WIGGLING THE BANDWIDTH

LUCA WELLMEIER

Abstract. We explore the possibility of using two different bandwidths in the fitting and evaluation part in kernel ridgeless regression. The provided experimental results indicate that *wiggling* the bandwidth can provide generalization and improve over standard estimators. Finally, we propose and test a new iterative method for optimizing the bandwidth parameter by employing wiggling in a cheap local parameter search.

Contents

1. Introduction

Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be a matrix of n training examples $X_1, \ldots, X_n \in \mathbb{R}^d$ in d-dimensional Euclidean space as rows and let Y be the column vector of responses $y_1, \ldots, y_n \in \mathbb{R}$. The main actor of this paper is the kernel ridgeless regression estimator that predicts the response on an unseen data point $X \in \mathbb{R}^d$ as

$$\hat{f}(X) = \sum_{i=1}^{n} (\mathbf{K}^\dagger Y)_i K(X, X_i). \tag{1}$$

Here, $\mathbf{K}$ denotes the kernel matrix $\mathbf{K}_{ij} = k(X_i, X_j)$ is the *kernel matrix* obtained by evaluating a kernel function $k \colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ on the training examples and $\mathbf{K}^\dagger$ denotes its *pseudo-inverse*. We briefly recall the motivations behind these notions. First, the *representer theorem* asserts that any minimizer to the least squares problem

$$\min_{f \in \mathcal{H}_k} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(X_i))^2$$

in the *reproducing kernel Hilbert space* $\mathcal{H}_k$ associated to k, is given by a linear combination of the form $\sum_{i=1}^{n} c_i k(X, X_i)$. This allows to apply the usual linear least squares method to a - at the first glance - non-linear approximation problem in a possibly infinite-dimensional hypothesis space by pushing the data points into the *feature space* spanned by $K(X, \cdot) \in \mathcal{H}_k$ for $X \in \mathbb{R}^d$. The kernel matrix in this setting corresponds to the empirical covariance matrix $\mathbf{X}^T\mathbf{X}$ and classical theory suggests that, even if it does not possess full-rank, there will always exist a minimizer to the least squares problem above. The pseudo-inverse then corresponds to the unique least-squares solution with minimal $\mathcal{H}_k$-norm. This reduction of an infinite-dimensional optimization problem to a finite-dimensional one using kernels is generally known as the *kernel trick*.

Recently, there was a new wave of interest into kernels sparked by the fundamental question of why highly over-parameterized deep neural nets perform as well as they do given that the usual established model, the *bias-variance trade-off*, does not capture this regime. The idea is that in order to find a good estimator one should choose the complexity of the hypothesis space to be high enough to avoid *underfitting* but simultaneously low enough to avoid *overfitting*. For instance, if we roughly identify the number of adjustable parameters of a model with the complexity of the hypothesis space, we would expect a U-shaped test error curve over the number of parameters with very high risk at the extrema where N_{param} is very small but also $N_{\text{param}} \approx N_{\text{train}}$, where the model is able to *interpolate* the training data - it learns them by heart and the analogy is the same as in the human experience where learning by heart does not give you true understanding. However, modern, practically successful, deep neural nets tend to have a parameter count that exceeds the training sample count by orders of magnitude. This is not captured by classical theory so the question is two-fold: is this the reason for their performance and what exactly happens in this *interpolating regime*. A good start into the literature would be the following papers on the *double descent* phenomenon ([1]) and on the concept of *benign overfitting* ([2]).

In particular, there are many indications that kernel learning methods behave analgously to deep learning from many different perspectives. Much evidence was provided in [3]. First, many kernels as for instance Laplacian and Gaussian are always able to interpolate the training data perfectly, providing a prime example of interpolating estimators. Unlike deep neural nets, kernel methods are easy to handle not only on paper but also practically due to the very small amount of hyper-parameters and there are many tasks in which their performance is comparable to the latter or even exceed it. In [4], the authors provide experimental and theoretic evidendence that it is better to interpolate instead of the classically suggested norm penalties (*à la* Tikhonov) under certain circumstances. Now, these circumstances have not been fully understood but some necessary conditions seem to have pinned down: high-dimensionality ([6, 2]) with low effective dimension ([2]) and properly chosen curvature of the kernel w.r.t. the nature of the data ([4]). Moreover, there seems to be strong links ([5]) between Laplacian kernels and ReLU feed-forward neural nets through another kernel

called the *neural tangent kernel* that is able to capture the learning dynamics of gradient descent there.

