JMLR: Workshop and Conference Proceedings vol 40:1–29, 2015

On the Complexity of Learning with Kernels

Nicolò Cesa-Bianchi NICOLO.CESA-BIANCHI@UNIMI.IT
Università degli Studi di Milano, Italy

Yishay Mansour MANSOUR@TAU.AC.IL
Microsoft Research and Tel-Aviv University, Israel

Ohad Shamir OHAD.SHAMIR@WEIZMANN.AC.IL
Weizmann Institute of Science, Israel

Abstract

A well-recognized limitation of kernel learning is the requirement to handle a kernel matrix, whose size is quadratic in the number of training examples. Many methods have been proposed to reduce this computational cost, mostly by using a subset of the kernel matrix entries, or some form of low-rank matrix approximation, or a random projection method. In this paper, we study lower bounds on the error attainable by such methods as a function of the number of entries observed in the kernel matrix or the rank of an approximate kernel matrix. We show that there are kernel learning problems where no such method will lead to non-trivial computational savings. Our results also quantify how the problem difficulty depends on parameters such as the nature of the loss function, the regularization parameter, the norm of the desired predictor, and the kernel matrix rank. Our results also suggest cases where more efficient kernel learning might be possible.

1. Introduction

We consider the well-known problem of kernel learning (see, e.g., Scholkopf and Smola (2001)), where given a training set of labeled examples $\{(\boldsymbol{x}_t, y_t)\}_{t=1}^m$ from a product domain $\mathcal{X} \times \mathcal{Y}$, our goal is to find a linear predictor $\boldsymbol{w}$ in a reproducing kernel Hilbert space which minimizes the average loss, possibly with some regularization. Formally, our goal is to solve

$$\min_{\boldsymbol{w} \in \mathcal{W}} \frac{1}{m} \sum_{t=1}^{m} \ell(\langle \boldsymbol{w}, \psi(\boldsymbol{x}_t) \rangle, y_t) + \frac{\lambda}{2} \|\boldsymbol{w}\|^2 , \tag{1}$$

where $\mathcal{W}$ is a convex subset of some reproducing kernel Hilbert space $\mathcal{H}$, $\psi : \mathcal{X} \mapsto \mathcal{H}$ is a feature mapping to the Hilbert space, ℓ is a loss function convex in its first argument, and $\lambda \geq 0$ is a regularization parameter. For example, in the standard formulation of Support Vector Machines, we take ℓ to be the hinge loss, pick some $\lambda > 0$, and let $\mathcal{W}$ be the entire Hilbert space. Alternatively, one can also employ hard regularization, e.g., setting $\lambda = 0$ and taking $\mathcal{W} = \{\boldsymbol{w} : \|\boldsymbol{w}\| \leq R\}$.

It is well-known that even if $\mathcal{H}$ is high or infinite dimensional, we can solve (1) in polynomial time, provided there is an efficiently computable *kernel function* k such that $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \psi(\boldsymbol{x}), \psi(\boldsymbol{x}') \rangle$. The key insight is provided by the representer theorem, which implies that an optimum of (1) exists in the span of $\psi(\boldsymbol{x}_1), \ldots, \psi(\boldsymbol{x}_m)$. Therefore, instead of optimizing over $\boldsymbol{w}$, we can optimize over a coefficient vector $\boldsymbol{\alpha}$, which implicitly specifies a predictor via $\boldsymbol{w}(\boldsymbol{\alpha}) =$

$\sum_{j=1}^{m} \alpha_j \psi(\boldsymbol{x}_j)$. In this case, (1) reduces to

$$\min_{\boldsymbol{\alpha}\,:\,\boldsymbol{w}(\boldsymbol{\alpha})\in\mathcal{W}} \frac{1}{m}\sum_{t=1}^{m} \ell\left(\sum_{j=1}^{m} \alpha_j \left\langle \psi(\boldsymbol{x}_j), \psi(\boldsymbol{x}_t)\right\rangle, y_t\right) + \frac{\lambda}{2}\|\boldsymbol{w}(\boldsymbol{\alpha})\|^2 .$$

Defining the $m \times m$ kernel matrix $K_{i,j} = \langle \psi(\boldsymbol{x}_i), \psi(\boldsymbol{x}_j)\rangle = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, we can re-write the above as

$$\min_{\boldsymbol{\alpha}\,:\,\boldsymbol{w}(\boldsymbol{\alpha})\in\mathcal{W}} \frac{1}{m}\sum_{t=1}^{m} \ell\left(\boldsymbol{\alpha}^\top K \boldsymbol{e}_t, y_t\right) + \frac{\lambda}{2}\boldsymbol{\alpha}^\top K \boldsymbol{\alpha} . \tag{2}$$

This is a convex problem, which can generally be solved in polynomial time. The resulting $\boldsymbol{\alpha}$ implicitly defines the linear predictor $\boldsymbol{w}(\boldsymbol{\alpha})$ in the Hilbert space: Given a new point $\boldsymbol{x}$ to predict on, this can be efficiently done according to

$$\langle \boldsymbol{w}(\boldsymbol{\alpha}), \psi(\boldsymbol{x})\rangle = \left\langle \sum_{j=1}^{m} \alpha_j \psi(\boldsymbol{x}_j), \psi(\boldsymbol{x})\right\rangle = \sum_{j=1}^{m} \alpha_j \left\langle \psi(\boldsymbol{x}_j), \psi(\boldsymbol{x})\right\rangle = \sum_{j=1}^{m} \alpha_j k(\boldsymbol{x}_j, \boldsymbol{x}) .$$

Unfortunately, a major handicap of kernel learning (at least in its standard implementation) is that it requires computing and handling an $m \times m$ matrix, where m is the size of the training data, and this can be prohibitive in large-data applications. This has led to a large literature on efficient kernel learning, which attempts to reduce its computational complexity. As far as we know, the algorithms proposed so far fall into one or more of the following categories (see below for specific references):

- *Limiting the number of kernel evaluations:* A dominant computational bottleneck in kernel learning is computing all entries of the kernel matrix. Thus, several algorithms attempt to learn using a much smaller number of kernel evaluations – either by sampling them or using other schemes which require "reading" only a small part of the kernel matrix.

- *Low-Rank Kernel Approximation:* Instead of using the full $m \times m$ kernel matrix, one can use instead a low-rank approximation of it. Learning with a low-rank matrix can be done in a computationally much more efficient manner than with a general kernel matrix (e.g., Scholkopf and Smola (2001); Bach (2013); Alaoui and Mahoney (2014)).

- *Projection to a low-dimensional space:* Each instance $\boldsymbol{x}$ is mapped to a finite-dimensional vector $\phi(\boldsymbol{x}) = \big(\phi_1(\boldsymbol{x}), \ldots, \phi_d(\boldsymbol{x})\big)$ where $d \ll m$, so that $\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}')\rangle \approx k(\boldsymbol{x}, \boldsymbol{x}')$. Note that this is equivalent to a kernel problem where the rank of the kernel matrix is d, so it can be seen as a different kind of low-rank kernel approximation technique.

Existing theoretical results focus on performance guarantees for various algorithms. In this work, we consider a complementary question, which surprisingly has not been thoroughly explored (to the best of our knowledge): What are the inherent obstacles to efficient kernel learning? For example, is it possible to reduce the number of kernel evaluations while maintaining the same learning performance? Is there always a price to pay for low-rank matrix approximation? Can finite-dimensional projection methods match the performance of algorithms working on the original kernel matrix?

Specifically, we study information-theoretic lower bounds on the attainable performance, measured in terms of optimization error on a given training set. We consider two distinct types of constraints:

- The number of kernel evaluations (or equivalently, the number of entries of the kernel matrix observed) is bounded by B, where B is generally assumed to be much smaller than m^2 (the number of entries in the kernel matrix).
- The algorithm solves (2), but using some low-rank matrix $\hat{K}$ instead of K. This can be seen as using a low-rank kernel matrix approximation.

We make no assumptions whatsoever on which kernel evaluations are used, or the type of low-rank approximation, so our results apply to all the methods mentioned previously, and any future potential method which uses these types of approaches. We note that although we focus on optimization error on a given training set, our lower bounds can also be potentially extended to generalization error, where the data is assumed to be sampled i.i.d. from an underlying distribution. We discuss this point further in Section 5.

