Efficient Hyperparameter Tuning for Large Scale Kernel Ridge Regression

Giacomo Meanti
MaLGa, DIBRIS, UniGe (IT)

Luigi Carratino
MaLGa, DIBRIS, UniGe (IT)

Ernesto De Vito
MaLGa, DIMA, UniGe (IT)

Lorenzo Rosasco
MaLGa, DIBRIS, UniGe (IT) – CBMM, MIT (USA) – IIT Genoa (IT)

Abstract

Kernel methods provide a principled approach to nonparametric learning. While their basic implementations scale poorly to large problems, recent advances showed that approximate solvers can efficiently handle massive datasets. A shortcoming of these solutions is that hyperparameter tuning is not taken care of, and left for the user to perform. Hyperparameters are crucial in practice and the lack of automated tuning greatly hinders efficiency and usability. In this paper, we work to fill in this gap focusing on kernel ridge regression based on the Nyström approximation. After reviewing and contrasting a number of hyperparameter tuning strategies, we propose a complexity regularization criterion based on a data dependent penalty, and discuss its efficient optimization. Then, we proceed to a careful and extensive empirical evaluation highlighting strengths and weaknesses of the different tuning strategies. Our analysis shows the benefit of the proposed approach, that we hence incorporate in a library for large scale kernel methods to derive adaptively tuned solutions.

1 INTRODUCTION

Learning from finite data requires fitting models of varying complexity to training data. The problem of finding the model with the right complexity is referred to as model selection in statistics and more broadly as hyperparameter tuning in machine learning. The problem is classical and known to be of utmost importance for machine learning algorithms to perform well in practice. The literature in statistics is extensive (Hastie et al., 2009), including a number of theoretical results (Arlot, 2007; Massart, 2007; Tsybakov, 2003). Hyperparameter (HP) tuning is also at the core of recent trends such as neural architecture search (Elsken et al., 2019) or AutoML (Hutter et al., 2018). In this paper, we consider the question of hyperparameter tuning in the context of kernel methods and specifically kernel ridge regression (KRR) (Smola and Schölkopf, 2000). Recent advances showed that kernel methods can be scaled to massive data-sets using approximate solvers (Chen et al., 2017; Ma and Belkin, 2019; Meanti et al., 2020). The latter take advantage of a number of ideas from optimization (Boyd and Vandenberghe, 2004) and randomized algorithms (El Alaoui and Mahoney, 2015), and exploit parallel computations with GPUs. While these solutions open up new possibilities for applying kernel methods, hyperparameter tuning is notably missing, ultimately hindering their practical use and efficiency. Indeed, available solutions which provide hyperparameter tuning are either limited to small data, or are restricted to very few hyperparameters (Pedregosa et al., 2011; Steinwart and Thomann, 2017; Suykens et al., 2002).

In this paper we work to fill in this gap. We consider approximate solvers based on the Nyström approximation and work towards an automated tuning of the regularization and kernel parameters, as well as the Nyström centers. On the one hand, we provide a careful review and extensive empirical comparison for a number of hyperparameter tuning strategies, while discussing their basic theoretical guarantees. On the other hand we propose, and provide an efficient implementation for, a novel criterion inspired by complexity regularization (Bartlett et al.,

Proceedings of the 25$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

2002) and based on a data-dependent bound. This bound treats separately the sources of variance due to the stochastic nature of the data. In practice, this results in better stability properties of the corresponding tuning strategy. As a byproduct of our analysis we complement an existing library for large-scale kernel methods with the possibility to adaptively tune a large number of hyperparameters. Code is available at `https://github.com/falkonml/falkon`.

In Section 2 we introduce the basic ideas behind empirical risk minimization and KRR, as well as hyperparameter tuning. In Section 3 we propose our new criterion, and discuss its efficient implementation in Section 4. In Section 5 we conduct a thorough experimental study and finally, in Section 6 we provide some concluding remarks.

2 BACKGROUND

We introduce the problem of learning a model's parameters, which leads to learning of the hyperparameters and then discuss various objective functions and optimization algorithms which have been proposed for the task.

2.1 Parameter and Hyperparameter Learning

Assume we are given a set of measurements $\{(x_i, y_i)\}_{i=1}^n \subset \mathcal{X} \times \mathcal{Y}$ related to each other by an unknown function $f^* : \mathcal{X} \to \mathcal{Y}$ and corrupted by some random noise ϵ_i with variance σ^2.

$$y_i = f^*(x_i) + \epsilon_i. \tag{1}$$

We wish to approximate the target function f^* using a model $f : \mathcal{X} \to \mathcal{Y}$ defined by a set of *parameters* which must be learned from the limited measurements at our disposal. In order for the learning procedure to succeed, one often assumes that f belongs to some hypothesis space $\mathcal{F}$, and this space typically depends on additional *hyperparameters* θ. Assume we are given a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$; we can learn a model by fixing the hyperparameters θ and minimizing the loss over the available training samples:

$$\hat{f}_\theta = \operatorname*{arg\,min}_{f \in \mathcal{F}_\theta} \sum_{i=1}^n \ell(f(x_i), y_i)$$

In this paper we are concerned with kernel ridge regression: a specific kind of model where the loss function is the squared loss $\ell(y, y') = \|y - y'\|^2$ and the hypothesis space is a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$. Associated to $\mathcal{H}$ is a kernel function $k_\gamma : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which depends on hyperparameters γ. To ensure that the minimization problem is well defined we must add a regularization term controlled by another hyperparameter λ:

$$\hat{f}_{\lambda,\gamma} = \operatorname*{arg\,min}_{f \in \mathcal{H}} \sum_{i=1}^n \|f(x_i) - y_i\|^2 + \lambda \|f\|_{\mathcal{H}}^2.$$

The solution to this minimization problem is unique (Caponnetto and De Vito, 2007), but is very expensive to compute requiring $O(n^3)$ operations and $O(n^2)$ space. An approximation to KRR considers a lower-dimensional subspace $\mathcal{H}_m \subset \mathcal{H}$ as hypothesis space, where $\mathcal{H}_m$ is defined from $m \ll n$ points $Z = \{z_j\}_{j=1}^m \subset \mathcal{X}$ (Williams and M. Seeger, 2001). While the inducing points Z (also known as Nyström centers) are often picked from the training set with different sampling schemes (Kumar et al., 2012), they can also be considered as hyperparameters. In fact this is common in sparse Gaussian Processes (GPs) and leads to models with a much smaller number of inducing points (Hensman, Fusi, et al., 2013; Hensman, Matthews, et al., 2015; Titsias, 2009). Minimizing the regularized error gives the unique solution

$$\hat{f}_{\lambda,Z,\gamma} = \sum_{j=1}^m \beta_j k_\gamma(\cdot, z_j), \quad \text{with}$$
$$\beta = (K_{nm}^\top K_{nm} + \lambda n K_{mm})^{-1} K_{nm}^\top Y \tag{2}$$

with $(K_{nm})_{i,j} = k_\gamma(x_i, z_j)$ and $(K_{mm})_{i,j} = k_\gamma(z_i, z_j)$. The Nyström KRR model (N-KRR) reduces the computational cost of finding the coefficients to $O(n\sqrt{n}\log n)$ when using efficient solvers (Ma and Belkin, 2019; Meanti et al., 2020; Rudi et al., 2017).

The ideal goal of hyperparameter optimization is to find a set of hyperparameters θ^* for which $\hat{f}_{\theta^*}$ minimizes the test error (over all unseen samples). By definition we cannot actually evaluate the test error: we must use the available data points. Naïvely one could think of minimizing the training error instead, but such a scheme inevitably chooses overly complex models which overfit the training set. Instead it is necessary to minimize a data-dependent criterion $\mathcal{L}$

$$\widehat{\theta} = \operatorname*{arg\,min}_{\theta} \mathcal{L}(\hat{f}_\theta)$$

such that complex models are penalized. In practice a common strategy for choosing $\mathcal{L}$ is for its expectation (with respect to the sampling of the data) to be equal to, or an upper bound of the test error. In the next section we will look at instances of $\mathcal{L}$ which appear in the literature and can be readily applied to N-KRR.

2.2 Objective Functions

Validation error A common procedure for HP tuning is to split the available n training samples into two

parts: a training set and a validation set. The first is used to learn a model $\hat{f}_\theta$ with fixed hyperparameters θ, while the validation set is used to estimate the performance of different HP configurations.

$$\mathcal{L}^{\mathrm{Val}}(\hat{f}_\theta) = \frac{1}{n_{\mathrm{val}}} \sum_{i=1}^{n_{\mathrm{val}}} \|\hat{f}_\theta(x_i^{\mathrm{val}}) - y_i^{\mathrm{val}}\|^2 \tag{3}$$

By using independent datasets for model training and HP selection, $\mathcal{L}^{\mathrm{Val}}$ will be an unbiased estimator of the test error and it can be proven that its minimizer is close to θ^* under certain assumptions (Arlot and Celisse, 2010). However, since $\hat{f}_\theta$ has been trained with $n_{\mathrm{tr}} < n$ samples, there is a small bias in the chosen hyperparameters (Varma and Simon, 2006). Furthermore the variance of the hold-out estimator is typically very high as it depends on a specific data split. Two popular alternatives which address this latter point are k-fold cross-validation (CV) which averages over k hold-out estimates and leave-one-out CV.

Leave-one-out CV and Generalized CV The LOOCV estimator is an average of the n estimators trained on all $n-1$ sized subsets of the training set and evaluated on the left out sample. The result is an almost unbiased estimate of the expected risk on the full dataset (Vapnik, 1998). For linear models a computational shortcut allows to compute the LOOCV estimator by training a single model on the whole dataset instead of n different ones (Cawley and Talbot, 2004). In particular in the case of N-KRR we can consider

$$\mathcal{L}^{\mathrm{LOOCV}}(\hat{f}_\theta) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i - \hat{f}_\theta(x_i)}{1 - H_{ii}} \right)^2, \tag{4}$$

where the so-called hat matrix H is $H = K_{nm}(K_{nm}^\top K_{nm} + \lambda n K_{mm})^{-1} K_{nm}$.

GCV is an approach proposed in Golub et al. (1979) to further improve LOOCV's computational efficiency and to make it invariant to data rotations:

$$\mathcal{L}^{\mathrm{GCV}}(\hat{f}_\theta) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i - \hat{f}_\theta(x_i)}{\frac{1}{n}\mathrm{Tr}(I - H)} \right)^2. \tag{5}$$

For GCV Cao and Golubev (2006) proved an oracle inequality which guarantees convergence to the neighborhood of θ^* when estimating λ for KRR.

Complexity regularization Complexity regularization, or covariance penalties (Efron, 2004; Mallows, 1973) are a general framework for expressing objective functions as the empirical error plus a penalty term to avoid overly complex models. For linear models the trace of the hat matrix acts as penalty against complexity. Applying these principles to N-KRR gives the objective

$$\mathcal{L}^{\mathrm{C-Reg}}(\hat{f}_{\lambda,Z,\gamma}) = \frac{1}{n}\|\hat{f}_{\lambda,Z,\gamma}(X) - Y\|^2 + \frac{2\sigma^2}{n}\mathrm{Tr}\Big((\widetilde{K} + n\lambda I)^{-1}\widetilde{K}\Big) \tag{6}$$

where $\widetilde{K} = K_{nm}K_{mm}^\dagger K_{nm}^\top$ (the Nyström kernel), and $A^\dagger$ denotes the Moore-Penrose inverse of matrix A. The first term can be interpreted as a proxy to the bias of the error, and the second as a variance estimate. For estimating λ in (N-)KRR, Arlot and Bach (2009) proved an oracle inequality if a precise estimate of the noise σ^2 is available.