These considerations have also sparked new interest in the other direction: due to all the analogies between kernels and neural nets researchers have started to scale up kernel methods to deal with big data (see for instance [7]). Interesting targets have been kernels coming from the Sobolev-Hilbert spaces $H^s(\mathbb{R}^d)$ with real smoothness index: the *Matérn kernels*. This family's smoothness limit member is the Gaussian kernel, but we will (due to the motivations above) consider only the well known the Laplacian kernel

$$k_\gamma(x, x') = \exp(-\gamma\|x - x'\|)$$

which belongs to the less smooth part of the spectrum. The only hyperparameter to adjust is the bandwidth γ and in this paper we investigate experimentally a new principle on how to both optimize it in large-scale contexts and use it for regularization purposes.

The full code of the experiments can be found on GitHub[1].

2. Wiggling

As announced, let us fix the Laplacian kernel function

$$k_\gamma(x, x') = \exp\left(-\gamma\|x - x'\|\right)$$

with bandwidth $\gamma > 0$. The convention here is that smaller bandwidth actually increases the area under the curve $x \mapsto k_0(x) \coloneqq k(0, x)$ and vice-versa. (different conventions everywhere!). This whole note is based on the observation that the kernel ridgeless estimators use the bandwidth twice in distinct places: once in the computation of the *dual coefficients*

$$c_i = (\mathbf{K}_\gamma^\dagger Y)_i$$

where the kernel matrix depends on it and then in the evaluation part

$$\sum_i c_i k_\gamma(x, x_i)$$

where the kernel is computed to check similarities between sample points x_i and unseen points x. Note that the dependence of the dual coefficients on the bandwidth is not even guaranteed to be continuous and generally hard to understand due to the appearing pseudo-inverse. However, the evaluation part offers a different perspective, if we fix the dual coefficients c_i and an unseen point x, then the relationship

$$\gamma \mapsto \sum_{i=1}^{n} c_i k_\gamma(x, x_i)$$

is smooth: indeed, γ appears only as a factor to the non-smooth norm in the exponential.

[1] `https://github.com/lucw0/wiggle-bandwidth`

Thus, we propose the new *bandwidth-wiggled* estimators

$$\hat{f}_{\gamma_0,\gamma}(x) \coloneqq \sum_{i=1}^{n} (K_{\gamma_0} Y)^\dagger k_\gamma(x, x_i).$$

The pseudo-inverse is computed for a given base bandwidth γ_0 but the evaluation part is performed with a different bandwidth γ. Due to the regularity argument above, we hypothesize that changing $\gamma_0 \to \gamma$ in the evaluation part is well-behaved. The next chapters are devoted to experimentally verifying this, but also to explore possible new methods based on this estimator.

The first method that we introduce here is referred to as *wiggle search*. Let Tr denote the training set with n samples. Split it into two non-empty parts: $\mathrm{Tr} = \mathrm{Tr}_1 \sqcup \mathrm{Tr}_2$ and train the a standard Laplacian ridgeless estimator only on Tr_1 with an initial guessed bandwidth γ_0. Then extract the dual coefficients $c_i^{\gamma_0}$ from there and use them to define the family predictors

$$\hat{f}_{\gamma_0,\gamma} \coloneqq \sum_{i \in \mathrm{Tr}_1} c_i^{\gamma_0} k_\gamma(\cdot, x_i)$$

for each γ in a finite set $\Gamma \subset \mathbb{R}_{>0}$. Then select γ^* such that

$$\gamma^* = \mathrm{argmin}_{\gamma \in \Gamma} \hat{R}_{\mathrm{Tr}_2}\left(\hat{f}_{\gamma_0,\gamma}\right),$$

i.e. the associated wiggled estimator minimizes the empricial mean squared error on the held-out part of the sample. This algorithm is more efficient in comparison to computing the standard estimators: The least squares computation to determine the dual coefficients is cubic in the sample size, so as a first improvement, we reduce this time effectively by splitting the sample. Moreover, existing techniques for optimizing the bandwidth (i.e. grid search, cross-validation, ...) test one parameter value per least squares computation. Here, instead, potentially thousands of parameters are tested with on single least squares assuming that $\hat{f}_{\gamma_0,\gamma}$ is representative of the actual standard estimator $\hat{f}_{\gamma,\gamma}$.