Our first conclusion, informally stated, is that it is generally impossible to make kernel learning more efficient in a non-trivial manner. For example, suppose we have a budget B on the number of kernel evaluations, where $B \ll m^2$. Then the following "trivial" sub-sampling method turns out to be optimal in general: Sub-sample $\sqrt{B}$ examples from the training data uniformly at random (throwing away all other examples), compute the full $\sqrt{B} \times \sqrt{B}$ kernel matrix based on the sub-sample, and train a predictor using this matrix. This is an extremely näive algorithm, throwing away almost all of the data, yet we show that there are cases where no algorithm can be substantially better. Another pessimistic result can be shown for the low-rank matrix approximation approach: There are cases where any low-rank approximation will impact the attainable performance.

Our formal results go beyond these observations, and quantify the attainable performance as a function of several important problem parameters, such as the kernel matrix rank, regularization parameter, norm of the desired predictor, and the nature of the loss function. In particular:

- Given a kernel evaluation budget constraint B:
  - For the absolute loss, no regularization ($\lambda = 0$), and a constant norm constraint on the domain, we have an error lower bound of $\Omega(B^{-1/4})$. A matching upper bound is obtained by the sub-sampling algorithm discussed earlier.
  - For soft regularization (with regularization parameter $\lambda > 0$ and no norm constraint), we attain error lower bounds which depend on the structure of the loss function. Some particular corollaries include:
    - For the absolute loss, $\Omega(1/\lambda\sqrt{B})$. Again, a matching upper bound is attained by a sub-sampling algorithm.
    - For the hinge loss, $\Omega(1)$ as long as $B < 1/\lambda^2$. Although it only applies in a certain budget regime, it is tight in terms of identifying the kernel evaluation budget required to make the error sub-constant. Moreover, it sheds some light on previous work (e.g., Cotter et al. (2012)) which considered efficient kernel learning methods for the hinge loss.
    - For the squared loss, $\Omega\left(\min\left\{1, \lambda\sqrt{B}\right\}\right)^{-3}$, as long as $B \ll m^2$. Like the result for the other losses, it implies that no sub-constant error is possible unless $B \geq 1/\lambda^2$.
- For learning with low-rank approximation, with rank parameter d, in the case of Ridge Regression (squared loss and soft regularization), we attain an error lower bound of $\Omega((\lambda d)^{-3})$. Thus, to get sub-constant error, we need the rank to scale at least like $1/\lambda$.

The role of the loss function is particularly interesting, since it has not been well-recognized in previous literature, yet our results indicate that it may play a key role in the complexity of kernel learning. For example, as we discuss in Section 3, efficient kernel learning is trivial with the linear loss, harder for smooth and non-linear losses, and appears to be especially hard for non-smooth losses. Our results also highlight the importance of the kernel matrix rank in determining the difficulty of kernel learning. While it has been recognized that low rank can make kernel learning easy (see references below), our results formally establish the reverse direction, namely that (some) high-rank matrices are indeed hard to learn with any algorithm.

Related Work

The literature on efficient kernel methods is vast and we cannot do it full justice. A few representative examples include sparse greedy kernel approximations Scholkopf and Smola (2001), Nyström-based methods, which sample a few rows and columns and use it to construct a low-rank approximation Drineas and Mahoney (2005); Kumar et al. (2009); Alaoui and Mahoney (2014), random finite-dimensional kernel approximations such as random kitchen sinks Rahimi and Recht (2007, 2008); Dai et al. (2014), the kernelized stochastic batch Perceptron for learning with few kernel evaluations Cotter et al. (2012), the random budget Perceptron and the Forgetron Cavallanti et al. (2007); Dekel et al. (2008), divide-and-conquer approaches Zhang et al. (2013); Hsieh et al. (2014), sequential algorithms with early stopping Yao et al. (2007); Raskutti et al. (2014), other numerical-algebraic methods for low-rank approximation, e.g., Fine and Scheinberg (2002); Shawe-Taylor and Cristianini (2004); Bach and Jordan (2005); Mahoney and Drineas (2009); Kumar et al. (2009), combinations of the above Dai et al. (2014), and more. Several works provide a theoretical analysis on the performance of such methods, as a function of the rank, number of kernel evaluations, dimensionality of the finite-dimensional space, and so on. Beyond the works mentioned above, a few other examples include Cortes et al. (2010); Yang et al. (2012); Bach (2013); Lin et al. (2014).

In terms of lower bounds, we note that there are existing results on the error of matrix approximation, based on partial access to the matrix (see Bar-Yossef (2003); Frieze et al. (2004)). However, the way the error is measured is not suitable to our setting, since they focus on the Frobenius norm of $K - \widehat{K}$, where K is the original matrix and $\widehat{K}$ is the approximation. In contrast, in our setting, we are interested in the error of a resulting predictor rather than the quality of matrix approximation. Indeed, as discussed later in the paper, finding such a good approximation of K is a sufficient – but not necessary – condition for learning a good predictor. Another distinct line of work studies how to reduce the complexity of a kernel predictor at test time, e.g., by making it supported on a few support vectors (see for instance Cotter et al. (2013) and references therein). This differs from our work, which focuses on efficiency at training time.

Paper Organization

Our paper is organized as follows. In Section 2, we introduce the class of kernel matrices which shall be used to prove our results, and discuss how they can be generated by standard kernels. In Section 3, we provide lower bounds in a model where the algorithm is constrained in terms of the number of kernel evaluations used. We consider this model in two flavors, one where there is a norm constraint and no regularization (Subsection 3.1), and one where there is regularization without norm constraint (Subsection 3.2). In the former case, we focus on a particular loss, while in the latter case, we provide a more general result and discuss how different types of losses lead

to different types of lower bounds. In Section 4, we consider the model where the algorithm is constrained to use a low-rank kernel matrix approximation. We conclude and discuss open questions in Section 5. Proofs appear in Appendix A.

2. Hard Kernel Matrices

For our results, we utilize a set $\mathcal{K}_{d,m}$ of "hard" kernel matrices, which are essentially permutations of block-diagonal $m \times m$ matrices with at most d blocks. More formally:

Definition 1 *Let $\mathcal{K}'_{d,m}$ be the class of all block-diagonal $m \times m$ matrices, composed of at most d blocks, with entry values of 1 within each block. We define $\mathcal{K}_{d,m}$ to be all matrices which belong to $\mathcal{K}'_{d,m}$ under some permutation $\pi : \{1 \dots m\} \mapsto \{1 \dots m\}$ of their rows and columns:*

$$\mathcal{K}_{d,m} = \left\{ K \in \{0,1\}^{m \times m} : \exists\, \pi, K' \in \mathcal{K}'_{d,m} \text{s.t. } \forall i,j \in \{1 \dots m\},\ K_{i,j} = K'_{\pi(i),\pi(j)} \right\} .$$

From the definition, it is immediate that any $K \in \mathcal{K}_{d,m}$ is positive semidefinite (and hence is a valid kernel matrix), with rank at most d. Moreover, the magnitude of the diagonal elements is at most 1, which means that our data lies in the unit ball in the Hilbert space.

Since our focus is on generic kernel learning, it is sufficient to consider this class in order to establish hardness results. However, it is still worthwhile to consider which kernels can induce this class of kernel matrices. A sufficient condition can be quantified via the following lemma.

Lemma 2 *Suppose there exist $\boldsymbol{z}_1, \dots, \boldsymbol{z}_d \in \mathcal{X}$ such that $k(\boldsymbol{z}_i, \boldsymbol{z}_j) = \mathbb{I}\{\boldsymbol{z}_i = \boldsymbol{z}_j\}$. Then any $K \in \mathcal{K}_{d,m}$ is induced by some m instances $\{\boldsymbol{x}_t\}_{t=1}^m \in \mathcal{X}$.*

The proof is immediate: Given any K, for any block i of size n_i, create n_i copies of $\boldsymbol{z}_i$, and order the instances according to the relevant permutation.

It is straightforward to see that Lemma 2 holds for linear kernels $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle$ and for homogeneous polynomial kernels $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^p$. It also essentially holds for Gaussian kernels $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\boldsymbol{x} - \boldsymbol{x}'\|^2/\gamma)$ if there exist d equi-distant points in $\mathcal{X}$, where the squared distance is much larger than γ. In that case, instead of 0 outside the blocks, we will have ϵ where ϵ is exponentially small, and can be shown to be negligible for our purposes.