Sparse GP Regression (Titsias, 2009) A different approach comes from a Bayesian perspective, where the equivalent of KRR is Gaussian Process Regression (GPR). Instead of estimating the test error, HP configurations are scored based on the "probability of a model given the data" (Rasmussen and Williams, 2006). A fully Bayesian treatment of the hyperparameters allows to write down their posterior distribution, from which the HP likelihood has the same form of the marginal likelihood in the model parameter's posterior. Hence maximizing the (log) marginal likelihood (MLL) with gradient-based methods is common practice in GPR.

Like with N-KRR, inducing points are used in GPR to reduce the computational cost, giving rise to models such as SoR, DTC, FiTC (Quiñonero-Candela and Rasmussen, 2005). Here we consider the SGPR model proposed in Titsias (2009) which treats the inducing points as variational parameters, and optimizes them along with the other HPs by maximizing a lower bound to the MLL. The objective to be minimized is

$$\mathcal{L}^{\mathrm{SGPR}}(\hat{f}_{\lambda,Z,\gamma}) = \log\Big|\widetilde{K} + n\lambda \boldsymbol{I}\Big| + Y^\top(\widetilde{K} + n\lambda \boldsymbol{I})^{-1}Y + \frac{1}{n\lambda}\mathrm{Tr}\Big(K - \widetilde{K}\Big). \tag{7}$$

The first term of Eq. (7) penalizes complex models, the second pushes towards fitting the training set well and the last term measures how well the inducing points approximate the full training set. Recently the approximate MLL was shown to converge to its exact counterpart (Burt et al., 2020), but we note that this does not guarantee convergence to the optimal hyperparameters.

2.3 Optimization Algorithms

In this section we describe three general approaches for the optimization of the objectives introduced above.

Grid search In settings with few hyperparameters the most widely used optimization algorithm is grid-search which tries all possible combinations from a predefined set, choosing the one with the lowest objective value at the end. Random search (Bergstra and Bengio, 2012) and adaptive grid search (used for SVMs in Steinwart and Thomann (2017)) improve on this basic idea, but they also become prohibitively costly with more than ~ 5 HPs as the number of combinations to be tested grows exponentially.

Black-box optimization A more sophisticated way to approach the problem is to take advantage of any smoothness in the objective. Sequential model-based optimization (SMBO) algorithms (Brochu et al., 2010; Shahriari et al., 2016; Snoek et al., 2012) take evaluations of the objective function as input, and fit a Bayesian *surrogate* model to such values. The surrogate can then be cheaply evaluated on the whole HP space to suggest the most promising HP values to explore. These algorithms do not rely on gradient information so they don't require the objective to be differentiable and can be applied for optimization of discrete HPs. However, while more scalable than grid search, black-box algorithms become very inefficient in high (i.e. > 100) dimensions.

Gradient-based methods Scaling up to even larger hyperparameter spaces requires exploiting the objective's local curvature. While the optimization problem is typically non-convex, gradient descent will usually reach a good local minimum. When the objective can be decomposed as a sum over the datapoints SGD can be used, which may provide computational benefits (e.g. the SVGP objective (Hensman, Fusi, et al., 2013) is optimized in mini-batches with SGD). In the context of KRR, gradient-based methods have been successfully used for HP optimization with different objective functions (Keerthi et al., 2007; M. W. Seeger, 2008). Recent extensions to gradient-based methods have been proposed for those cases when the trained model cannot be written in closed form. Either by unrolling the iterative optimization algorithm (Franceschi et al., 2017; Grazzi et al., 2020; Maclaurin et al., 2015), or by taking the model at convergence with the help of the implicit function theorem (Pedregosa, 2016; Rajeswaran et al., 2019), it is then possible to differentiate a simple objective (typically a hold-out error) through the implicitly defined trained model. This has proven to be especially useful for deep neural nets (Lorraine et al., 2020), but is unnecessary for N-KRR where the trained model can be easily written in closed form.

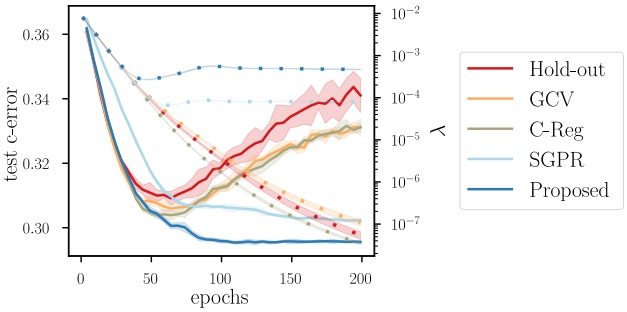

Figure 1: Test-error and penalty (λ) as a function of optimization epoch on the small-HIGGS dataset. $m = 100$ centers, d lengthscales and λ were optimized with equal initial conditions. The three unbiased proxy functions lead to overfitting, while SGPR and the proposed objective do not.

3 HYPERPARAMETER TUNING FOR NYSTRÖM KRR

The objectives introduced in the previous section can be applied to HP tuning for kernel methods. Always keeping in mind efficiency but also usability, our goal is to come up with an objective and associated optimization algorithm which: 1) can be used to tune the hyperparameters of Nyström kernel ridge regression including the inducing points and 2) can be computed efficiently, even for large scale problems.

To satisfy the first point, an algorithm of the first-order is needed since the inducing points are typically between a hundred and a few thousands (each point being of the same dimension as the data). Regarding the second point we found empirically that the unbiased objectives are prone to overfitting on certain datasets. An example of this behavior is shown in Figure 1 on a small subset of the HIGGS dataset. The first three objectives (Hold-out, GCV and C-Reg) are unbiased estimates of the test error, hence it is their variance which causes overfitting. To mitigate such possibility in our objective we may look into the different sources of variance: *hold-out* depends strongly on which part of the training set is picked for validation, *GCV* and *C-Reg* don't rely on data splitting but still suffer from the variance due to the random initial choice of inducing points.

We set out to devise a new objective in the spirit of complexity regularization, which is an upper bound on the test error. A biased estimate – which is therefore overpenalizing – will be more resistant to noise than an unbiased one (as was noted in Arlot (2007)), and we tailor our objective specifically to N-KRR in order to explicitly take into account the variance from inducing

point selection.

We base our analysis of the N-KRR error in the fixed design setting, where the points $x_i \in \mathcal{X}, i = (1, \dots, n)$ are assumed to be fixed, and the stochasticity comes from i.i.d. random variables $\epsilon_i, \dots, \epsilon_n$ such that $\mathbb{E}[\epsilon_i] = 0$ and $\mathbb{E}\left[\epsilon_i^\top \epsilon_i\right] = \sigma^2$. Denote the empirical error of an estimator $f \in \mathcal{H}$ as $\hat{L}(f) = n^{-1}\|f(X) - Y\|^2$ and the test error as $L(f) = n^{-1}\|f(X) - f^*(X)\|^2$ (recall f^* from Eq. (1)). Consider inducing points z_j and a subspace of $\mathcal{H}$: $\mathcal{H}_m = \operatorname{span}\{k_\gamma(z_1, \cdot), \dots, k_\gamma(z_m, \cdot)\}$, $m \ll n$, and let P be the projection operator with range $\mathcal{H}_m$. Denote the regularized empirical risk as $\hat{L}_\lambda(f) = \hat{L}(f) + \lambda\|f\|_{\mathcal{H}}^2$,

Assessing a particular hyperparameter configuration (λ, Z, γ) requires estimating the expected test error at the empirical risk minimizer trained with that configuration $\hat{f}_{\lambda,Z,\gamma}$; the optimal HPs then are found by $(\lambda, Z, \gamma)^* = \arg\min_{(\lambda,Z,\gamma)} L(\hat{f}_{\lambda,Z,\gamma})$. The following theorem gives an upper bound on the ideal objective; a full proof is available in Appendix A.

Theorem 1. *Under the assumptions of fixed-design regression we have that,*

$$\begin{aligned}\mathbb{E}\Big[L(\hat{f}_{\lambda,Z,\gamma})\Big] \leq & \frac{2\sigma^2}{n}\operatorname{Tr}\Big((\widetilde{K} + n\lambda I)^{-1}\widetilde{K}\Big) \\ & + \frac{2}{n\lambda}\operatorname{Tr}\Big(K - \widetilde{K}\Big)\mathbb{E}\Big[\hat{L}(f_{\lambda,\gamma})\Big] \\ & + 2\mathbb{E}\Big[\hat{L}(f_{\lambda,\gamma})\Big] \end{aligned} \tag{8}$$

Proof sketch. We decompose the test error expectation in the following manner

$$\begin{aligned}\mathbb{E}\Big[L(\hat{f}_{\lambda,Z,\gamma})\Big] \leq \mathbb{E}\Big[& \underbrace{L(\hat{f}_{\lambda,Z,\gamma}) - \hat{L}(\hat{f}_{\lambda,Z,\gamma})}_{①} \\ & + \underbrace{\hat{L}(\hat{f}_{\lambda,Z,\gamma}) + \lambda\|\hat{f}_{\lambda,Z,\gamma}\|_{\mathcal{H}}^2 - \hat{L}_\lambda(Pf_{\lambda,\gamma})}_{②} + \underbrace{\hat{L}_\lambda(Pf_{\lambda,\gamma})}_{③}\Big]\end{aligned}$$

by adding and subtracting $\hat{L}(\hat{f}_{\lambda,Z,\gamma})$, $\hat{L}_\lambda(Pf_{\lambda,\gamma})$ and summing the positive quantity $\lambda\|\hat{f}_{\lambda,Z,\gamma}\|_{\mathcal{H}}^2$. Since $\hat{f}_{\lambda,Z,\gamma}$ is the minimizer of $\hat{L}(\hat{f}_{\lambda,Z,\gamma}) + \lambda\|\hat{f}_{\lambda,Z,\gamma}\|_{\mathcal{H}}^2$ in the space $\mathcal{H}_m$ and since $Pf_{\lambda,\gamma} \in \mathcal{H}_m$, the second term is negative and can be discarded.

Term ① is the variance of N-KRR and can be computed exactly by noting that

$$\begin{aligned}\mathbb{E}\Big[\hat{L}(\hat{f}_{\lambda,Z,\gamma})\Big] &= \mathbb{E}\Big[n^{-1}\|\hat{f}_{\lambda,Z,\gamma}(X) - f^*(X) - \epsilon\|^2\Big] \\ &= \mathbb{E}\Big[L(\hat{f}_{\lambda,Z,\gamma})\Big] + \sigma^2 \\ &\quad - \frac{2}{n}\mathbb{E}\Big[\langle \hat{f}_{\lambda,Z,\gamma}(X) - f^*(X), \epsilon\rangle\Big]\end{aligned}$$

where the first part cancels and we can ignore σ^2 which is fixed and positive. Expanding the inner product and taking its expectation we are left with

$$\frac{2}{n}\mathbb{E}\Big[\langle \hat{f}_{\lambda,Z,\gamma}(X) - f^*(X), \epsilon\rangle\Big] = \frac{2\sigma^2}{n}\operatorname{Tr}\Big((\widetilde{K} + n\lambda I)^{-1}\widetilde{K}\Big)$$

which is the *effective dimension* or the *degrees of freedom* of the hypothesis space $\mathcal{H}_m$, times the noise variance σ^2.

