued since the GCV curve still suffers from estimation error, so we can err on the side of 'over-smoothing' (less wiggly) and pick a larger value of **span**, such as 0.5.

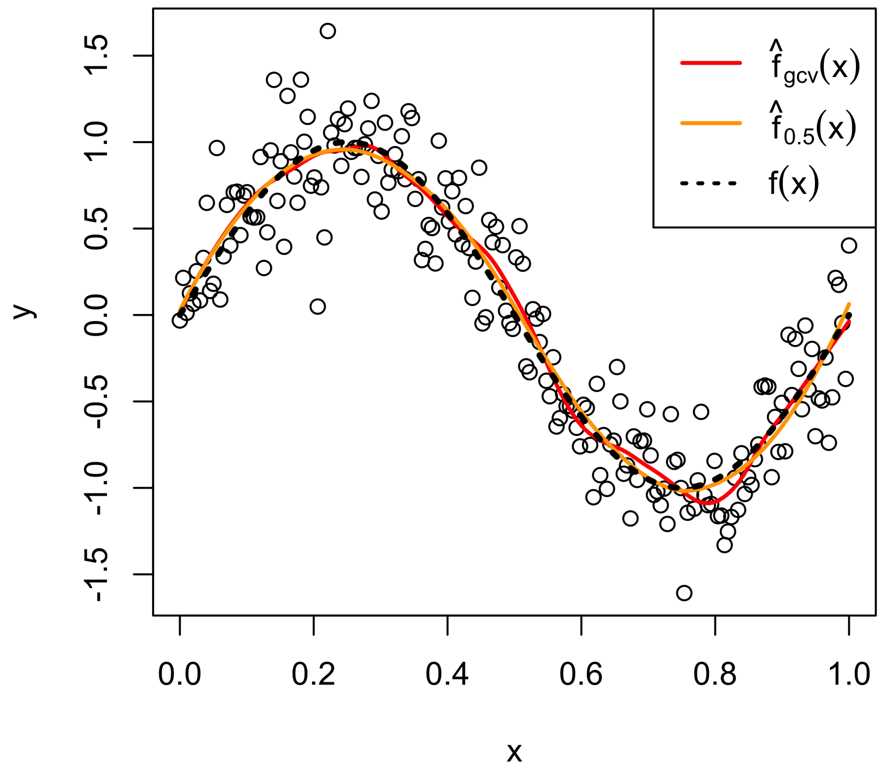

FIGURE 1. Comparison of two loess fits: GCV selected minimum (red) and 'trial and error' version (orange) along with the true regression function (black).

Lastly, it should be noted the GCV and AICC curves in Figure 2 both show a steady increasing trend with **span** past a certain point, around 0.5, meaning a consistently worse fit. Since the true function is non-linear (specifically, quadratic in x), this makes sense and shows the 'over-smoothing' behavior mentioned previously.

2.3. **Akaike Information Criterion (Corrected).** Another option for selecting **span** is the use of Akaike Information Criterion (AIC), a popular criteria for selecting competing models. An improved version is the *corrected* AIC (AICC), which corrects the original AIC for small sample sizes. When the sample size is large enough, AIC and AICC are very similar. For that reason, it's just as well to use AICC in general.