That being said, our proof techniques essentially rely on the block-diagonal structure of the covariance matrix, rather the values inside or outside the blocks. Thus, they are potentially applicable to a much wider class of kernels, which can induce kernel matrices composed of some constant a inside each block, and a different constant c outside the blocks. This only requires finding d equi-distant points in the kernel space, and holds for most kernels we are aware of. However, since our results are technical enough as they are, we will concentrate on the boolean case defined previously, where $a = 1, c = 0$, and leave the general case to future work.

Although our formal results and proofs contain many technical details, their basic intuition is quite simple: When d is sufficiently large, any matrix in $\mathcal{K}_{d,m}$ is of high rank, and cannot be approximated well by any low-rank matrix. Therefore, under suitable conditions, no low-rank matrix approximation approach can work well. Moreover, when d is large, then the kernel matrix is quite sparse, and contains a large number of relatively small blocks. Thus, for an appropriate randomized choice of a matrix in $\mathcal{K}_{d,m}$, any algorithm with a limited budget of kernel evaluations will find it difficult to detect these blocks. With a suitable construction, we can reduce the kernel optimization problem to that of detecting these blocks, from which our results follow.

3. Budget Constraints

We now turn to present our results for the case of budget constraints. In this setting, the learning algorithm is given the target values $y_1, \ldots, y_m$, but not the kernel matrix K. Instead, the algorithm may *query* at most B entries in the kernel matrix (where B is a user-defined positive integer), and then return a coefficient vector based on this information, either deterministically or by using additional internal randomness. We denote such an algorithm as a *budgeted* algorithm. This model represents approaches which attempt to reduce the computational complexity of kernel learning by reducing the number of kernel evaluations needed. Standard learning algorithms essentially require $B = \Omega(m^2)$, and the goal is to learn to similar accuracy with a budget $B \ll m^2$. In this section, we discuss the inherent limitations of this approach.

3.1. Norm Constraint, Absolute Loss

We begin by demonstrating a lower bound using the absolute loss $\ell_1(u, y) = |u - y|$ on the domain $\mathcal{W} = \left\{\boldsymbol{w} \,:\, \|\boldsymbol{w}\|^2 \leq 2\right\}$ (or equivalently, coefficient vectors $\boldsymbol{\alpha}$ satisfying $\boldsymbol{\alpha}^\top K \boldsymbol{\alpha} \leq 2$), and our goal is to minimize the average loss, which equals

$$\min_{\boldsymbol{\alpha}\,:\,\boldsymbol{\alpha}^\top K\boldsymbol{\alpha}\leq 2} \frac{1}{m}\sum_{t=1}^{m}\left|\boldsymbol{\alpha}^\top K\boldsymbol{e}_t - y_t\right| \ .$$

Theorem 3 *For any rank parameter d, any sample size $m \geq 2^7 d$, any budget size $B < \frac{3}{50}d^2$, and for any (possibly randomized) budgeted algorithm, there exists a kernel matrix $K \in \mathcal{K}_{2d,m}$ and target values $y_1, \ldots, y_m \in [-1, +1]$, such that the returned coefficient vector $\boldsymbol{\alpha}$ satisfies*

$$\left(\frac{1}{m}\sum_{t=1}^{m}\left|\boldsymbol{\alpha}^\top K\boldsymbol{e}_t - y_t\right|\right) - \min_{\boldsymbol{\alpha}\,:\,\boldsymbol{\alpha}^\top K\boldsymbol{\alpha}\leq 2} \frac{1}{m}\sum_{t=1}^{m}\left|\boldsymbol{\alpha}^\top K\boldsymbol{e}_t - y_t\right| \geq \frac{1}{70\sqrt{d}} \ . \tag{3}$$

The proof and the required construction appears in Subsection A.1. Note that the algorithm is allowed to return any coefficient vector (not necessarily one satisfying the domain constraint $\boldsymbol{\alpha}^\top K\boldsymbol{\alpha} \leq 2$).

The theorem provides a lower bound on the attainable error, for any rank parameter d and assuming the sample size m and budget B are in an appropriate regime. A different way to phrase this is that if B is sufficiently smaller than m^2, then we can find some d on the order of $\sqrt{B}$, such that Theorem 3 holds. More formally:

Corollary 4 *There exist universal constants $c, c' > 0$ such that if $B \leq cm^2$, there is an $m \times m$ kernel matrix K (belonging to $\mathcal{K}_{2d,m}$ for some appropriate d) and target values in $[-1, +1]$ such that the returned coefficient vector $\boldsymbol{\alpha}$ satisfies*

$$\left(\frac{1}{m}\sum_{t=1}^{m}\left|\boldsymbol{\alpha}^\top K\boldsymbol{e}_t - y_t\right|\right) - \min_{\boldsymbol{\alpha}\,:\,\boldsymbol{\alpha}^\top K\boldsymbol{\alpha}\leq 2} \frac{1}{m}\sum_{t=1}^{m}\left|\boldsymbol{\alpha}^\top K\boldsymbol{e}_t - y_t\right| \geq \frac{c'}{\sqrt[4]{B}} \ .$$

In words, the attainable error given a budget of B cannot go down faster than $1/\sqrt[4]{B}$. Next, we show that this is in fact the optimal rate, and is achieved by the following simple strategy:

1. Given a training set $\{(\boldsymbol{x}_t, y_t)\}_{t=1}^m$ of size m, sample $\lfloor\sqrt{B}\rfloor$ training examples uniformly at random (with replacement), getting $\{(\boldsymbol{x}_{t_j}, y_{t_j})\}_{j=1}^{\lfloor\sqrt{B}\rfloor}$.

2. Compute the kernel matrix $\widehat{K} \in \mathbb{R}^{\lfloor\sqrt{B}\rfloor\times\lfloor\sqrt{B}\rfloor}$ defined as $\widehat{K}_{j,j'} = k(\boldsymbol{x}_{t_j}, \boldsymbol{x}_{t_{j'}})$, using at most B queries.

3. Solve the kernel learning problem on the sampled set, getting a coefficient vector $\widehat{\boldsymbol{\alpha}}$:

$$\min_{\widehat{\boldsymbol{\alpha}}\,:\,\widehat{\boldsymbol{\alpha}}^\top\widehat{K}\widehat{\boldsymbol{\alpha}}\leq 2} \left(\frac{1}{\lfloor\sqrt{B}\rfloor} \sum_{j=1}^{\lfloor\sqrt{B}\rfloor} \left| \widehat{\boldsymbol{\alpha}}^\top \widehat{K}\boldsymbol{e}_j - y_{t_j} \right| \right) .$$

4. Return the coefficient vector $\boldsymbol{\alpha}$ such that $\alpha_{t_j} = \widehat{\alpha}_j$ for $j = 1, \ldots, \lfloor\sqrt{B}\rfloor$, and $\alpha_t = 0$ otherwise.

Essentially, this strategy approximately solves the original problem by drawing a subset of the training data —small enough so that we can compute its kernel matrix in full— and solving the learning problem on that data. Since we use a sample of size $\lfloor\sqrt{B}\rfloor$, then by standard generalization guarantees for learning bounded-norm predictors using Lipschitz loss functions (e.g., Kakade et al. (2009)), we get a generalization error upper bound of $\mathcal{O}\left(\frac{1}{\sqrt{\lfloor\sqrt{B}\rfloor}}\right) = \mathcal{O}\left(\frac{1}{\sqrt[4]{B}}\right)$ which matches the lower bound in Corollary 4 up to constants.

To summarize, we see that with the absolute loss, given a constraint on the number of kernel evaluations, there exist no better method than throwing away most of the data, and learning on a sufficiently small subset. Moreover, any method using a non-trivial budget (significantly smaller than m^2) must suffer a performance degradation.

3.2. Soft Regularization, General Losses

Having obtained an essentially tight result for the absolute loss, it is natural to ask what can be obtained for more generic losses. To study this question, it will be convenient to shift to the setting where the domain $\mathcal{W}$ is the entire Hilbert space, and we use a regularization term. Following (2), this reduces to solving

$$\min_{\boldsymbol{\alpha}} \frac{1}{m}\sum_{t=1}^m \ell\left(\boldsymbol{\alpha}^\top K\boldsymbol{e}_t, y_t\right) + \frac{\lambda}{2}\boldsymbol{\alpha}^\top K\boldsymbol{\alpha} .$$

We start by defining the main quantity we are interested in,

$$\Delta_\ell(m,\alpha,K,\lambda,y) = \left(\frac{1}{m}\sum_{t=1}^m \ell\left(\boldsymbol{\alpha}^\top K\boldsymbol{e}_t, y_t\right) + \frac{\lambda}{2}\boldsymbol{\alpha}^\top K\boldsymbol{\alpha}\right) - \min_{\boldsymbol{\alpha}}\left(\frac{1}{m}\sum_{t=1}^m \ell\left(\boldsymbol{\alpha}^\top K\boldsymbol{e}_t, y_t\right) + \frac{\lambda}{2}\boldsymbol{\alpha}^\top K\boldsymbol{\alpha}\right),$$

where ℓ is a loss function.