Term ③ takes into account the difference between estimators in $\mathcal{H}$ and in $\mathcal{H}_m$. We begin by upper-bounding the regularized empirical error of $Pf_{\lambda,\gamma}$ with a first part containing the projection operator and a second term without P

$$\begin{aligned}&\mathbb{E}\Big[\hat{L}(Pf_{\lambda,\gamma}) + \lambda\|Pf_{\lambda,\gamma}\|_{\mathcal{H}}^2\Big] \\ &\quad \leq \mathbb{E}\left[\frac{2}{n}\|K^{1/2}(I - P)\|^2\|f_{\lambda,\gamma}\|^2 + 2\hat{L}_\lambda(f_{\lambda,\gamma})\right].\end{aligned}$$

Now $\|K^{1/2}(I - P)\|^2 \leq \operatorname{Tr}(K - \widetilde{K})$ the difference between full and approximate kernels, and $\|f_{\lambda,\gamma}\|^2 \leq \lambda^{-1}\hat{L}_\lambda(f_{\lambda,\gamma})$ which leads us to the desired upper bound. □

We now make two remarks on computing Eq. (8).

Remark 1. *(Computing $\mathbb{E}\Big[\hat{L}_\lambda(f_{\lambda,\gamma})\Big]$)* In the spirit of complexity regularization we can approximate this bias term by the empirical risk of N-KRR $\hat{L}_\lambda(\hat{f}_{\lambda,Z,\gamma})$, so that the final objective will consist of a data-fit term plus two complexity terms: the effective dimension and the Nyström approximation error.

Remark 2. *(Estimating σ^2)* Once again following the principle of overpenalizing rather than risking to overfit, we note that in binary classification the variance of Y is capped at 1 for numerical reasons, while for regression we can preprocess the data dividing Y by its standard deviation. Then according to Eq. (1) we must have that the label standard deviation is greater than the noise standard deviation hence $\hat{\sigma}^2 = 1 \geq \sigma^2$.

Our final objective then has a form which we can compute efficiently

$$\begin{aligned}\mathcal{L}^{\text{Prop}} = & \frac{2}{n}\operatorname{Tr}\Big((\widetilde{K} + n\lambda I)^{-1}\widetilde{K}\Big) \\ & + \frac{2}{n\lambda}\operatorname{Tr}\Big(K - \widetilde{K}\Big)\hat{L}_\lambda(\hat{f}_{\lambda,Z,\gamma}) \\ & + \frac{2}{n}\|\hat{f}_{\lambda,Z,\gamma}(X) - Y\|^2 + \lambda\|\hat{f}_{\lambda,Z,\gamma}\|_{\mathcal{H}}^2.\end{aligned} \tag{9}$$

We make two further remarks on the connections to the objectives of Section 2.2.

Remark 3. *(Similarities with complexity regularization)* $\mathcal{L}^{\text{Prop}}$ has a similar form to Eq. (6) with an extra term which corresponds to the variance introduced by the Nyström centers which we were aiming for (up to multiplication by the KRR bias).

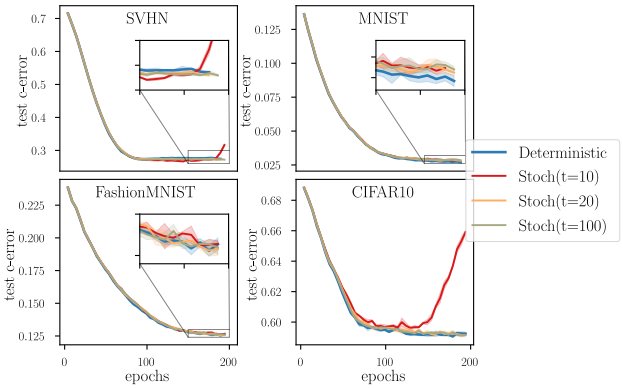

Figure 2: The effect of stochastic trace estimation. We plot the optimization curves of the exact objective $\mathcal{L}^{\text{Prop}}$ (*Deterministic*) and the approximated objectives with 10, 20 and 100 STE vectors. On the four datasets we optimized $m = 200$ centers, λ and γ.

Remark 4. *(Similarities with SGPR)* Eq. (9) shares many similarities with the SGPR objective: the log-determinant is replaced by the model's effective dimension – another measure of model complexity – and the term $\text{Tr}(K - \widetilde{K})$ is present in both objectives. Furthermore the data-fit term in $\mathcal{L}^{\text{SGPR}}$ is

$$\begin{aligned} &Y^\top (\widetilde{K} + n\lambda I)^{-1} Y \\ &\quad = \frac{1}{\lambda}(n^{-1}\|\hat{f}_{\lambda,Z,\gamma}(X) - Y\|^2 + \lambda\|\hat{f}_{\lambda,Z,\gamma}\|^2_{\mathcal{H}}) \\ &\quad = \frac{1}{\lambda}\hat{L}_\lambda(\hat{f}_{\lambda,Z,\gamma}) \end{aligned}$$

which is the same as in the proposed objective up to a factor λ^{-1}.

4 SCALABLE APPROXIMATIONS

Some practical considerations are needed to apply the objective of Eq. (9) to large-scale datasets – for which direct computation is not possible due to space or time constraints. We examine the terms comprising $\mathcal{L}^{\text{Prop}}$ and discuss their efficient computations. In Figure 2, we verify that the resulting approximation is close to the exact objective.

Starting with the last part of the optimization objective (the one which measures data-fit) we have that

$$\begin{aligned} &\|\hat{f}_{\lambda,Z,\gamma}(X) - Y\|^2 + \lambda\|\hat{f}_{\lambda,Z,\gamma}\|^2_{\mathcal{H}} \\ &= Y^\top (I - \underbrace{K_{nm}(\overbrace{K_{nm}^\top K_{nm} + n\lambda K_{mm}}^{B})^{-1} K_{nm}^\top}_{=\hat{f}_{\lambda,Z,\gamma}(X)})Y \end{aligned}$$

which can be computed quickly using a fast, memory-efficient N-KRR solver such as Falkon (Meanti et al., 2020) or EigenPro (Ma and Belkin, 2019). However we must also compute the objective's gradients with respect to all HPs, and since efficient solvers proceed by iterative minimization, such gradients cannot be trivially computed using automatic differentiation, indeed, it would be in principle possible to unroll the optimization loops and differentiate through them, the memory requirements for this operation would be too high for large datasets.

Efficient gradients A solution to compute the gradients efficiently is to apply the chain rule by hand until they can be expressed in terms of matrix vector products $(\nabla K)\boldsymbol{v}$ with K any kernel matrix (i.e. K_{nm} or K_{mm}) and $\boldsymbol{v}$ a vector. As an example the gradient of the data-fit term is

$$\begin{aligned} &\nabla(Y^\top K_{nm} B^{-1} K_{nm}^\top Y) = \\ &\quad 2Y^\top(\nabla K_{nm})B^{-1}K_{nm}^\top Y \\ &\quad - Y^\top K_{nm} B^{-1}(\nabla B)B^{-1}K_{nm}^\top Y \end{aligned}$$

where we can obtain all $B^{-1}K_{nm}^\top Y$ vectors via a non-differentiable N-KRR solver, and multiply them by the (differentiable) kernel matrices for which gradients are required. Computing these elementary operations is efficient, with essentially the same cost as the forward pass $K\boldsymbol{v}$, and can be done row-wise over K. Block-wise computations are essential for low memory usage since kernel matrices tend to be huge but kernel-vector products are small, and they allow trivial parallelization across compute units (CPU cores or GPUs). In many cases these operations can also be accelerated using KeOps (Charlier et al., 2021).

The remaining two terms of Eq. (9) are harder to compute. Note that in $\text{Tr}(K - \widetilde{K})$ we can often ignore $\text{Tr}(K)$ since common kernel functions are trivial when computed between a point and itself, but more in general it only requires evaluating the kernel function n times. We thus focus on

$$\text{Tr}\Big(\widetilde{K}\Big) = \text{Tr}\big(K_{nm}K_{mm}^\dagger K_{nm}^\top\big) \tag{10}$$

and on the effective dimension

$$\text{Tr}\Big((\widetilde{K} + \lambda I)^{-1}\widetilde{K}\Big) = \text{Tr}\big(K_{nm}B^{-1}K_{nm}^\top\big). \tag{11}$$

Both these terms are traces of huge $n \times n$ matrices. By their symmetry we can express them as squared norms reducing the space requirements to $n \times m$, but they still remain slow to compute: just the $K_{nm}^\top K_{nm}$ term costs more than training a N-KRR model with the Falkon solver.

Trace estimation A simple approximation can vastly improve the efficiency of computing Equations

(10), (11), and their gradients: stochastic trace estimation (STE). The Hutchinson estimator (Hutchinson, 1990) approximates $\mathrm{Tr}(A)$ by $\frac{1}{t}\sum_{i=1}^{t} r_i^\top A r_i$ where r_i are zero mean, unit standard deviation random vectors. We can use this to estimate Eq. (11) by running the Falkon solver with $R = [r_1, \dots, r_t]$ instead of the labels Y to obtain $(K_{nm}^\top K_{nm} + \lambda K_{mm})^{-1} K_{nm}^\top R$, then multiplying the result by $K_{nm}^\top R$ and normalizing by the number of stochastic estimators t. The same random vectors R can be used to compute $K_{nm}^\top R$ for Eq. (10), coupled with the Cholesky decomposition of K_{mm}. STE reduces the cost for both terms from $O(nm^2)$ to $O(nmt)$ which is advantageous since $t < m$. In Figure 4 we investigate whether the approximate objective matches the exact one, and how t affects the approximation. The observed behavior is that as few as 10 vectors are enough to approximate the full objective for a large part of the optimization run, but it can happen that such coarse approximation causes the loss to diverge. Increasing t to 20 solves the numerical issues, and on all the datasets tested we found $t = 20$ to be sufficient.

Alternatively, Eq. (10) can be approximated with a Nyström-like procedure: taking a random subsample of size p from the whole dataset, denote K_{pm} as the kernel matrix between such p points and the m Nyström centers; then

$$\mathrm{Tr}\big(K_{nm} K_{mm}^\dagger K_{nm}^\top\big) \approx \frac{n}{p}\mathrm{Tr}\big(K_{pm} K_{mm}^\dagger K_{pm}^\top\big)$$

which can be computed in $pm^2 + m^3$ operations. By choosing $p \sim m$ the runtime is then $O(m^3)$, which does not depend on the dataset size, and is more efficient than the STE approach. Unfortunately, this additional Nyström step cannot be effectively applied for computing Eq. (11) where the inversion of B is the most time-consuming step.

5 EXPERIMENTS

To validate the objective we are proposing for HP optimization of N-KRR models we ran a series of experiments aimed at answering the following questions:

1. Since our objective is an upper-bound on the test error, is the over-penalization acceptable, and what are its biases?
2. What is its behavior during gradient-based optimization: does it tend to overfit, does it lead to accurate models?
3. Does the approximation of Section 4 enable us to actually tune the hyperparameters on large datasets?