Note that the comparison of the different bandwidths can be computed in a highly vectorized way: Let $\mathbf{D}$ denote the distance matrix of 2-norms between Tr_1 and Tr_2. Once it is computed, we can test a new wiggled estimator for *any* other bandwidth γ by running the vectorized computation

$$\left(\mathbf{c}^{\gamma_0} \exp(-\gamma \mathbf{D})\right)^T$$

yielding the new predictions of the inputs in Tr_2. Thus, the big hard part of the computation, i.e. finding $\mathbf{D}$, needs to be done only once for size of Γ.

If wiggle search performs well and compares to the standard estimator with the optimal wiggled bandwidth (as we will see in the experiments), we can start considering an *itertated wiggle search*: run a wiggle search, find the optimal bandwidth, recompute the least squares with that bandwidth on Tr_1 and repeat. That will make up our final experiment in the paper.

Summing up, here is the questions we aim to provide experimental evidence for:

(1) How close are wiggled estimators and standard estimators with the wiggle bandwidth? What is the radius of γ's around γ_0 such that one can move knowledge from one to the other? How to choose the wiggle range Γ?
(2) What would be a good point to split the training set in the context of wiggle search? When does wiggle search provide an advantage over standard estimators (or if at all)?
(3) Does an iterated wiggle search find the best bandwidth? Is it less expensive than classical model selection techniques?

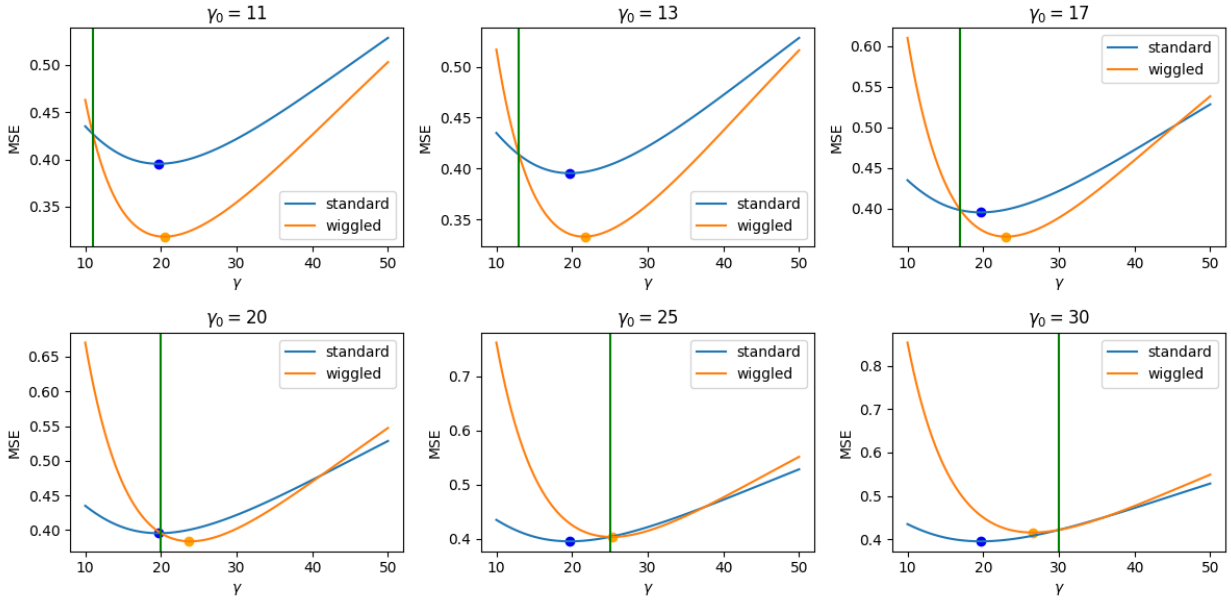

FIGURE 1. Standard versus wiggled estimators. The green vertical line shows the base bandwidth. The dots indicate the minima. The blue line is indepdent of the base bandwidth.