First, we provide a general result, which applies to any non-negative loss function, and then draw from it corollaries for specific losses:

Theorem 5 *Suppose the loss function ℓ is non-negative. For any rank parameter d, any sample size $m \geq 2^7 d$, any budget $B < \frac{3}{50}d^2$, and for any (possibly randomized) budgeted algorithm,*

there exists a kernel matrix $K \in \mathcal{K}_{2d,m}$ and target values $y_1, \ldots, y_m$ in $\mathcal{Y}$, such that the returned coefficient vector $\boldsymbol{\alpha}$ satisfies

$$\Delta_\ell(m, \alpha, K, \lambda, y) \geq \frac{1}{60}\lambda d \min_{p \in \left[\frac{1}{2}, 2\right]} \max_{y \in \mathcal{Y}} (2u_1^* - u_2^*)^2 \tag{4}$$

where

$$u_1^* = \operatorname*{argmin}_u \ell(u, y) + p\lambda d u^2 \qquad \textit{and} \qquad u_2^* = \operatorname*{argmin}_u \ell(u, y) + \frac{p\lambda d}{2} u^2 \,.$$

The proof and the required construction appears in Subsection A.1.2.

Roughly speaking, to get a non-trivial bound, we need the loss to be such that when the regularization parameter is order of λd, then scaling it by a factor of 2 changes the location of the optimum u^* by a factor different than 2. For instance, this rules out linear losses of the form $\ell(u, y) = yu$. For such a loss, we have

$$u_1^* = \arg\min_u \frac{n}{m} yu + \lambda u^2 = -\frac{ny}{2\lambda m} \qquad \text{and} \qquad u_2^* = \arg\min_u \frac{n}{m} yu + \frac{\lambda}{2} u^2 = -\frac{ny}{\lambda m} \,.$$

Thus we get that $(2u_1^* - u_2^*)^2 = 0$ and the lower bound is trivially 0. While this may seem at first like an unsatisfactory bound, in fact this should be expected: For linear loss and no domain constraints, we don't need to observe the kernel matrix K at all in order to find the optimal solution! To see this, note that the optimization problem in (2) reduces to $\min_{\boldsymbol{\alpha}} \frac{1}{m} \sum_{t=1}^m y_t \boldsymbol{\alpha}^\top K \boldsymbol{e}_t + \frac{\lambda}{2} \boldsymbol{\alpha}^\top K \boldsymbol{\alpha}$, or equivalently $\min_{\boldsymbol{\alpha}} \ \boldsymbol{\alpha}^\top K \boldsymbol{v} + \frac{\lambda}{2} \boldsymbol{\alpha}^\top K \boldsymbol{\alpha}$, where $\boldsymbol{v}$ is a known vector and K is the partially-unknown kernel matrix. Differentiating the expression by $\boldsymbol{\alpha}$ and equating to $\mathbf{0}$, we get $K\boldsymbol{v} + \lambda K \boldsymbol{\alpha} = \mathbf{0}$. Thus, an optimum of this problem is simply $-\frac{1}{\lambda}\boldsymbol{v}$, regardless of what is K. This shows that for linear losses, we can find the optimal predictor with zero queries of the kernel matrix.

Thus, the kernel learning problem is non-trivial only for non-linear losses, which we now turn to examine in more detail.

3.2.1. ABSOLUTE LOSS

First, let us consider again the absolute loss in this setting. We easily get the following corollary of Theorem 5:

Corollary 6 *Let $\ell_1(u, y) = |u - y|$ be the absolute loss. There exist universal constants $c, c' > 0$, such that if $B \leq cm^2$, then for any budgeted algorithm there exists an $m \times m$ kernel matrix K and target values y such that $\Delta_{\ell_1}(m, \alpha, K, \lambda, y)$ is lower bounded by $\frac{c'}{\lambda\sqrt{B}}$.*

Proof To apply Theorem 5, let us compute $(2u_1^* - u_2^*)^2$, where we use the particular choice $y = \frac{1}{2p\lambda d}$. It is readily verified that $u_1^* = u_2^* = y = \frac{1}{2p\lambda d}$, leading to the lower bound

$$\frac{1}{60}\lambda d \min_{p \in \left[\frac{1}{2}, 2\right]} (u_1^*)^2 \ = \ \frac{1}{60}\lambda d \min_{p \in \left[\frac{1}{2}, 2\right]} \left(\frac{1}{2p\lambda d}\right)^2 \ = \ \frac{1}{60}\lambda d \left(\frac{1}{4\lambda d}\right)^2 = \frac{1}{960\lambda d} \,.$$

In particular, suppose we choose $d = \left\lceil \sqrt{\frac{100}{3}B} \right\rceil$. Then we get a lower bound of $\frac{c'}{\lambda\sqrt{B}}$ for $c' = 2^{-13}$. The conditions of Theorem 5 are satisfied if $m \geq 2^7 d = 2^7 \left\lceil \sqrt{\frac{100}{3}B} \right\rceil$ and $B < \frac{3}{50} d^2 =$

$\frac{3}{50}\left(\left\lceil\sqrt{\frac{100}{3}B}\right\rceil\right)^2$. The latter always holds, whereas the former is indeed true if B is smaller than cm^2 for $c = 2^{-20}$. ■

As in the setting of Theorem 3, this lower bound is tight, and we can get a matching $\mathcal{O}\left(1/\lambda\sqrt{B}\right)$ upper bound by learning with a random sub-sample of $\Theta\left(\sqrt{B}\right)$ training examples, using generalization bounds for minimizers of strongly-convex and Lipschitz stochastic optimization problems Sridharan et al. (2009).

Note that, unlike our other lower bounds, Corollary 6 is proven using a different choice of y for each λ. It is not clear whether this requirement is real, or is simply an artifact of our proof technique.

3.2.2. HINGE LOSS

Intuitively, the proof of Corollary 6 relied on the absolute loss having a non-smooth "kink" at $\Theta(1/\lambda\sqrt{B})$, which prevented the optimal u_1^*, u_2^* from moving as a result of the changed regularization parameter. Results of similar flavor can be obtained with any other loss which has an optimum at a non-smooth point. However, when we do not control the location of the "kink" the results may be weaker. A good example is the hinge loss, $\ell_h(u, y) = \max\{0, 1 - uy\}$, which is non-differentiable at the fixed location $p = 1$:

Corollary 7 *Let $\ell_h(u, y) = \max\{0, 1 - uy\}$ be the hinge loss. There exist universal constants $c, c', c'' > 0$, such that if $\lambda \leq \frac{1}{4}$ and $B < \frac{c}{\lambda^2} \leq c'm^2$, then for any budgeted algorithm, there exist an $m \times m$ kernel matrix K and target values y in $\{-1, +1\}$ such that $\Delta_{\ell_h}(m, \alpha, K, \lambda, y)$ is lower bounded by c''.*

Proof To apply Theorem 5, let us compute $(2u_1^* - u_2^*)^2$, where we use the particular choice $y = 1$. It is readily verified that $u_1^* = u_2^* = 1$, as long as $p\lambda d \leq 1$, and is certainly satisfied for any $p \in \left[\frac{1}{2}, 2\right]$ by assuming $\lambda d \leq \frac{1}{2}$. Therefore, if $\lambda d \leq \frac{1}{2}$, then in Theorem 5, we get $u_1^* = u_2^* = 1$, and thus a lower bound of $\frac{1}{60}\lambda d \cdot 1^2 = \frac{1}{60}\lambda d$.