The first point is a sanity check: would the objective be a good proxy for the test error in a grid-search scenario

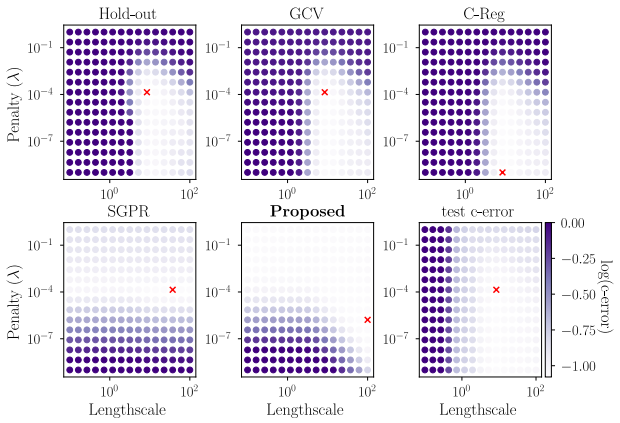

Figure 3: Effectiveness of test error proxies on a grid. The objective values (log transformed) are plotted at different λ, γ points for the *small-HIGGS* dataset. Lighter points indicate a smaller objective and hence a better hyperparameter configuration. The minimum of each objective is denoted by a cross.

over two hyperparameters (λ and γ with the RBF kernel). This doesn't necessarily transfer to larger HP spaces, but gives an indication of its qualitative behavior. In Figure 3 we compare 5 objective functions to the test error on such 2D grid. It is clear that the three functions which are unbiased estimators of the test error have very similar landscapes. Both SGPR and the proposed objective instead have the tendency to *overpenalize*: SGPR strongly disfavors low values of λ, while our objective prefers high λ and γ. This latter feature is associated with simpler models: a high γ produces smooth functions and a large λ restricts the size of the hypothesis.

We will see that the subdivision of objective functions into two distinct groups persists during optimization. However, in general it will not be true that the unbiased objectives produce models with lower test error than the overpenalized ones. The best performing method is going to depend on the dataset.

Small-scale optimization We used the exact formulas, along with automatic differentiation and the Adam optimizer to minimize the objectives on 20 datasets taken from the UCI repository, the LibSVM datasets, or in-house sources (more details on the datasets in Appendix B). We automated the optimization runs as much as possible to avoid having to set many meta-hyperparameters: fixed learning rate, the initial value for λ set to $1/n$ and the initial value for γ set with the median heuristic (Garreau et al., 2017). We used early stopping when the objective values started increasing. The results – shown in Figure 4 – confirm our previous observations: there

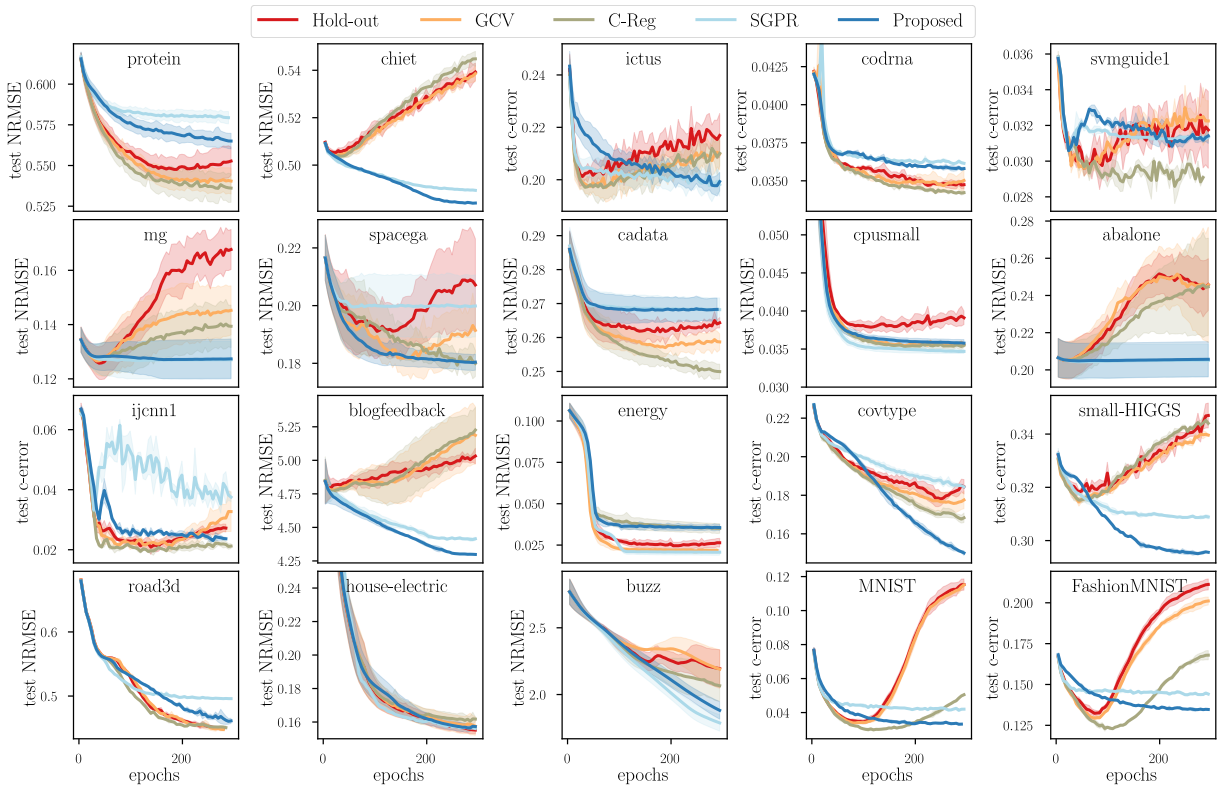

Figure 4: Empirical comparison of five objective functions for hyperparameter tuning. On each dataset we optimized $m = 100$ Nyström centers, a separate lengthscale for each dimension and λ for 200 epochs with a learning rate of 0.05 using the Adam optimizer. Also reported is the standard deviation from 5 runs of the same experiment with a different random seed. Each dataset has its own error metric. Labels of regression datasets were normalized to have unit standard deviation.

are some datasets (among which *small-HIGGS*, *buzz*, *house-electric*) on which the unbiased objectives overfit the training set while the proposed proxy function does not. In fact in some cases the hyperparameters found with our objective are much better than the ones found, for example, with the C-Reg objective. On the other hand, there is another group of datasets (e.g. *protein*, *energy* or *codrna*) where the extra bias of the proposed objective becomes detrimental as the optimization gets stuck into a suboptimal configuration with higher test error than what would be attainable with an unbiased objective.

Among the three unbiased objectives, hold-out clearly performs the worst. This is due to its high variance, and could be mitigated (at the expense of a higher computational cost) by using k-fold cross-validation. The GCV and C-Reg objectives perform similarly to each other in many cases. Especially in the image datasets however, GCV overfits more than C-Reg.

SGPR closely matches the proposed objective as it doesn't overfit. However, on several datasets it produces worse HPs than our objective displaying a larger bias. On the other hand there are other datasets for which the ranking is reversed, so there is no one clear winner. We must note however that the SGPR objective cannot be efficiently computed due to the log-determinant term, when datasets are large.

Large-scale optimization We tested the performance of the proposed objective with STE on three large-scale datasets, comparing it against two variational sparse GP solvers (Gardner et al., 2018; Matthews et al., 2017) which also learn a compact model with optimized inducing points and a classic N-KRR model with lots of randomly chosen centers trained with Falkon. Our tests are all performed in comparable conditions, details available in Appendix C. The results in Table 1 tell us that we can approach (but not quite reach) the performance – both in terms of speed and accuracy – of a very large model using a small fraction of the inducing points. They also support the conclusion that our objective is effective at optimizing a large number of hyperparameters, at least on par with methods in the GPR framework.

Table 1: Error and running time of kernel solvers on large-scale datasets. We compare our objective with two approximate GPR implementations and hand-tuned N-KRR (Falkon).

		$\mathcal{L}^{\text{Prop}}$	GPyTorch	GPFlow	Falkon
Flights $n \approx 10^6$	error	0.794	0.803	0.790	0.758
	time(s)	355	1862	1720	245
	m	5000	1000	2000	10^5
Flights-Cls $n \approx 10^6$	error	32.2	33.0	32.6	31.5
	time(s)	310	1451	627	186
	m	5000	1000	2000	10^5
Higgs $n \approx 10^7$	error	0.191	0.199	0.196	0.180
	time(s)	1244	3171	1457	443
	m	5000	1000	2000	10^5

6 CONCLUSIONS

In this paper, we improved the usability of large scale kernel methods proposing a gradient-based solution for tuning a large number of hyperparameters, on large problems. We incorporate this method into an existing library for large scale kernel methods with GPUs. We showed that it is possible to train compact Nyström KRR models if the centers are allowed to deviate from the training set, which can speed up inference by orders of magnitude. A future work will be to consider complex parametrized kernels which allow to improve the state of the art of kernel-based models on structured datasets such as those containing images or text.

Acknowledgments

The authors would like to thank the Anonymous Reviewers for their helpful comments on trace approximation. Lorenzo Rosasco acknowledges the financial support of the European Research Council (grant SLING 819789), the AFOSR projects FA9550-18-1-7009, FA9550-17-1-0390 and BAA-AFRL-AFOSR-2016-0007 (European Office of Aerospace Research and Development), the EU H2020-MSCA-RISE project NoMADS - DLV-777826, and the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF-1231216.

References

Arlot, S. (2007). "Resampling and Model selection". PhD thesis. University Paris-Sud (Orsay).

Arlot, S. and F. Bach (2009). "Data-driven calibration of linear estimators with minimal penalties". In: *NeurIPS 22*.

Arlot, S. and A. Celisse (2010). "A survey of cross-validation procedures for model selection". In: *Statistics Surveys* 4, pp. 40–79.

Bartlett, P. L., S. Boucheron, and G. Lugosi (2002). "Model Selection and Error Estimation". In: *Machine Learning* 48.

Bergstra, J. and Y. Bengio (2012). "Random Search for Hyper-Parameter Optimization". In: *J. Mach. Learn. Res.* 13, pp. 281–305.

Boyd, S. and L. Vandenberghe (2004). *Convex Optimization*. Cambridge University Press.

Brochu, E., V. M. Cora, and N. de Freitas (2010). *A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning*. arXiv: 1012.2599.

Burt, D. R., C. E. Rasmussen, and M. van der Wilk (2020). "Convergence of Sparse Variational Inference in Gaussian Processes Regression". In: *JMLR* 21, pp. 1–63.

Cao, Y. and Y. Golubev (2006). "On oracle inequalities related to smoothing splines". In: *Mathematical Methods of Statistic* 15.4.

Caponnetto, A. and E. De Vito (2007). "Optimal Rates for the Regularized Least-Squares Algorithm". In: *Foundations of Computational Mathematics* 7, pp. 331–368.

Cawley, G. C. and N. L. C. Talbot (2004). "Fast exact leave-one-out cross-validation of sparse least-squares support vector machines". In: *Neural Networks* 17.10, pp. 1467–1475.

Charlier, B., J. Feydy, J. A. Glaunès, F.-D. Collin, and G. Durif (2021). "Kernel Operations on the GPU, with Autodiff, without Memory Overflows". In: *JMLR* 22.74, pp. 1–6.