2.1. **Experiment: How close are wiggled and standard estimators?** In this first experiment we want to find out the relationship between wiggled estimator $\hat{f}_{\gamma_0,\gamma}$ and standard estimator $\hat{f}_{\gamma,\gamma}$. We choose a reasonably hard function $[3/2, 3] \to \mathbb{R}$, namely

$$f(x) = \zeta(x)\sin(20x),$$

and pick $N_{\mathrm{Tr}} = 30$ uniformly sampled training points. In a range of $\Gamma = [10, 50]$ we plot the performance on a training set (regular grid, $N_{\mathrm{Te}} = 1000$) of the "standard" estimator for each γ and the "wiggled" estimator wiggled for the same γ but previously fit on a fixed base bandwidth γ_0. We plot this 6 times for different base bandwidth. See fig. 1 for the results.

We summarize the observations:

- Generally, the behavior of wiggling γ away from γ_0 is smooth as expected.

- The first four plots show that wiggling improves over standard estimators even if the base bandwidth is already optimal.
- If the base bandwidth is chosen too small, then a iterated wiggle search goes in the direction of the standard optimal bandwidth.
- However, plot no. 5 shows that this seems not be the case in general.

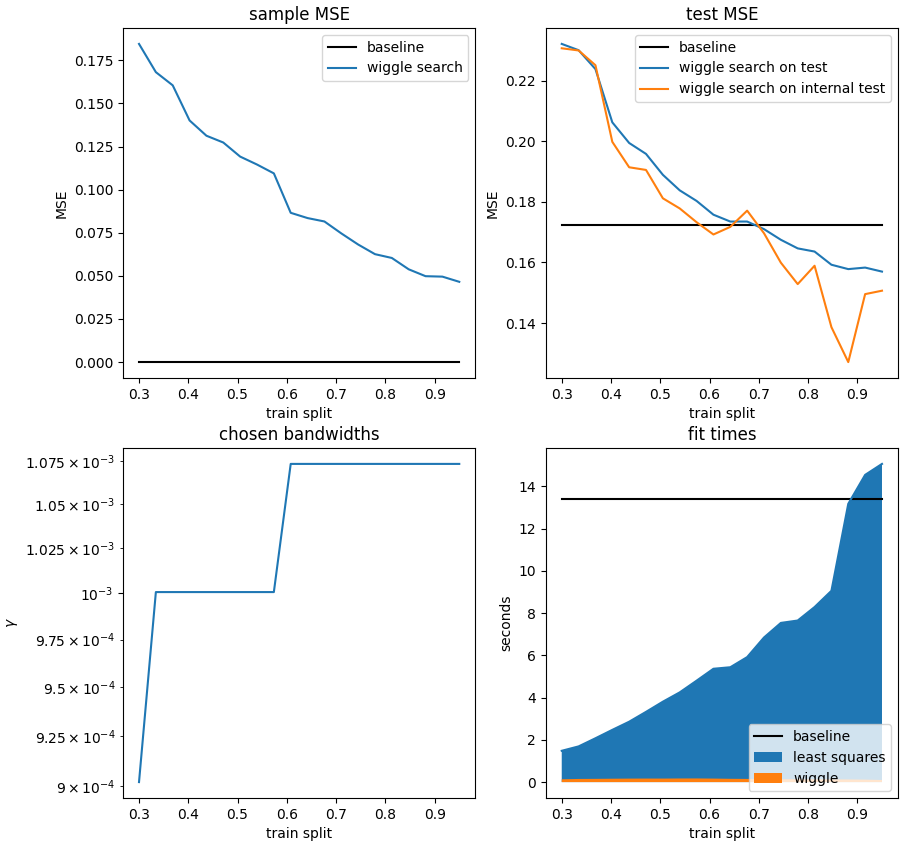

FIGURE 2. Figuring out the sweet spot for the train split on real data. "Baseline" always refers to the standard estimator with the base bandwidth on the full train set. The test MSE curves refer to the best wiggled bandwidth.

2.2. **Experiment: Where to split the training set?** The last experiment gave hope that wiggle search can work well. In this part we will try to figure out the sweet spot between computational savings and performance gain. We move to real data now.