In particular, suppose we pick $d = \left\lfloor\frac{1}{2\lambda}\right\rfloor$. Since we assume $\lambda \leq \frac{1}{4}$, this means that the lower bound above is $\frac{1}{60}\lambda\left\lfloor\frac{1}{2\lambda}\right\rfloor \geq \frac{1}{240} = c''$. The conditions of Theorem 5 are satisfied if $m \geq 2^7 d = 2^7\left\lfloor\frac{1}{2\lambda}\right\rfloor$ and $B < \frac{3}{50}d^2 = \frac{3}{50}\left(\left\lfloor\frac{1}{2\lambda}\right\rfloor\right)^2 < \frac{3}{200}\frac{1}{\lambda^2} = \frac{c}{\lambda^2} < \frac{c}{2^{10}}m^2 = c'm^2$, which are indeed implied by the corollary's conditions. ■

Unlike the bound for the absolute loss, here the result is weaker, and only quantifies a regime under which sub-constant error is impossible. In particular, the condition $B = \mathcal{O}\left(\frac{1}{\lambda^2}\right)$ is not interesting for constant λ. However, in learning problems λ usually scales down as m^{-q} where $q \geq 1/2$ and often $q = 1$. In that case, we get constant error as long as $B \ll m^{2q}$, which establishes that learning is impossible for a budget smaller than a quantity in the range from $\Omega(m)$ to $\Omega(m^2)$, depending on the value of q. For $q = 1$, that is $\lambda = \Theta(1/m)$, learning is impossible without querying a constant fraction of the kernel matrix.

Moreover, it is possible to show that our lower bound is tight, in terms of identifying the threshold for making the error sub-constant. As before, we consider the strategy of sub-sampling $\sqrt{B}$ training examples and learning with respect to the induced kernel matrix. Since we use $\sqrt{B}$ examples and λ-strongly convex regularization, the expected error scales as $\frac{1}{\lambda\sqrt{B}}$ Sridharan et al. (2009). This is sub-constant in the regime $B = \omega(1/\lambda^2)$, and matches our lower bound. We emphasize that when B is $\omega(1/\lambda^2)$, we do not have a non-trivial lower bound, and it remains an open problem to understand what can be attained for the hinge loss in this regime.

Another interesting consequence of the corollary is the required budget as a function of the norm of a "good predictor" we want to compete with. In Cotter et al. (2012), several algorithmic approaches have been studied, which were all shown to require $\Omega(\|\boldsymbol{u}\|^4)$ kernel evaluations to be competitive with a given predictor $\boldsymbol{u}$, even in the "realizable" case where the predictor attains zero average hinge loss. An examination of the proof of theorem 2 reveals that the construction is such that there exists a predictor $\boldsymbol{u}$ which attains zero hinge loss on all the examples, and whose norm[1] is $\mathcal{O}(1/\sqrt{\lambda})$. Corollary 7 shows that the budget must be at least $\Omega(1/\lambda^2) = \Omega(\|\boldsymbol{u}\|^4)$ to get sub-constant error in the worst case. Although our setting is slightly different than Cotter et al. (2012), this provides evidence that the $\Omega(\|\boldsymbol{u}\|^4)$ bounds in Cotter et al. (2012) are tight in terms of the norm dependence.

3.2.3. SQUARED LOSS

In the case of absolute loss and hinge loss, the results depend on a non-differentiable point in the loss function. It is thus natural to conclude by considering a smooth differentiable loss, such as the squared loss:

Corollary 8 *Let $\ell_2(u,y) = (u-y)^2$ be the squared loss. There exist universal constants $c, c' > 0$, such that*

- *If $1 \leq B \leq \frac{1}{\lambda^2} \leq cm^2$, then for any budgeted algorithm there exists an $m \times m$ kernel matrix and target values in $[-1,+1]$ such that $\Delta_{\ell_2}(m,\alpha,K,\lambda,y)$ is lower bounded by c'.*
- *If $\frac{1}{\lambda^2} \leq B \leq cm^2$, then for any budgeted algorithm there exists an $m \times m$ kernel matrix and target values in $[-1,+1]$ such that $\Delta_{\ell_2}(m,\alpha,K,\lambda,y)$ is lower bounded by $c'(\lambda\sqrt{B})^{-3}$.*

This lower bound is weaker than the $\Omega(1/\lambda\sqrt{B})$ lower bound attained for the absolute loss. This is essentially due to the smoothness of the squared loss, and we do not know if it is tight. In any case, it proves that even for the squared loss, at least $1/\lambda^2$ kernel evaluations are required to get sub-constant error. In learning problems, where λ often scales down as m^{-q} (where $q \geq 1/2$ and often $q = 1$), we get a required budget size of m^{2q}. This is super-linear when $p > 1/2$, and becomes m^2 when $q = 1$ – in other words, we need to compute a constant portion of the entire kernel matrix.

Proof To apply Theorem 5, let us compute $(2u_1^* - u_2^*)^2$. It is readily verified that $u_1^* = \frac{y}{1+p\lambda d}$ and $u_2^* = \frac{y}{1+p\lambda d/2}$, leading to the lower bound

$$\frac{1}{60}\lambda d \min_{p\in\left[\frac{1}{2},2\right]} \max_{y\in\mathcal{Y}} \left(\frac{2y}{1+p\lambda d} - \frac{y}{1+p\lambda d/2}\right)^2 = \frac{1}{60}\lambda d \min_{p\in\left[\frac{1}{2},2\right]} \max_{y\in\mathcal{Y}} \left(\frac{y}{(1+p\lambda d)\left(1+\frac{p\lambda d}{2}\right)}\right)^2$$
$$\geq \frac{1}{60}\lambda d \min_{p\in\left[\frac{1}{2},2\right]} \max_{y\in\mathcal{Y}} \left(\frac{y}{(1+p\lambda d)^2}\right)^2 .$$

Taking in particular $y = 1$, we get $\frac{1}{60}\lambda d \min_{p\in\left[\frac{1}{2},2\right]} \left(\frac{1}{(1+p\lambda d)^2}\right)^2 \geq \frac{1}{60}\frac{\lambda d}{(1+2\lambda d)^4}$. We now consider two ways to pick d, corresponding to the two cases considered in the corollary:

1. To see this, recall that we use a block-diagonal kernel matrix composed of at most $2d$ all-ones blocks, and where $y = 1$ always. So by picking $\alpha_t = 1/n_i$ for any index t in block i (where n_i is the size of the block), we get zero hinge loss, and the norm is $\sqrt{\boldsymbol{\alpha}^\top K \boldsymbol{\alpha}} \leq \sqrt{\sum_i 1} \leq \sqrt{2d}$. Moreover, in the proof of Corollary 7 we pick $d = \left\lfloor \frac{1}{2\lambda} \right\rfloor$, so the norm is $\mathcal{O}(1/\sqrt{\lambda})$.

- If $1 \leq B \leq \frac{1}{\lambda^2}$, we pick $d = \left\lceil \sqrt{\frac{100}{3\lambda^2}} \right\rceil$. Since $\lambda \leq 1$, we have $d\lambda < 7$, and this means that the bound above is bounded below by $c' = 2^{-18}$. The conditions of Theorem 5 are satisfied if $m \geq 2^7 d = 2^7 \left\lceil \sqrt{\frac{100}{3\lambda^2}} \right\rceil$ and $B < \frac{3}{50} d^2 = \frac{3}{50} \left(\left\lceil \sqrt{\frac{100}{3\lambda^2}} \right\rceil \right)^2$. These are satisfied by assuming $B \leq \frac{1}{\lambda^2} \leq cm^2$ for $c = 2^{-20}$.

- If $B \geq \frac{1}{\lambda^2}$, we pick $d = \left\lceil \sqrt{\frac{100}{3} B} \right\rceil$. Plugging this into the bound above and using the assumption $B \geq \frac{1}{\lambda^2}$ (or equivalently, $\lambda\sqrt{B} \geq 1$), we get a lower bound of $c' \frac{\lambda\sqrt{B}}{(\lambda\sqrt{B})^4} = c'(\lambda\sqrt{B})^{-3}$ for an appropriate constant $c' = 2^{-18} < 1/960$. Moreover, the conditions of Theorem 5 are satisfied if $m \geq 2^7 d = 2^7 \left\lceil \sqrt{\frac{100}{3} B} \right\rceil$ and $B < \frac{3}{50} d^2 = \frac{3}{50} \left(\left\lceil \sqrt{\frac{100}{3} B} \right\rceil \right)^2$. The latter always holds, whereas the former indeed holds if B is less than cm^2 for $c = 2^{-20}$.