Chen, J., H. Avron, and V. Sindhwani (2017). "Hierarchically Compositional Kernels for Scalable Nonparametric Learning". In: *JMLR* 18.1, pp. 2214–2255.

Efron, B. (2004). "The estimation of prediction error: covariance penalties and cross-validation". In: *Journal of the American Statistical Association* 99.467, pp. 619–632.

El Alaoui, A. and M. W. Mahoney (2015). "Fast randomized kernel methods with statistical guarantees". In: *NeurIPS 28*.

Elsken, T., J. H. Metzen, and F. Hutter (2019). "Neural architecture search: A survey". In: *JMLR* 20.1, pp. 1997–2017.

Franceschi, L., M. Donini, P. Frasconi, and M. Pontil (2017). "Forward and Reverse Gradient-Based Hyperparameter Optimization". In: *ICML 34*.

Gardner, J. R., G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson (2018). "GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration". In: *NeurIPS 31*.

Garreau, D., W. Jitkrittum, and M. Kanagawa (2017). *Large sample analysis of the median heuristic*. arXiv: 1707.07269.

Golub, G. H., M. Heath, and G. Wahba (1979). "Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter". In: *Technometrics* 21.2, pp. 215–223.

Grazzi, R., L. Franceschi, M. Pontil, and S. Salzo (2020). "On the Iteration Complexity of Hypergradient Computation". In: *ICML 37*.

Hastie, T., R. Tibshirani, and J. Friedman (2009). *The Elements of Statistical Learning*. Springer, Berlin.

Hensman, J., N. Fusi, and N. D. Lawrence (2013). "Gaussian Processes for Big Data". In: *UAI*.

Hensman, J., A. Matthews, and Z. Ghahramani (2015). "Scalable variational Gaussian process classification". In: *AISTATS*. PMLR, pp. 351–360.

Hutchinson, M. F. (1990). "A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines". In: *Communications in Statistics-Simulation and Computation* 19.2, pp. 433–450.

Hutter, F., L. Kotthoff, and J. Vanschoren, eds. (2018). *Automated Machine Learning: Methods, Systems, Challenges*. Springer.

Keerthi, S. S., V. Sindhwani, and O. Chapelle (2007). "An Efficient Method for Gradient-Based Adaptation of Hyperparameters in SVM Models". In: *NeurIPS 19*.

Kumar, S., M. Mohri, and A. Talwalkar (2012). "Sampling Methods for the Nyström Method". In: *JMLR* 13, pp. 981–1006.

Lorraine, J., P. Vicol, and D. Duvenaud (2020). "Optimizing Millions of Hyperparameters by Implicit Differentiation". In: *AISTATS 23*.

Ma, S. and M. Belkin (2019). "Kernel machines that adapt to GPUs for effective large batch training". In: *Proceedings of the 2nd Conference on Machine Learning and Systems*.

Maclaurin, D., D. Duvenaud, and R. P. Adams (2015). "Gradient-Based Hyperparameter Optimization through Reversible Learning". In: *ICML 32*.

Mallows, C. L. (1973). "Some comments on C_p". In: *Technometrics* 15.4, pp. 661–675.

Massart, P. (2007). *Concentration inequalities and model selection*. Springer, Berlin.

Matthews, A., M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman (2017). "GPflow: A Gaussian process library using TensorFlow". In: *JMLR* 18.40, pp. 1–6.

Meanti, G., L. Carratino, L. Rosasco, and A. Rudi (2020). "Kernel methods through the roof: handling billions of points efficiently". In: *NeurIPS 34*.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). "Scikit-learn: Machine Learning in Python". In: *JMLR* 12, pp. 2825–2830.

Pedregosa, F. (2016). "Hyperparameter optimization with approximate gradient". In: *ICML 33*.

Quiñonero-Candela, J. and C. E. Rasmussen (2005). "A Unifying View of Sparse Approximate Gaussian Process Regression". In: *JMLR* 6.65, pp. 1939–1959.

Rajeswaran, A., C. Finn, S. M. Kakade, and S. Levine (2019). "Meta-Learning with Implicit Gradients". In: *NeurIPS 32*, pp. 113–124.

Rasmussen, C. E. and C. K. I. Williams (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Rudi, A., L. Carratino, and L. Rosasco (2017). "FALKON: An Optimal Large Scale Kernel Method". In: *NeurIPS 29*.

Seeger, M. W. (2008). "Cross-validation optimization for large scale structured classification kernel methods". In: *JMLR* 9, pp. 1147–1178.

Shahriari, B., K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas (2016). "Taking the Human Out of the Loop: A Review of Bayesian Optimization". In: *Proceedings of the IEEE* 104.1, pp. 148–175.

Smola, A. J. and B. Schölkopf (2000). "Sparse Greedy Matrix Approximation for Machine Learning". In: *Proceedings of the 17th Conference on Machine Learning*.

Snoek, J., H. Larochelle, and R. P. Adams (2012). "Practical Bayesian Optimization of Machine Learning Algorithms". In: *Neurips 25*.

Steinwart, I. and P. Thomann (2017). *liquidSVM: A fast and versatile SVM package*. arXiv: 1702.06899.

Suykens, J., T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle (2002). *Least Squares Support Vector Machines*. World Scientific.

Titsias, M. (2009). "Variational Learning of Inducing Variables in Sparse Gaussian Processes". In: *AISTATS 12*.

Tsybakov, A. B. (2003). "Optimal Rates of Aggregation". In: *Learning Theory and Kernel Machines*. Springer, pp. 303–313.

Vapnik, V. N. (1998). "Statistical Learning Theory". In: John Wiley & Sons. Chap. 10.

Varma, S. and R. Simon (2006). "Bias in error estimation when using cross-validation for model selection". In: *BMC Bioinformatics* 91.

Williams, C. K. I. and M. Seeger (2001). "Using the Nyström Method to Speed Up Kernel Machines". In: *NeurIPS 13*.

Supplementary Material: Efficient Hyperparameter Tuning for Large Scale Kernel Ridge Regression

A Full Derivation of a Complexity Penalty for N-KRR

We split the proof of Theorem 1 into a few intermediate steps: after introducing the relevant notation and definitions we give a few ways in which the Nyström estimator can be expressed, useful in different parts of the proof. Then we proceed with three more technical lemmas, used later on. We split the main proof into two parts to handle the two terms of the decomposition introduced in the main text of the paper: Lemma 6 for the sampling variance and Lemma 7 for the inducing point variance. Finally we restate Theorem 1 for completeness, whose proof follows directly from the two variance bounds.

A.1 Definitions

Using the same notation as in the main text we are given data $\{(x_i, y_i)\}_{i=1}^n \subset \mathcal{X} \times \mathcal{Y}$ such that

$$y_i = f^*(x_i) + \epsilon_i$$

where $f^* : \mathcal{X} \to \mathcal{Y}$ is an unknown function, and the noise ϵ_i is such that $\mathbb{E}[\epsilon_i] = 0, \mathbb{E}\left[\epsilon_i^2\right] = \sigma^2$. We let $\mathcal{H}$ be a RKHS and its subspace $\mathcal{H}_m = \operatorname{span}\{k_\gamma(z_1, \cdot), \dots, k_\gamma(z_m, \cdot)\}$ defined using the inducing points $\{z_j\}_{j=1}^m \subset \mathcal{X}$. We define a few useful operators, for vectors $\boldsymbol{v} \in \mathbb{R}^m$ and $\boldsymbol{w} \in \mathbb{R}^n$:

$$\begin{aligned}
&\widetilde{\Phi}_m : \mathcal{H} \to \mathbb{R}^m, && \widetilde{\Phi}_m = (k_\gamma(z_1, \cdot), \dots, k_\gamma(z_m, \cdot)) \\
&\widetilde{\Phi}_m^* : \mathbb{R}^m \to \mathcal{H}, && \widetilde{\Phi}_m^* \boldsymbol{v} = \sum_{j=1}^m \boldsymbol{v}_j k_\gamma(z_j, \cdot) \\
&\Phi : \mathcal{H} \to \mathbb{R}^n, && \Phi = (k_\gamma(x_1, \cdot), \dots, k_\gamma(x_n, \cdot)) \\
&\Phi^* : \mathbb{R}^n \to \mathcal{H}, && \Phi^* \boldsymbol{w} = \sum_{i=1}^n \boldsymbol{w}_j k_\gamma(x_i, \cdot).
\end{aligned}$$

Let $\Sigma : \mathcal{H} \to \mathcal{H} = \Phi^*\Phi$ be the covariance operator, and $K = \Phi\Phi^* \in \mathbb{R}^{n \times n}$ the kernel operator. Further define $K_{nm} = \Phi\widetilde{\Phi}_m^* \in \mathbb{R}^{n \times m}$, $K_{mm} = \widetilde{\Phi}_m\widetilde{\Phi}_m^* \in \mathbb{R}^{m \times m}$, and the approximate kernel $\widetilde{K} = K_{nm}K_{mm}^\dagger K_{nm}^\top \in \mathbb{R}^{n \times n}$. The SVD of the linear operator $\widetilde{\Phi}_m$ is

$$\widetilde{\Phi}_m = U\Lambda V^*$$

with $U : \mathbb{R}^k \to \mathbb{R}^m$, Λ the diagonal matrix of singular values sorted in non-decreasing order, $V : \mathbb{R}^k \to \mathcal{H}$, $k \le m$ such that $U^*U = I$, $V^*V = I$. The projection operator with range $\mathcal{H}_m$ is given by $P = VV^*$.

The KRR estimator $\hat{f}_{\lambda,\gamma}$ is defined as follows,

$$\hat{f}_{\lambda,\gamma} = \operatorname*{arg\,min}_{f \in \mathcal{H}} \frac{1}{n}\|f(X) - Y\|^2 + \lambda\|f\|_{\mathcal{H}}^2.$$

It can be shown (Caponnetto and De Vito, 2007) that $\hat{f}_{\lambda,\gamma}$ is unique and can be expressed in closed form as $\hat{f}_{\lambda,\gamma} = \Phi^*(K + n\lambda I)^{-1}Y$. In the proofs, we will also use the noise-less KRR estimator, denoted by $f_{\lambda,\gamma}$ and defined as,

$$f_{\lambda,\gamma} = \operatorname*{arg\,min}_{f \in \mathcal{H}} \frac{1}{n}\|f(X) - f^*(X)\|^2 + \lambda\|f\|_{\mathcal{H}}^2.$$

This estimator cannot be computed since we don't have access to f^*, but it is easy to see that

$$f_{\lambda,\gamma} = \Phi^*(K + n\lambda I)^{-1} f^*(X).$$

The N-KRR estimator, found by solving

$$\hat{f}_{\lambda,Z,\gamma} = \arg\min_{f\in\mathcal{H}_m} \frac{1}{n}\|f(X)-Y\|^2 + \lambda\|f\|^2_{\mathcal{H}}.$$

is unique, and takes the form (see Rudi, Camoriano, et al. (2015), Lemma 1)

$$\hat{f}_{\lambda,Z,\gamma} = (P\Sigma P + n\lambda I)^{-1} P\Phi^* Y$$

where P is the projection operator with range $\mathcal{H}_m$.

The estimator $\hat{f}_{\lambda,Z,\gamma}$ can be characterized in different ways as described next.

A.2 Preliminary Results on the Nyström estimator

The following lemma provides three different formulation of the Nyström estimator. We will use the notation $A^\dagger$ to denote the Moore-Penrose pseudo-inverse of a matrix A.