We choose the MNIST binary classification task with digits 4 and 9 (the hardest to distinguish) with fixed sample of size $N_{\mathrm{Tr}} = 1000$. The set $\Gamma = \Gamma(\gamma_0)$ is being chosen at each base bandwidth as a small logarithmic scale clustered at the center γ_0 (contains 30 bandwidths) and we fix $\gamma_0 = 1/784$ which is the `sklearn` default choice (based purely on dimensionality). The splits are percentages p of N_{Tr}. All plotted values are with respect to $p \in [0.3, 0.95]$: sample error, error on held-out set, error on test set, as well as the chosen bandwidths and the fitting times. See fig. 2 for the results.

We summarize the observations:

- The more data we are considering, the bigger we choose the bandwidth parameter (e.g. the smaller the area under the kernel function will be) as one would expect.
- We achieve equivalent performance just by using only 70% of the training set plus wiggling. After that point we see only improvements in favor of wiggling.

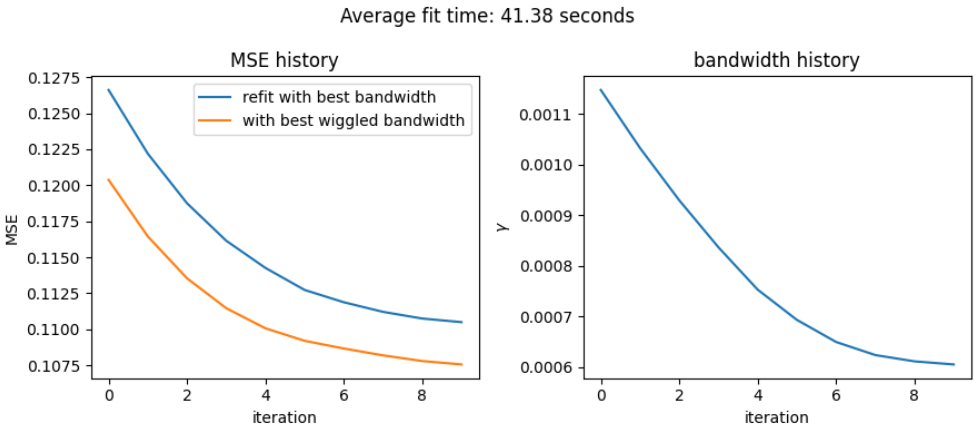

FIGURE 3. Iteratively applying wiggle search and then refitting a standard estimator with the found optimal bandwidth.

2.3. **Experiment: Iterative wiggling as cheap bandwidth optimization.** We have already seen in the first experiment that the direction given by wiggle search seems to be the right one. In this experiment we are going to test the iterated wiggle search as described above: fit least squares, wiggle search, refit with the best bandwidth, and so on. If the chosen bandwidth converges to something, we are likely to have hit an optimum or at least something close to one.

The experimental setup is the same as above. MNIST binary, initial bandwidth $\gamma_0 = 1/784$ and by our previous findings, train split $p = 0.85$. We plot both the wiggled estimator's MSE with best bandwidth and that of refit standard estimator with exactly that MSE. We also plot the history of the best bandwidth found. The algorithm was run for more than 9 iterations but it converged and we cut the plot. See the results in fig. 3.

Summary:

- The algorithm converges and decreases the MSE as hoped for.
- Remarkably, the curves for wiggled and refit estimators are qualitatively the same! This gives hope for finding good theoretical guarantees for this method.
- The wiggled estimators reliably outperform the standard ones even though they are not fit on the full training set.

3. Conclusions and next steps

This paper proposes two new methods for leveraging the bandwidth in Laplacian kernel ridgeless regression. The results seem to indicate that there is potential for them to replace classical methods for optimizing the parameter (grid search, cross-validation) as they are expensive and don't scale well to big data. The methods have been demonstrated to improve over standard estimators in some cases even with nearly optimal bandwidth guess. However, much further experimentation is necessary and providing theoretical guarantees is thinkable as the recent progresses in the field have opened new doors.

Finally, I add a list of todos and questions for further research on the method.