■

4. Low-Rank Constraints

In this section, we turn to discuss the second broad class of approaches, which replace the original kernel matrix K by a low-rank approximation K'. As explained earlier, many rank-reduction approaches – including Nyström method and random features – use a low-rank approximation K' with entries defined by $K'_{t,t'} = \langle \phi(\boldsymbol{x}_t), \phi(\boldsymbol{x}_{t'}) \rangle$, where $\{(\boldsymbol{x}_t, y_t)\}_{t=1}^m$ is the training set and $\phi : \mathcal{X} \mapsto \mathbb{R}^d$ is a given feature mapping, typically depending on the data.

The next result shows a lower bound on the error for any such low-rank approximation method when the algorithm used for learning is kernel Ridge Regression (i.e., when we use the squared loss and employ soft regularization).

Theorem 9 *Given a training set $\{(\boldsymbol{x}_t, y_t)\}_{t=1}^m \in \left(\mathcal{X} \times \{-1, +1\}\right)^m$ with corresponding kernel matrix K, consider a Ridge Regression algorithm operating on any matrix K' of rank at most d (where $2d$ divides m), and $K'_{t,t'} = \langle \phi(\boldsymbol{x}_t), \phi(\boldsymbol{x}_{t'}) \rangle$ for some mapping $\phi : \mathcal{X} \mapsto \mathbb{R}^d$ (possibly depending on the training set). Furthermore, suppose there exist $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{2d} \in \mathcal{X}$ such that $k(\boldsymbol{v}_i, \boldsymbol{v}_j) = \mathbb{I}\{\boldsymbol{v}_i = \boldsymbol{v}_j\}$. Then there exists a training set such that the coefficient vector $\boldsymbol{\alpha}$ returned by the algorithm satisfies*

$$\Delta_{\ell_2}(m, \alpha, K, \lambda, y) \geq \frac{1}{2(\lambda d)^2(1 + \lambda d)}$$

When $\lambda d \geq 1$, we get a $\Omega\big((\lambda d)^{-3}\big)$ bound. This bears similarities to the bound in Corollary 8, which considered the squared loss in the budgeted setting, where d is replaced by $\sqrt{B}$ (i.e., $\Omega\big((\lambda\sqrt{B})^{-3}\big)$ when $\lambda\sqrt{B} \geq 1$). The bound implies that to get sub-constant error, the rank required must be larger than $1/\lambda$. When λ itself scales down with the sample size m, we get that the required rank grows with the sample size. When $\lambda = 1/m$, the required rank is $\Omega(m)$, which means that any low-rank approximation scheme (where $d \ll m$) will lead to constant error. As in the case of Corollary 8, we do not know whether our lower bound is tight.

5. Discussion and Open Questions

In this paper, we studied fundamental information-theoretic barriers to efficient kernel learning, focusing on algorithms which either limit the number of kernel evaluations, or use a low-rank kernel matrix approximation. We provided several results under various settings, highlighting the influence of the kernel matrix rank, regularization parameter, norm constraint and nature of the loss function on the attainable performance.

For general losses and kernel matrices, our conclusion is generally pessimistic. In particular, when the number of kernel evaluations is bounded, there are cases where no algorithm attains performance better than a trivial sub-sampling strategy, where most of the data is thrown away. Also, no algorithm can work well when the regularization parameter is sufficiently small or the norm constraint is sufficiently large. On a more optimistic note, our lower bounds are substantially weaker when dealing with smooth losses. Although we do not know if these weaker lower bounds are tight, they may indicate that better kernel learning algorithms are possible by exploiting smoothness of the loss. Smoothness of the squared loss has been used in Zhang et al. (2013), but perhaps this property can be utilized more generally.

In our results, we focused on the problem of minimizing regularized training error on a given training set. This is a different goal than minimizing generalization error in a stochastic setting, where the data is assumed to be drawn i.i.d. from some underlying distribution. However, we believe that our lower bounds should also be applicable in terms of optimizing the risk (or expected error with respect to the underlying distribution). The main obstacle is that our lower bounds are proven for a given class of kernel matrices, which are not induced by an explicit i.i.d. sampling process of training instances. However, inspecting our basic construction in Subsection A.1, it can be seen that it is very close to such a process: The kernel is constructed by *pairs* of instances sampled i.i.d. from a finite set $\{\boldsymbol{v}_i^1, \boldsymbol{v}_i^2\}_{i=1}^d$. We believe that all our results would hold if the instances were to be sampled i.i.d. from $\{\boldsymbol{v}_i^1, \boldsymbol{v}_i^2\}_{i=1}^d$. The reason that we sample pairs is purely technical, since it ensures that for every i, there is an equal number of $\boldsymbol{v}_i^1$ and $\boldsymbol{v}_i^2$ in the training set, making the calculations more tractable. Morally, the same techniques should work with i.i.d. sampling, as long as the probability of sampling $\boldsymbol{v}_i^1$ and $\boldsymbol{v}_i^2$ are the same for all i.

Our work leaves several questions open. First, while the results for the absolute loss are tight, we do not know if this is the case for our other results. Second, the low-rank result in Section 4 applies only to squared loss (Ridge Regression), and it would be interesting to extend it to other losses. Third, it should be possible to extend our results also to randomized algorithms that query the kernel matrix a number of times bounded by B only in expectation (with respect to the algorithm's internal randomization), rather than deterministically. Finally, our results may indicate that at least for smooth losses, better kernel learning algorithms are possible, and remain to be discovered.

Acknowledgments

This research was carried out in part while the authors were attending the research program on the Mathematics of Machine Learning, at the Centre de Recerca Matemàtica of the Universitat Autònoma de Barcelona (Spain). Partial support is gratefully acknowledged. YM is supported in part by a grant from the Israel Science Foundation, a grant from the United States-Israel Binational Science Foundation (BSF), a grant by Israel Ministry of Science and Technology and the Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11). OS is supported in part by an Israeli Science Foundation grant (no. 425/13) and an FP7 Marie Curie CIG grant.

References

A. El Alaoui and M. Mahoney. Fast randomized kernel methods with statistical guarantees. *CoRR*, abs/1411.0306, 2014.

F. Bach. Sharp analysis of low-rank kernel matrix approximations. In *COLT*, 2013.

F. Bach and M. Jordan. Predictive low-rank decomposition for kernel methods. In *ICML*, 2005.

Z. Bar-Yossef. Sampling lower bounds via information theory. In *STOC*, 2003.

Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile. Tracking the best hyperplane with a simple budget Perceptron. *Machine Learning*, 69(2-3):143–167, 2007.

C. Cortes, M. Mohri, and A. Talwalkar. On the impact of kernel approximation on learning accuracy. In *AISTATS*, 2010.

A. Cotter, S. Shalev-Shwartz, and N. Srebro. The kernelized stochastic batch Perceptron. In *ICML*, 2012.

A. Cotter, S. Shalev-Shwartz, and N. Srebro. Learning optimally sparse Support Vector Machines. In *ICML*, 2013.

B. Dai, B. Xie, N. He, Y. Liang, A. Raj, M.-F. Balcan, and L. Song. Scalable kernel methods via doubly stochastic gradients. *arXiv preprint arXiv:1407.5599*, 2014.

O. Dekel, S. Shalev-Shwartz, and Y. Singer. The forgetron: A kernel-based Perceptron on a budget. *SIAM Journal on Computing*, 37(5):1342–1372, 2008.

P. Drineas and M. Mahoney. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *The Journal of Machine Learning Research*, 2:243–264, 2002.

A. Frieze, R. Kannan, and S. Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.

C.-J. Hsieh, S. Si, and I. Dhillon. A divide-and-conquer solver for kernel Support Vector Machines. In *ICML*, 2014.

S. Kakade, K. Sridharan, and A. Tewari. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. In *Advances in Neural Information Processing Systems*, pages 793–800, 2009.

S. Kumar, M. Mohri, and A. Talwalkar. Sampling techniques for the Nyström method. In *AISTATS*, 2009.

M. Lin, S. Weng, and C. Zhang. On the sample complexity of random Fourier features for online learning: How many random Fourier features do we need? *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(3):13, 2014.

M. Mahoney and P. Drineas. CUR matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, 2009.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, 2007.

A. Rahimi and B. Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *NIPS*, 2008.

G. Raskutti, M. Wainwright, and B. Yu. Early stopping and non-parametric regression: an optimal data-dependent stopping rule. *Journal of Machine Learning Research*, 15(1):335–366, 2014.

B. Scholkopf and A. Smola. *Learning with kernels: Support Vector Machines, regularization, optimization, and beyond*. MIT press, 2001.