Lemma 1. *(Alternative forms of the Nyström estimator)*
The following equalities hold

$$\hat{f}_{\lambda,Z,\gamma} = (P\Sigma P + n\lambda I)^{-1} P\Phi^* Y \tag{12}$$
$$= V(V^*\Sigma V + n\lambda I)^{-1} V^*\Phi^* Y \tag{13}$$
$$= \widetilde{\Phi}^*_m (K_{nm}^\top K_{nm} + \lambda n K_{mm})^\dagger K_{nm}^\top Y \tag{14}$$

This Lemma is a restatement of results already found in the literature (e.g. in Rudi, Carratino, et al. (2017), Lemmas 2 and 3) which are condensed here with slightly different proofs.

Proof. Going from Eq. (12) to Eq. (13) consists in expanding $P = VV^*$ and applying the push-through identity

$$\begin{aligned}(P\Sigma P + n\lambda I)^{-1} P\Phi^* Y &= (VV^*\Sigma VV^* + n\lambda I)^{-1} VV^*\Phi^* Y \\ &= V(V^*\Sigma VV^*V + n\lambda I)^{-1} V^*\Phi^* Y \\ &= V(V^*\Sigma V + n\lambda I)^{-1} V^*\Phi^* Y.\end{aligned}$$

To go from Eq. (14) to Eq. (13) we split the proof into two parts. We first expand Eq. (14) rewriting the kernel matrices

$$\begin{aligned}\widetilde{\Phi}^*_m (K_{nm}^\top K_{nm} + \lambda n K_{mm})^\dagger K_{nm}^\top Y &= \widetilde{\Phi}^*_m (\widetilde{\Phi}_m \Phi^*\Phi \widetilde{\Phi}^*_m + n\lambda \widetilde{\Phi}_m \widetilde{\Phi}^*_m)^\dagger K_{nm}^\top Y \\ &= \widetilde{\Phi}^*_m (\widetilde{\Phi}_m (\Sigma + n\lambda I) \widetilde{\Phi}^*_m)^\dagger K_{nm}^\top Y.\end{aligned}$$

Then, we use some properties of the pseudo-inverse (Ben-Israel and Greville, 2001) to simplify $(\widetilde{\Phi}_m(\Sigma+n\lambda I)\widetilde{\Phi}^*_m)^\dagger$, in particular, using the SVD of $\widetilde{\Phi}_m$, write

$$(\underbrace{U\Lambda}_{F}\ \underbrace{V^*(\Sigma+n\lambda I)V}_{H}\ \underbrace{\Lambda U^*}_{F^*})^\dagger.$$

Since U has orthonormal columns, $F^\dagger = (U\Lambda)^\dagger = \Lambda^{-1}U^\dagger = \Lambda^{-1}U^*$. A consequence is that $(F^*)^\dagger = (\Lambda U^*)^\dagger = (\Lambda^{-1}U^*)^* = U\Lambda^{-1}$. Then we split $(FHF^*)^\dagger$ into the pseudo-inverse of its three components in two steps. Firstly $(HF^*)^\dagger = (F^*)^\dagger H^\dagger$ if $H^\dagger H = I$ and $(F^*)(F^*)^\dagger = I$:

1. Since $H = V^*(\Sigma + n\lambda I)V$ is invertible, $H^\dagger = H^{-1}$ and the first condition is verified.
2. $F^*(F^*)^\dagger = \Lambda U^* U\Lambda^{-1} = I$.

Also we have $(FHF^*)^\dagger = (HF^*)^\dagger F^\dagger$ if $F^\dagger F = I$ and $HF^*(HF^*)^\dagger = I$:

1. $F^\dagger F = \Lambda^{-1} U^* U \Lambda = I$,
2. $HF^*(HF^*)^\dagger = HF^*(F^*)^\dagger H^\dagger = HH^\dagger = I$.

The end result of this reasoning is that

$$(FHF^*)^\dagger = (F^*)^\dagger H^{-1} F^\dagger = U\Lambda^{-1}(V^*(\Sigma + n\lambda I)V)^{-1}\Lambda^{-1}U^*$$

and hence

$$\begin{aligned}\widetilde{\Phi}_m^*(K_{nm}^\top K_{nm} + \lambda n K_{mm})^\dagger K_{nm}^\top Y &= V\Lambda U^*(U\Lambda V^*(\Sigma + n\lambda I)V\Lambda U^*)^\dagger U\Lambda V^*\Phi^* Y \\ &= V\Lambda U^* U\Lambda^{-1}(V^*(\Sigma + n\lambda I)V)^{-1}\Lambda^{-1}U^*U\Lambda V^*\Phi^* Y \\ &= V(V^*\Sigma V + n\lambda I)^{-1}V^*\Phi^* Y\end{aligned}$$

□

Another useful equivalent form, for the Nyström estimator is given in the following lemma

Lemma 2. *Given the kernel matrices $K_{nm} \in \mathbb{R}^{n\times m}$, $K_{mm} \in \mathbb{R}^{m\times m}$, and the Nyström kernel $\widetilde{K} = K_{nm}K_{mm}^\dagger K_{nm}^\top \in \mathbb{R}^{n\times n}$, the following holds*

$$(\widetilde{K} + n\lambda I)^{-1}\widetilde{K} = K_{nm}(K_{nm}^\top K_{nm} + n\lambda K_{mm})^\dagger K_{nm}^\top \tag{15}$$

Proof. We state some facts about the kernel and image of the Nyström feature maps

$$\begin{aligned}(\ker \widetilde{\Phi}_m)^\perp &= \operatorname{span}\{k(z_1,\cdot),\dots,k(z_m,\cdot)\} = \operatorname{Im}\widetilde{\Phi}_m^* \\ (\ker \widetilde{\Phi}_m^*)^\perp &= \operatorname{Im}\widetilde{\Phi}_m = \operatorname{Im} K_{mm} = (\ker K_{mm})^\perp = W \subseteq \mathbb{R}^m.\end{aligned}$$

The space $\mathbb{R}^m$ is hence composed of $\mathbb{R}^m = W \oplus \ker\widetilde{\Phi}_m^*$. Take a vector $v \in \ker\widetilde{\Phi}_m^*$. We have that $\widetilde{\Phi}_m^* v = 0$, and $(K_{nm}^\top K_{nm} + n\lambda K_{mm})v = \widetilde{\Phi}_m(\Phi^*\Phi + n\lambda I)\widetilde{\Phi}_m^* v = 0$.

If instead $v \in W$, then $\widetilde{\Phi}_m(\Phi^*\Phi + n\lambda I)\widetilde{\Phi}_m^* v \in W$. Hence we have that

$$K_{nm}^\top K_{nm} + n\lambda K_{mm} : W \to W$$

and that K_{mm} is invertible when restricted to the subspace W, but also $K_{nm}^\top K_{nm} + n\lambda K_{mm}$ is invertible on W. Furthermore by the properties of the pseudo-inverse, we have that

$$(K_{nm}^\top K_{nm} + n\lambda K_{mm})(K_{nm}^\top K_{nm} + n\lambda K_{mm})^\dagger = P_W \tag{16}$$

with P_W the projector onto set W.

Furthermore we have the following equalities concerning the projection operator: $K_{mm}^\dagger K_{mm} = P_W$, as before; since $K_{nm} = \Phi\widetilde{\Phi}_m^*$, $K_{nm}P_W = \Phi\widetilde{\Phi}_m^* P_W = K_{nm}$ and similarly its transpose $K_{nm}^\top = \widetilde{\Phi}_m\Phi^*$ hence $P_W K_{nm}^\top = K_{nm}^\top$.

Using these properties we can say

$$\begin{aligned}K_{nm}K_{mm}^\dagger(K_{nm}^\top K_{nm} + n\lambda K_{mm}) &= K_{nm}K_{mm}^\dagger K_{nm}^\top K_{nm} + n\lambda K_{nm}K_{mm}^\dagger K_{mm} \\ &= K_{nm}K_{mm}^\dagger K_{nm}^\top K_{nm}P_W + n\lambda K_{nm}P_W \\ &= (K_{nm}K_{mm}^\dagger K_{nm}^\top + n\lambda I)K_{nm}P_W\end{aligned}$$

which implies that

$$(K_{nm}K_{mm}^\dagger K_{nm}^\top + n\lambda I)^{-1}K_{nm}K_{mm}^\dagger(K_{nm}^\top K_{nm} + n\lambda K_{mm}) = K_{nm}P_W.$$

Multiplying both sides by $(K_{nm}^\top K_{nm} + n\lambda K_{mm})^\dagger$, and using Eq. (16)

$$(K_{nm}K_{mm}^\dagger K_{nm}^\top + n\lambda I)^{-1}K_{nm}K_{mm}^\dagger P_W = K_{nm}P_W(K_{nm}^\top K_{nm} + n\lambda K_{mm})^\dagger \tag{17}$$

Hence we can write the left-hand side of our statement (Eq. (15)), and use the properties of projection P_W and Eq. (17) to get

$$\begin{aligned}(K_{nm}K_{mm}^{\dagger}K_{nm}^{\top}+n\lambda I)^{-1}K_{nm}K_{mm}^{\dagger}K_{nm}^{\top} &= (K_{nm}K_{mm}^{\dagger}K_{nm}^{\top}+n\lambda I)^{-1}K_{nm}K_{mm}^{\dagger}P_W K_{nm}^{\top}\\ &= K_{nm}P_W(K_{nm}^{\top}K_{nm}+n\lambda K_{mm})^{\dagger}K_{nm}^{\top}\\ &= K_{nm}(K_{nm}^{\top}K_{nm}+n\lambda K_{mm})^{\dagger}K_{nm}^{\top}\end{aligned}$$

which is exactly the right-hand side of our statement. □

Finally, the algebraic transformation given in the following lemma allows to go from a form which frequently appears in proofs involving the Nyström estimator ($\mathrm{Tr}((I-P)\Sigma)$) to a form which can easily be computed: the trace difference between the full and the Nyström kernel.

Lemma 3. *Let $\widetilde{\Phi}_m : \mathcal{H} \to \mathbb{R}^m$ be the kernel feature-map of the inducing points with SVD $\widetilde{\Phi}_m = U\Lambda V^*$, such that the projection operator onto $\mathcal{H}_m$ can be written $P = VV^*$. Also let $\widetilde{K} = K_{nm}K_{mm}^{\dagger}K_{nm}^{\top}$ be the Nyström kernel. Then the following equivalence holds*

$$\mathrm{Tr}((I-P)\Sigma) = \mathrm{Tr}\Big(K-\widetilde{K}\Big). \tag{18}$$

Proof. Note that we can write $K_{mm} = \widetilde{\Phi}_m\widetilde{\Phi}_m^* = U\Lambda V^*V\Lambda U^* = U\Lambda^2U^*$, which is a full-rank factorization since both $U\Lambda$ and $\Lambda U^{\top}$ are full-rank. Then we can use the formula for the full-rank factorization of the pseudoinverse (Ben-Israel and Greville (2001), Chapter 1, Theorem 5, Equation 24) to get

$$\begin{aligned}K_{mm}^{\dagger} &= (U\Lambda V^*V\Lambda U^*)^{\dagger} = (U\Lambda\Lambda U^*)^{\dagger}\\ &= U\Lambda(\Lambda U^*U\Lambda^2U^*U\Lambda)^{-1}\Lambda U^*\\ &= U\Lambda^{-2}U^*.\end{aligned}$$

Now we can prove the statement by expanding the left-hand side, and recalling $U^{\top}U = I$

$$\begin{aligned}\mathrm{Tr}((I-P)\Sigma) &= \mathrm{Tr}((I-VV^*)\Sigma)\\ &= \mathrm{Tr}\big((I-V(\Lambda U^*U\Lambda^{-2}U^*U\Lambda)V^*)\Phi^*\Phi\big)\\ &= \mathrm{Tr}\Big(\Phi(I-V\Lambda U^*(\widetilde{\Phi}_m\widetilde{\Phi}_m^*)^{\dagger}U\Lambda V^*)\Phi^*\Big)\\ &= \mathrm{Tr}\Big(\Phi\Phi^*-\Phi\widetilde{\Phi}_m^*(\widetilde{\Phi}_m\widetilde{\Phi}_m^*)^{\dagger}\widetilde{\Phi}_m\Phi^*\Big)\\ &= \mathrm{Tr}\big(K-K_{nm}K_{mm}^{\dagger}K_{nm}^{\top}\big) = \mathrm{Tr}\Big(K-\widetilde{K}\Big).\end{aligned}$$

□

The following two lemmas provide some ancillary results which are used in the proof of the main lemmas below.