- Bigger scale experiments are necessary to prove that the method really scales well (my computer couldn't do it) Also because the MNIST dataset have been shown in [4] to be predicted best by ridgeless Laplacian estimators. What happens with more noisy datasets?
- Given that we move away from the interpolating state (in a smooth and controlled manner) does wiggling compare to Tikhonov? What happens with the norm/complexity of a wiggled estimator compared to the standard one?
- As a mathematician I really want to leave the experimental world and analyze on paper providing risk bounds. There is in particular one phenomenon that I found by mistake and that I couldn't label. If it is a general rule, that would imply that a wiggled estimator "knows" about the the standard estimator and can be used to find its global risk minimum (always w.r.t. the bandwidth). See appendix A.
- Check the literature for any papers on these methods. I haven't found anything so far but my research was not at all exhaustive.

Appendix A. A strange effect

In one of my other experiments I used the inverse bandwidth convention. It is more or less the same plot as in fig. 1 just with $\gamma \to 1/\gamma$ so that I had to manually edit the bandwidth values to obtain readable results in the new scale. However, in that inverse, it appeared to me that the wiggled estimator "knew" perfectly well where the true optimal bandwidth of the standard estimator is. Indeed, fig. 4 shows extrema at exactlt that best point even if the base bandwidth is far away. I coudln't confirm that this is a big coincidence but I would like to investigate that more.

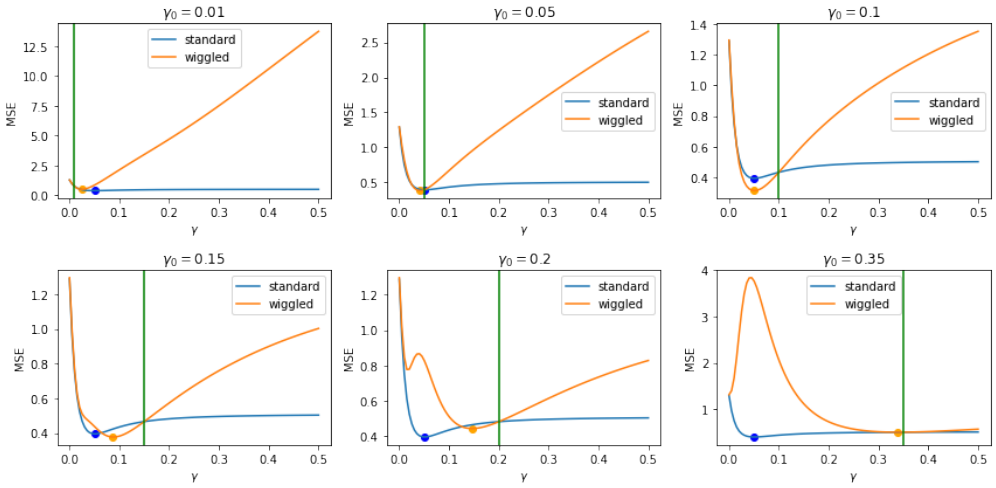

FIGURE 4. Strange: Cheap wiggled estimators seems to know the expensive standard ones perfectly well?

REFERENCES

[1] Belkin, Mikhail, et al. "Reconciling modern machine learning practice and the bias-variance trade-off." arXiv preprint arXiv:1812.11118 (2018).

[2] Bartlett, Peter L., et al. "Benign overfitting in linear regression." Proceedings of the National Academy of Sciences 117.48 (2020): 30063-30070.

[3] Belkin, Mikhail, Siyuan Ma, and Soumik Mandal. "To understand deep learning we need to understand kernel learning." International Conference on Machine Learning. PMLR, 2018.

[4] Liang, Tengyuan, and Alexander Rakhlin. "Just interpolate: Kernel "ridgeless" regression can generalize." The Annals of Statistics 48.3 (2020): 1329-1347.

[5] Geifman, Amnon, et al. "On the similarity between the laplace and neural tangent kernels." Advances in Neural Information Processing Systems 33 (2020): 1451-1461.

[6] Rakhlin, Alexander, and Xiyu Zhai. "Consistency of interpolation with Laplace kernels is a high-dimensional phenomenon." Conference on Learning Theory. PMLR, 2019.

[7] Rudi, Alessandro, Luigi Carratino, and Lorenzo Rosasco. "Falkon: An optimal large scale kernel method." Advances in neural information processing systems 30 (2017).