J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge University Press, 2004.

K. Sridharan, S. Shalev-Shwartz, and N. Srebro. Fast rates for regularized objectives. In *NIPS*, 2009.

T. Yang, Y.-F. Li, M. Mahdavi, R. Jin, and Z.-H. Zhou. Nyström method vs random Fourier features: A theoretical and empirical comparison. In *NIPS*, 2012.

Y. Yao, L. Rosasco, and A. Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

Y. Zhang, J. Duchi, and M. Wainwright. Divide and conquer kernel Ridge Regression. In *COLT*, 2013.

Appendix A. Proofs

A.1. Construction properties from Section 3

We consider a randomized strategy, where the kernel matrix is sampled randomly from $\mathcal{K}_{2d,m}$ (according to a distribution $\mathcal{D}$ to be defined shortly), and $y_1, \ldots, y_m$ are fixed deterministically in a certain way. We will analyze what is the best possible performance using any budgeted algorithm, in expectation over this strategy.

To define the distribution $\mathcal{D}$, we let $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{2d}$ be the standard basis vectors in $\mathbb{R}^{2d}$, and sample a kernel matrix from $\mathcal{D}$ as follows:

- Pick $\boldsymbol{\sigma} \in \{0,1\}^d$ uniformly at random.
- For all $i \in \{1, \ldots, d\}$, define $\boldsymbol{v}_i^1, \boldsymbol{v}_i^2$ as
  - $\boldsymbol{v}_i^1 = \boldsymbol{v}_i^2 = \boldsymbol{e}_i$ if $\sigma_i = 1$,
  - $\boldsymbol{v}_i^1 = \boldsymbol{e}_i$ and $\boldsymbol{v}_i^2 = \boldsymbol{e}_{i+d}$ if $\sigma_i = 0$.
- For $j = 1, \ldots, m/2$, choose $(\boldsymbol{z}_{2j-1}, \boldsymbol{z}_{2j})$ uniformly at random from $\{(\boldsymbol{v}_i^1, \boldsymbol{v}_i^2)\}_{i=1}^d$.
- Choose a permutation $\pi : \{1, \ldots, m\} \mapsto \{1, \ldots, m\}$ uniformly at random.
- Return the kernel matrix K defined as $K_{i,j} = \left\langle \boldsymbol{z}_{\pi(i)}, \boldsymbol{z}_{\pi(j)} \right\rangle$ for all $i, j = 1, \ldots, m$.

To understand the construction, we begin by noting that K represents the inner product of a set of vectors, and hence is always positive semidefinite and a valid kernel matrix. Moreover, $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m$ are all in the set $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{2d}\}$, and therefore the resulting kernel matrix equals (up to permutation of rows and columns) a block-diagonal matrix of the following form:

$$\begin{array}{|ccccccc|}
\hline
\mathbf{1} & S_1 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\
S_1^\top & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{1} & S_2 & \cdots & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & S_2^\top & \mathbf{1} & \cdots & \mathbf{0} & \mathbf{0} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{1} & S_d \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & S_d^\top & \mathbf{1} \\
\hline
\end{array}$$

Here, S_i is an all-zero block if $\sigma_i = 0$, and an all-ones block if $\sigma_i = 1$. In other words, the matrix is composed of d blocks, one for each value of $i = 1, \ldots, d$. If $\sigma_i = 1$, then block i is a monolithic all-ones block (corresponding to $\boldsymbol{e}_i$), and if $\sigma_i = 0$, then block i is composed of two equal-sized sub-blocks (corresponding to $\boldsymbol{e}_i$ and to $\boldsymbol{e}_{d+i}$). This implies that the kernel matrix is indeed in $\mathcal{K}_{2d,m}$.

Our proofs rely on the following intuition: To achieve small error, the learning algorithm must know the values of the entries in $S_1, S_2, \ldots, S_d$ (i.e., the values of $\boldsymbol{\sigma}$). However, when d is large, these blocks are rather small, and their entries are randomly permuted in the matrix. Thus, any algorithm with a constrained query budget is likely to "miss" many of these blocks.

To simplify the presentation, we will require a few auxiliary definitions. First, given a kernel matrix $K \in \mathcal{K}_{2d,m}$ constructed as above, let

$$T_{i,1} = \{\pi(t) : \boldsymbol{z}_t = \boldsymbol{v}_i^1\} \qquad \text{and} \qquad T_{i,2} = \{\pi(t) : \boldsymbol{z}_t = \boldsymbol{v}_i^2\}$$

denote the set of row/column indices in the kernel matrix, corresponding to instances which were chosen to be $\boldsymbol{v}_i^1$ (respectively $\boldsymbol{v}_i^2$). Note that $\{T_{i,1}, T_{i,2}\}_{i=1}^d$ is a disjoint partition of all indices $\{1, \ldots, m\}$, and $|T_{i,1}| = |T_{i,2}|$. We then define,

$$T_i = T_{i,1} \cup T_{i,2} \qquad \text{and} \qquad N_i = |T_i|, \tag{5}$$

and also define,

$$\beta_{i,1} = \sum_{t \in T_{i,1}} \alpha_t \qquad \text{and} \qquad \beta_{i,2} = \sum_{t \in T_{i,2}} \alpha_t, \tag{6}$$

to be the sum of the corresponding coefficients in the solution $\boldsymbol{\alpha}$ returned by the algorithm. With these definitions, we can re-write the average loss and the regularization term as follows.

Lemma 10 *For any coefficient vector $\boldsymbol{\alpha}$,*

$$\frac{1}{m}\sum_{t=1}^{m} \ell\left(\boldsymbol{\alpha}^\top K \boldsymbol{e}_i, y\right) = \sum_{i=1}^{d} \frac{N_i}{2m}\left(\ell(\beta_{i,1} + \sigma_i\beta_{i,2}, y) + \ell(\sigma_i\beta_{i,1} + \beta_{i,2}, y)\right) \text{ and}$$

$$\boldsymbol{\alpha}^\top K \boldsymbol{\alpha} = \sum_{i=1}^{d}\left(\beta_{i,1}^2 + \beta_{i,2}^2 + 2\sigma_i\beta_{i,1}\beta_{i,2}\right),$$

where $\beta_{i,1}, \beta_{i,2}$ are defined in (6).

The proof is a straightforward exercise based on the definition of K. Finally, we define E_i to be the event that the algorithm never queries a pair of inputs in T_i, i.e., the algorithm's queries $(s_1, r_1), \ldots, (s_B, r_B)$ on the kernel matrix satisfy

$$s_t \notin T_i \ \vee \ r_t \notin T_i \qquad t = 1, \ldots, B\,.$$

To prove our results, we will require two key lemmas, presented below, which quantify how any budgeted algorithm is likely to "miss" many blocks, and hence have its output relatively insensitive to $\boldsymbol{\sigma}$.

Lemma 11 *Suppose $m \geq 2d$ and $B < \frac{3}{50}d^2$. Then for any deterministic learning algorithm,*

$$\sum_{i=1}^{d} \mathbb{P}(E_i) > \frac{d}{2}\,.$$

The formal proof is provided below. Although it is quite technical, the lemma's intuition is very simple: Recall that the kernel matrix is composed of d blocks, each of size $\frac{m}{d} \times \frac{m}{d}$ in expectation. Thus, if we choose an entry uniformly at random, the chance of "hitting" some block is approximately $d\frac{(m/d)^2}{m^2} = \frac{1}{d}$. Thus, if we sample B points uniformly at random, where $B \ll d^2$, then the number of "missed" blocks $\sum_{i=1}^{d} \mathbb{I}\{E_i\}$ is likely to be $\Omega(d)$. The lemma above simply quantifies this, and shows that this holds not just for uniform sampling, but for any algorithm with a budgeted number of queries.

Proof Recall that each T_i corresponds to one of d blocks in the kernel matrix (possibly composed of two sub-blocks). The algorithm queries $(s_1, r_1), \ldots, (s_B, r_B)$. For each possible query at time t we

define the set $Q_{s,t}$ of blocks such that s was queried with a member of that block and we obtained a value zero in the kernel matrix. Namely,

$$Q_{s,t} = \Big\{ i = 1, \dots, d \,:\, (\exists \tau < t)\; s_\tau = s \wedge r_\tau \in T_i \wedge K_{s_\tau, r_\tau} = 0 \Big\} \\ \cup \Big\{ i = 1, \dots, d \,:\, (\exists \tau < t)\; r_\tau = s \wedge s_\tau \in T_i \wedge K_{s_\tau, r_\tau} = 0 \Big\} .$$

Given the query (s_t, r_t) we define $L_t = Q_{s_t,t}$ to be the blocks in which some member was queried with s_t, and $R_t = Q_{r_t,t}$ the blocks in which some member was queried with r_t.