Lemma 4. *Let P be the projection operator onto $\mathcal{H}_m$, and $f_{\lambda,\gamma}$ be the noise-less KRR estimator. Then the following bound holds*

$$\|Pf_{\lambda,\gamma}\|_{\mathcal{H}}^2 \leq \|f_{\lambda,\gamma}\|^2. \tag{19}$$

Proof. This is a simple application of the definition of operator norm, coupled with the fact that orthogonal projection operators have eigenvalues which are either 0 or 1 (hence their norm is at most 1).

$$\begin{aligned}\|Pf_{\lambda,\gamma}\|_{\mathcal{H}}^2 &\leq \|P\|^2\|f_{\lambda,\gamma}\|_{\mathcal{H}}^2\\ &\leq \|f_{\lambda,\gamma}\|_{\mathcal{H}}^2.\end{aligned}$$

□

Lemma 5. *Recall the notation $\hat{L}_\lambda(f) = n^{-1}\|f(X)-Y\|^2 + \lambda\|f\|_{\mathcal{H}}^2$, and let $f_{\lambda,\gamma}$ be the noise-less KRR estimator as before. Then the following statement holds:*

$$\|f_{\lambda,\gamma}\|_{\mathcal{H}}^2 \leq \mathbb{E}\left[\frac{\hat{L}_\lambda(f_{\lambda,\gamma})}{\lambda}\right] \tag{20}$$

where the expectation is taken with respect to the noise.

Proof. Recall that in the fixed design setting, given a fixed (i.e. not dependent on the label-noise) estimator, we always have

$$\mathbb{E}\Big[\hat{L}(f)\Big] = L(f) + \sigma^2$$

where σ^2 is the label-noise variance.

In our case, noting that $L(f_{\lambda,\gamma})$ is always non-negative

$$\begin{aligned}
\|f_{\lambda,\gamma}\|_{\mathcal{H}}^2 &= \frac{\lambda}{\lambda}\|f_{\lambda,\gamma}\|_{\mathcal{H}}^2 \\
&\leq \frac{L(f_{\lambda,\gamma}) + \lambda\|f_{\lambda,\gamma}\|_{\mathcal{H}}^2}{\lambda} \\
&\leq \frac{L(f_{\lambda,\gamma}) + \sigma^2 + \lambda\|f_{\lambda,\gamma}\|_{\mathcal{H}}^2}{\lambda} \\
&= \frac{\mathbb{E}\Big[\hat{L}_\lambda(f_{\lambda,\gamma})\Big]}{\lambda}.
\end{aligned}$$

□

A.3 Proof of the main Theorem

The proof of Theorem 1 starts from the error decomposition found in Section 3 which we report here:

$$\begin{aligned}
\mathbb{E}\Big[L(\hat{f}_{\lambda,Z,\gamma})\Big] \leq \mathbb{E}\Big[&\underbrace{L(\hat{f}_{\lambda,Z,\gamma}) - \hat{L}(\hat{f}_{\lambda,Z,\gamma})}_{①} \\
&+ \underbrace{\hat{L}(\hat{f}_{\lambda,Z,\gamma}) + \lambda\|\hat{f}_{\lambda,Z,\gamma}\|_{\mathcal{H}}^2 - \hat{L}_\lambda(Pf_{\lambda,\gamma})}_{②} + \underbrace{\hat{L}_\lambda(Pf_{\lambda,\gamma})}_{③}\Big]
\end{aligned}$$

and proceeds by bounding terms ① (see Lemma 6) and ③ (see Lemma 7). After the two necessary lemmas we restate the proof of the main theorem which now becomes trivial.

Lemma 6. *(Bounding the data-sampling variance)*
Denoting by $\hat{f}_{\lambda,Z,\gamma}$ the N-KRR estimator, the expected difference between its empirical and test errors can be calculated exactly:

$$\mathbb{E}\Big[L(\hat{f}_{\lambda,Z,\gamma}) - \hat{L}(\hat{f}_{\lambda,Z,\gamma})\Big] = \frac{2\sigma^2}{n}\mathrm{Tr}\Big((\widetilde{K} + n\lambda I)^{-1}\widetilde{K}\Big)$$

with σ^2 the noise variance and $\widetilde{K}$ the Nyström kernel.

Proof. For the sake of making the proof self-contained we repeat the reasoning of Section 3. Starting with the expectation of the empirical error we decompose it into the expectation of the test error minus an inner product term:

$$\begin{aligned}
\mathbb{E}\Big[\hat{L}(\hat{f}_{\lambda,Z,\gamma})\Big] &= \mathbb{E}\left[\frac{1}{n}\|\hat{f}_{\lambda,Z,\gamma}(X) - f^*(X) - \epsilon\|^2\right] \\
&= \mathbb{E}\Big[L(\hat{f}_{\lambda,Z,\gamma})\Big] + \sigma^2 - \frac{2}{n}\mathbb{E}\Big[\langle \hat{f}_{\lambda,Z,\gamma}(X) - f^*(X), \epsilon\rangle\Big].
\end{aligned}$$

The σ^2 term is fixed for optimization purposes, so we must deal with the inner-product. We use the form of $\hat{f}_{\lambda,Z,\gamma}$ from Eq. (14), Lemma 1, and $\mathbb{E}[\epsilon] = 0$, and to clean the notation we call $H := K_{nm}(K_{nm}^\top K_{nm} + n\lambda K_{mm})^\dagger K_{nm}^\top$:

$$\begin{aligned}\frac{2}{n}\mathbb{E}\Big[\langle \hat{f}_{\lambda,Z,\gamma}(X) - f^*(X), \epsilon\rangle\Big] &= \frac{2}{n}\mathbb{E}[\langle H(f^*(X)+\epsilon) - f^*(X), \epsilon\rangle] \\ &= \frac{2}{n}\mathbb{E}\big[\epsilon^\top H\epsilon\big] = \frac{2\sigma^2}{n}\mathrm{Tr}(H),\end{aligned}$$

and using Lemma 2 H can be expressed as $(\widetilde{K} + n\lambda I)^{-1}\widetilde{K}$.

Going back to the original statement we have

$$\begin{aligned}\mathbb{E}\Big[L(\hat{f}_{\lambda,Z,\gamma}) - \hat{L}(\hat{f}_{\lambda,Z,\gamma})\Big] &= \mathbb{E}\Big[L(\hat{f}_{\lambda,Z,\gamma}) - L(\hat{f}_{\lambda,Z,\gamma}) + \frac{2\sigma^2}{n}\mathrm{Tr}\Big((\widetilde{K} + n\lambda I)^{-1}\widetilde{K}\Big)\Big] \\ &= \frac{2\sigma^2}{n}\mathrm{Tr}\Big((\widetilde{K} + n\lambda I)^{-1}\widetilde{K}\Big)\end{aligned}$$

□

Lemma 7. *(Bounding the Nyström variance)*
Under the fixed-design assumptions, denote by P the orthogonal projector onto space $\mathcal{H}_m$, by $\hat{L}_\lambda(f)$ the regularized empirical risk of estimator f, and by $f_{\lambda,\gamma} \in \mathcal{H}$ the noise-less KRR estimator. Then the following upper-bound holds

$$\mathbb{E}\Big[\hat{L}_\lambda(Pf_{\lambda,\gamma})\Big] \leq \frac{2}{n\lambda}\mathrm{Tr}\Big(K - \widetilde{K}\Big)\mathbb{E}\Big[\hat{L}_\lambda(f_{\lambda,\gamma})\Big] + 2\mathbb{E}\Big[\hat{L}_\lambda(f_{\lambda,\gamma})\Big]. \tag{21}$$

Proof. Note that for estimators $f \in \mathcal{H}$ we can always write $f(X) = \Phi f$. Hence for the projected KRR estimator we use that $(Pf_{\lambda,\gamma})(X) = \Phi P f_{\lambda,\gamma}$. We start by rewriting the left hand side to obtain a difference between projected and non-projected terms:

$$\begin{aligned}\mathbb{E}\Big[\hat{L}(Pf_{\lambda,\gamma}) + \lambda\|Pf_{\lambda,\gamma}\|_{\mathcal{H}}^2\Big] &= \mathbb{E}\left[\frac{1}{n}\|\Phi P f_{\lambda,\gamma} - f^*(X) - \epsilon\|^2 + \lambda\|Pf_{\lambda,\gamma}\|_{\mathcal{H}}^2\right] \\ &= \mathbb{E}\left[\frac{1}{n}\|\Phi P f_{\lambda,\gamma} - f^*(X)\|^2 + \frac{1}{n}\|\epsilon\|^2 + \lambda\|Pf_{\lambda,\gamma}\|_{\mathcal{H}}^2\right] \\ &= \mathbb{E}\left[\frac{1}{n}\|\Phi P f_{\lambda,\gamma} - \Phi f_{\lambda,\gamma} + \Phi f_{\lambda,\gamma} - f^*(X)\|^2 + \frac{1}{n}\|\epsilon\|^2 + \lambda\|Pf_{\lambda,\gamma}\|_{\mathcal{H}}^2\right] \\ &\leq \mathbb{E}\left[\frac{2}{n}\|\Phi P f_{\lambda,\gamma} - \Phi f_{\lambda,\gamma}\|^2 + \frac{2}{n}\|\Phi f_{\lambda,\gamma} - f^*(X)\|^2 + \frac{2}{n}\|\epsilon\|^2 + 2\lambda\|Pf_{\lambda,\gamma}\|_{\mathcal{H}}^2\right]\end{aligned}$$

where we used the fact that $\mathbb{E}[\epsilon] = 0$, and the triangle inequality in the last step.