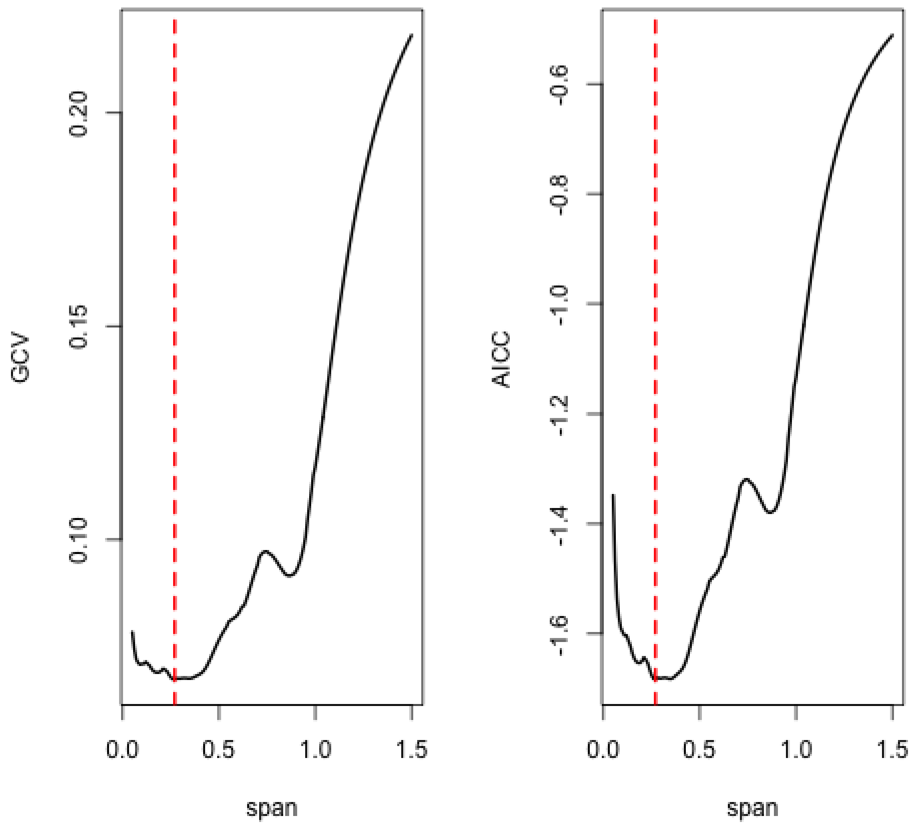

FIGURE 2. Both estimated curves for selecting **span** have similar minimum values (red dashed line) and a similar overall shape. Given the variability in the estimated curve, we can see that the 'trial and error' choice of 0.5 is in the neighborhood of the minimum for both curves.

The AICC criterion is given by

$$AICC(\textbf{span}) = \log\left(\tfrac{1}{n}\textstyle\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2\right) + 1 + \frac{2(\text{trace}(S)+1)}{n - \text{trace}(S) - 2}.$$

Selecting **span** by AICC can be done simultaneously, and in a similar fashion, as with GCV, since they both rely on the smoothing matrix S. In general, there should be little difference in the choice of **span** by AICC or GCV for most problems.

3. Bias Correction for log-log regression

Consider the log-log version with non-parametric regression.

$$\log y_i = a(x) + b(x)\log(x_i) + \varepsilon_i$$

which is fitted locally by loess, using the local coefficients $a(x)$ and $b(x)$. This gives the residuals

$$\hat{\varepsilon}_i = \log y_i - \hat{a}(x) - \hat{b}(x) \times \log(x_i)$$

that can then be used for bias-correction methods such as [Ferguson, 1986] and [Duan, 1983]. Specifically, the Ferguson correction first calculates the log-log scale error variance

$$\hat{\sigma}^2 = \frac{1}{\delta_1}\sum_{i=1}^{n} \hat{\varepsilon}_i^2,$$

with $\delta_1 = \text{trace}(L^T L)$ where $L = I_n - S$. Then, assuming normality and uncorrelated errors ε_i, you form the bias-correction factor

$$\widehat{\text{bias}}_F = e^{\ln(10)^2\hat{\sigma}^2/2}$$

and the bias-corrected estimate is then

$$\hat{y}_i = 10^{\hat{a}(x)} \times x_i^{\hat{b}(x)} \times \widehat{\text{bias}}_F.$$

Similarly, the Duran bias-correction factor is given by

$$\widehat{\text{bias}}_D = \frac{1}{n}\sum_{i=1}^{n} 10^{\hat{\varepsilon}_i},$$

which is a non-parametric alternative that is used in a similar fashion.

References

[Cleveland et al., 1992] Cleveland, W. S., Grosse, E., and Shyu, W. M. (1992). Local regression models. In *Statistical Models in S*, pages 309–376. CRC Press, 1 edition.

[Duan, 1983] Duan, N. (1983). Smearing estimate: A nonparametric retransformation method. *Journal of the American Statistical Association*, 78(383):605–610.

[Ferguson, 1986] Ferguson, R. I. (1986). River loads underestimated by rating curves. *Water Resources Research*, 22(1):74–76.