By Lemma 4, and the definition of $\mathbb{E}\Big[\hat{L}(f)\Big]$ we have that

$$\mathbb{E}\left[\frac{2}{n}\|\Phi f_{\lambda,\gamma} - f^*(X)\|^2 + \frac{2}{n}\|\epsilon\|^2 + 2\lambda\|Pf_{\lambda,\gamma}\|_{\mathcal{H}}^2\right] \leq 2\mathbb{E}\Big[\hat{L}(f_{\lambda,\gamma})\Big].$$

Next we use again the definition of operator norm to deal with the difference between projected and non-projected noise-less KRR estimators:

$$\begin{aligned}\mathbb{E}\left[\frac{2}{n}\|\Phi P f_{\lambda,\gamma} - \Phi f_{\lambda,\gamma}\|^2\right] &= \frac{2}{n}\|\Phi(P-I)f_{\lambda,\gamma}\|^2 \\ &\leq \frac{2}{n}\|\Phi(I-P)\|^2\|f_{\lambda,\gamma}\|^2.\end{aligned}$$

The first part of this latter term is

$$\|\Phi(I-P)\|^2 = \|(I-P)\Phi^\top\Phi(I-P)\| \leq \mathrm{Tr}\big((I-P)\Phi^\top\Phi\big) = \mathrm{Tr}((I-P)\Sigma)$$

since the trace norm controls the operator norm, and using the cyclic property of the trace and the idempotence of the projection operator $I - P$. By Lemma 3 we have that $\|\Phi(I-P)\|^2 \leq \mathrm{Tr}\Big(K - \widetilde{K}\Big)$. For the second part we use Lemma 5 so that

$$\|f_{\lambda,\gamma}\|^2 \leq \mathbb{E}\left[\frac{\hat{L}_\lambda(f_{\lambda,\gamma})}{\lambda}\right]$$

which concludes the proof. □

We now have all the ingredients to prove Theorem 1 which we restate below for the reader.

Theorem. *(Restated from the main text)*
Under the assumptions of fixed-design regression we have that,

$$\begin{aligned}\mathbb{E}\Big[L(\hat{f}_{\lambda,Z,\gamma})\Big] \leq & \frac{2\sigma^2}{n}\mathrm{Tr}\Big((\widetilde{K}+\lambda I)^{-1}\widetilde{K}\Big) \\ & + \frac{2}{n\lambda}\mathrm{Tr}\Big(K-\widetilde{K}\Big)\mathbb{E}\Big[\hat{L}(f_{\lambda,\gamma})\Big] \\ & + 2\mathbb{E}\Big[\hat{L}(f_{\lambda,\gamma})\Big] \end{aligned} \tag{22}$$

Proof. The decomposition is the same:

$$\begin{aligned}\mathbb{E}\Big[L(\hat{f}_{\lambda,Z,\gamma})\Big] \leq \mathbb{E}\Big[& \underbrace{L(\hat{f}_{\lambda,Z,\gamma}) - \hat{L}(\hat{f}_{\lambda,Z,\gamma})}_{①} \\ & + \underbrace{\hat{L}(\hat{f}_{\lambda,Z,\gamma}) + \lambda\|\hat{f}_{\lambda,Z,\gamma}\|^2_{\mathcal{H}} - \hat{L}_\lambda(Pf_{\lambda,\gamma})}_{②} + \underbrace{\hat{L}_\lambda(Pf_{\lambda,\gamma})}_{③}\Big]\end{aligned}$$

where ② ≤ 0. We may then use Lemma 6 for term ① and Lemma 7 for term ③ to obtain

$$\mathbb{E}\Big[L(\hat{f}_{\lambda,Z,\gamma})\Big] \leq \frac{2\sigma^2}{n}\mathrm{Tr}\Big((\widetilde{K}+n\lambda I)^{-1}\widetilde{K}\Big) + \frac{2}{n\lambda}\mathrm{Tr}\Big(K-\widetilde{K}\Big)\mathbb{E}\Big[\hat{L}_\lambda(f_{\lambda,\gamma})\Big] + 2\mathbb{E}\Big[\hat{L}_\lambda(f_{\lambda,\gamma})\Big].$$

□

B Datasets

We used a range of datasets which represent a wide spectrum of scenarios for which kernel learning can be used. They can be divided into three groups: medium sized unstructured datasets (both for regression and binary classification), medium sized image recognition datasets (multiclass classification) and large unstructured datasets (classification and regression). We applied similar preprocessing steps to all datasets (namely standardization of the design matrix, standardization of the labels for regression datasets, one-hot encoding of the labels for multiclass datasets). When an agreed-upon test-set existed we used it (e.g. for MNIST), otherwise we used random 70/30 or 80/20 train/test set splits, with each experiment repetition using a different split. Below we provide more details about the datasets used, grouping several of them together if the same procedures apply. The canonical URLs at which the datasets are available, along with their detailed dimensions and training/test splits are shown in Table 2

The error metrics used are dataset-dependent, and outlined below. For regression problems we use the RMSE, defined as $\sqrt{n^{-1}\sum_{i=1}^n (y_i - \hat{f}(x_i))^2}$ and its normalized version the NRMSE:

$$NRMSE: \left|\frac{\sqrt{\frac{1}{n}\sum_{i=1}^n (y_i - \hat{f}(x_i))^2}}{\frac{1}{n}\sum_{i=1}^n y_i}\right|.$$

For classification problems we use the fraction of misclassified examples (c-error), and the area under the curve (AUC) metric.

SpaceGA, Abalone, MG, CpuSmall, Energy Small regression datasets between 1385 (MG) and 8192 (CpuSmall) samples, label standardization is performed and error is measured as NRMSE. The predictor matrix is also standardized.

Road3D, Buzz, Protein, HouseElectric, BlogFeedback Regression datasets of medium to large size from the UCI ML repositories. We used label standardization for Road3D, BlogFeedback, Buzz and Protein, and an additional log transformation for HouseElectric. Measured error is NRMSE. The predictor matrix is standardized.

MNIST, FashionMNIST, SVHN, CIFAR-10 Four standard image recognition datasets. Here the labels are one-hot encoded (all datasets have 10 classes), and the design matrix is normalized in the 0-1 range. Standard train/test splits are used.

Chiet A time-series dataset for short-term wind prediction. The labels and predictors are standardized, and the error is measured with the NRMSE. A fixed split in time is used.

Ictus A dataset simulating brain MRI. Predictors are standardized and a random 80/20 split is used.

Cod-RNA, SVMGuide1, IJCNN1, CovType Four datasets for binary classification ranging between approximately 3000 points for SvmGuide1 and 5×10^5 points for CovType. The design matrix is standardized while the labels are -1 and $+1$.

Higgs, SmallHiggs HIGGS is a very large binary classification dataset from high energy physics. We took a small random subsample to generate the SmallHiggs dataset, which has predefined training and test sets. The design matrix is normalized by the features' variance. For the HIGGS dataset we measure the error as 1 minus the AUC.

Flights, Flights-Cls A regression dataset found in the literature (Hensman, Durrande, et al., 2017; Hensman, Fusi, et al., 2013) which can also be used for binary classification by thresholding the target at 0.

C Experiment Details

All experiments were run on a machine with a single NVIDIA Quadro RTX 6000 GPU, and 256GB of RAM. The details of all hyperparameters and settings required to reproduce our experiments are provided below. Relevant code is available in the repository at `https://github.com/falkonml/falkon`.

C.1 Small scale experiments

We ran the small scale experiments by optimizing the exact formulas for all objectives, computed with Cholesky decompositions and solutions to triangular systems of equations. We used the Adam optimizer with default settings and ran it for 200 epochs with a fixed learning rate of 0.05. We optimized $m = 100$ inducing points initialized to the a random data subset, used the Gaussian kernel with a separate length-scale for each data-dimension (the initialization using the median heuristic was the same for each dimension), and the amount of regularization λ which was initialized to $1/n$. The validation set size (for the *Hold-out* objective) was fixed to 60% of the full training data. While this may seem large, in our setting the size of the hyperparameter space (in first approximation $m \times d$) is larger than the number of model parameters ($m \times o$ where o is the dimension of the target space $\mathcal{Y}$, most commonly $o = 1$).

C.2 Large scale experiments

We ran the large-scale experiments for just the $\mathcal{L}^{\text{Prop}}$ objective, while the other performance numbers in Table 1 are taken from Meanti et al. (2020). For our objective we again used the Adam optimizer. For the Flights and Higgs dataset we trained with learning rate 0.05 for 20 epochs, while we trained Flights-Cls with a smaller learning rate of 0.02 for 10 epochs. We used the Gaussian kernel with a single length-scale, initialized as in (Meanti et al., 2020) (Flights $\sigma_0 = 1$, Flights-Cls $\sigma_0 = 1$, Higgs $\sigma_0 = 4$) and $\lambda_0 = 1/n$. We used $t = 20$ stochastic trace

Table 2: Key details on the datasets used.

	n	d	train/test	error
SpaceGA	3107	6	70%/30%	NRMSE
Abalone	4177	8	70%/30%	NRMSE
MG	1385	6	70%/30%	NRMSE
CpuSmall	8192	12	70%/30%	NRMSE
Energy	768	8	80%/20%	NRMSE
Road3D	434 874	3	70%/30%	RMSE
Buzz	2 049 280	11	70%/30%	RMSE
Protein	45 730	9	80%/20%	NRMSE
BlogFeedback	60 021	280	52 397/7624	RMSE
MNIST	70 000	784	60 000/10 000	10 class c-error
FashionMNIST	70 000	784	60 000/10 000	10 class c-error
SVHN	99 289	1024	73 257/26 032	10 class c-error
CIFAR-10	60 000	1024	50 000/10 000	10 class c-error
Chiet	34 059	144	26 227/7832	NRMSE
Ictus	29 545	992	80%/20%	binary c-error
Cod-RNA	331 152	8	59 535/271 617	binary c-error
SVMGuide1	7089	4	3089/4000	binary c-error
IJCNN1	141 691	22	49 990/91 701	binary c-error
CovType	581 012	54	70%/30%	binary c-error
SmallHiggs	30 000	28	10 000/20 000	binary c-error
Higgs	1.1×10^7	20	80%/20%	1 - AUC
Flights	5.93×10^6	8	66%/34%	MSE
Flights-Cls	5.93×10^6	8	5 829 413/100 000	binary c-error

estimation vectors for all three experiments, sampling them from the standard Gaussian distribution. The STE vectors were kept fixed throughout optimization. The conjugate gradient tolerance for the Falkon solver was set to 5×10^{-4} for Flights-Cls, and 1×10^{-3} for Flights and Higgs (a higher tolerance corresponds to longer training time), while we always capped the number of Falkon iterations to 100.

References

Ben-Israel, A. and T. N. E. Greville (2001). *Generalized Inverses: Theory and Applications*. 2nd ed. Springer.

Caponnetto, A. and E. De Vito (2007). "Optimal Rates for the Regularized Least-Squares Algorithm". In: *Foundations of Computational Mathematics* 7, pp. 331–368.

Hensman, J., N. Durrande, and A. Solin (2017). "Variational Fourier Features for Gaussian Processes". In: *JMLR* 18.1, pp. 5537–5588.

Hensman, J., N. Fusi, and N. D. Lawrence (2013). "Gaussian Processes for Big Data". In: *UAI*.

Meanti, G., L. Carratino, L. Rosasco, and A. Rudi (2020). "Kernel methods through the roof: handling billions of points efficiently". In: *NeurIPS 34*.

Rudi, A., R. Camoriano, and L. Rosasco (2015). "Less is More: Nyström Computational Regularization". In: *NeurIPS 28*.

Rudi, A., L. Carratino, and L. Rosasco (2017). "FALKON: An Optimal Large Scale Kernel Method". In: *NeurIPS 29*.