We introduce a quantity P_t defined as follows: $P_t = d + 1$ if there is a query $t' < t$ such that $K_{s_{t'}, r_{t'}} = 1$ and, moreover, $s_t = s_{t'}$ or $r_t = r_{t'}$ (that is, the block of s_t or the block of r_t was already discovered). Otherwise, let $P_t = \max\big\{|L_t|, |R_t|\big\}$.

Let D_t be the event that the t-th query discovers a new block. That is, D_t is true if and only if $K_{s_t, r_t} = 1$ and $P_t < d + 1$. Using this notation,

$$\sum_{i=1}^{d} \mathbb{I}\{\neg E_i\} = \sum_{t=1}^{B} \mathbb{I}\{D_t\} = \sum_{t=1}^{B} \mathbb{I}\{D_t \wedge P_t < d/2\} + \underbrace{\sum_{t=1}^{B} \mathbb{I}\{D_t \wedge P_t \geq d/2\}}_{N} . \tag{7}$$

We will now show that unless $B \geq \frac{3}{50} d^2$, we can upper bound N deterministically by $\sqrt{2B}$. We do this by considering separately the case $N \leq \frac{d}{2}$ and the case $N > \frac{d}{2}$:

- Assume first $N > \frac{d}{2}$, and let $t_1, \dots, t_N$ be the times t_k such that $\mathbb{I}\{D_{t_k} \wedge P_{t_k} \geq d/2\} = 1$. Now fix some k and note that, because the common block to which s_{t_k} and r_{t_k} both belong is discovered, neither s_{t_k} nor r_{t_k} can occur in a future query (s_t, r_t) that discovers a new block. Therefore, in order to have $\mathbb{I}\{D_t \wedge P_t \geq d/2\} = 1$ for $N > \frac{d}{2}$ times, at least

  $$\frac{d}{2} + \left(\frac{d}{2} - 1\right) + \cdots + 1$$

  queries must be made, where each term $\frac{d}{2} - k + 1$ accounts for the fact that each one of the previous $k - 1$ discovered blocks might contribute with at most a query to making $P_t \geq \frac{d}{2}$. So, it must be

  $$B \geq \sum_{k=1}^{d/2} \left(\frac{d}{2} - (k-1)\right) \geq \frac{d^2}{8}$$

  queries to discover the first $\frac{d}{2}$ blocks, which contradicts the lemma's assumption that $B \leq \frac{3}{50} d^2$. Therefore, $N \leq \frac{d}{2}$.

- Assume that $N \leq \frac{d}{2}$. Using the same logic as before, in order to have $\mathbb{I}\{D_t \wedge P_t \geq d/2\} = 1$ for $N \leq \frac{d}{2}$ times, at least

  $$\frac{d}{2} + \left(\frac{d}{2} - 1\right) + \cdots + \left(\frac{d}{2} - N + 1\right)$$

  queries must be made. So, it must be

  $$B \geq \sum_{k=1}^{N} \left(\frac{d}{2} - (k-1)\right) = (d+1)\frac{N}{2} - \frac{N^2}{2}$$

or, equivalently, $N^2 - (d+1)N + 2B \geq 0$. Solving this quadratic inequality for N, and using the lemma's assumption that $N \leq \frac{d}{2}$, we have that $N \leq \frac{(d+1)-\sqrt{(d+1)^2-8B}}{2}$. Using the lemma's assumption that $B \leq \frac{3}{50}d^2$ we get that $N \leq \sqrt{2B}$.

We now bound the first term of (7) in expectation. For any time t and query (s_t, r_t), we say that s_t is paired with r_t if $K_{s_t,r_t} = \langle \boldsymbol{z}_{2j-1}, \boldsymbol{z}_{2j} \rangle$ for some $j \in \{1, \ldots, m/2\}$, where $\{s_t, r_t\} \equiv \{\pi(2j-1), \pi(2j)\}$. Clearly, $\mathbb{P}(s_t \text{ paired with } r_t) = \frac{1}{m}$, where the probability is over the random draw of the permutation π. Hence,

$$\begin{aligned}
&\sum_{t=1}^{B} \mathbb{P}\big(D_t \wedge P_t < d/2\big) \\
&\quad \leq \sum_{t=1}^{B} \mathbb{P}\big(D_t \mid P_t < d/2,\ s_t \text{ not paired with } r_t\big) + \sum_{t=1}^{B} \mathbb{P}\big(s_t \text{ paired with } r_t\big) \\
&\quad \leq \sum_{t=1}^{B} \mathbb{P}\big(D_t \mid P_t < d/2,\ s_t \text{ not paired with } r_t\big) + \frac{B}{m}\,.
\end{aligned}$$

Let $\mathbb{P}' = \mathbb{P}\big(\cdot \mid P_t < d/2,\ s_t \text{ not paired with } r_t\big)$. Note that the two points s_t and r_t have independent block assignments when conditioned on s_t not paired with r_t. Moreover, conditioned on $P_t < d/2$, the event D_t implies $s_t, r_t \in T_i$ for some $i \in \neg L_t \cap \neg R_t$, where $|L_t|, |R_t| < \frac{d}{2}$ and for any $S \subseteq \{1, \ldots, d\}$ we use $\neg S$ to denote $\{1, \ldots, d\} \setminus S$.

Since, by definition of L_t, the block of s_t is not in L_t, and there were no previous queries involving s_t and a point belonging to a block in $\neg L_t$, we have that

$$\mathbb{P}'\big(s_t \in T_i \,\big|\, L_t\big) = \frac{1}{|\neg L_t|} \qquad \forall i \notin L_t\,.$$

Likewise,

$$\mathbb{P}'\big(r_t \in T_j \,\big|\, R_t\big) = \frac{1}{|\neg R_t|} \qquad \forall j \notin R_t\,.$$

Hence, for L', R' ranging over all subsets of $\{1, \ldots, d\}$ of size strictly less than $\frac{d}{2}$,

$$\begin{aligned}
\mathbb{P}'(D_t) &= \sum_{L',R'} \sum_{i \in \neg L' \cap \neg R'} \mathbb{P}'\big(s_t \in T_i \wedge r_t \in T_i \,\big|\, L_t = L',\ R_t = R'\big)\, \mathbb{P}'(L_t = L' \wedge R_t = R') \\
&= \sum_{L',R'} \sum_{i \in \neg L' \cap \neg R'} \mathbb{P}'\big(s_t \in T_i \,\big|\, L_t = L'\big)\, \mathbb{P}'\big(r_t \in T_i \,\big|\, R_t = R'\big)\, \mathbb{P}'(L_t = L' \wedge R_t = R')\,. \\
&= \sum_{L',R'} \sum_{i \in \neg L' \cap \neg R'} \frac{1}{|\neg L'|}\,\frac{1}{|\neg R'|}\, \mathbb{P}'(L_t = L' \wedge R_t = R') \\
&= \sum_{L',R'} \frac{|\neg L' \cap \neg R'|}{|\neg L'|\,|\neg R'|}\, \mathbb{P}'(L_t = L' \wedge R_t = R') \leq \frac{2}{d}
\end{aligned}$$

because $|\neg L'| \geq \frac{d}{2}$, $|\neg R'| \geq \frac{d}{2}$ and $|\neg L' \cap \neg R'| \leq \min\{|\neg L'|, |\neg R'|\}$. Therefore, using $m \geq 2d$ we can write

$$\sum_{t=1}^{B} \mathbb{P}\big(D_t \wedge P_t < d/2\big) \leq \frac{2B}{d} + \frac{B}{m} \leq \frac{5B}{2d}$$

where we used the lemma's assumption that $m \geq 2d$. Putting everything together, we get the following upper bound on the expectation of (7):

$$\mathbb{E}\left[\sum_{i=1}^{d} \mathbb{I}\{\neg E_i\}\right] \leq \frac{5B}{2d} + \sqrt{2B}\,. \tag{8}$$

On the other hand, we have

$$\mathbb{E}\left[\sum_{i=1}^{d} \mathbb{I}\{\neg E_i\}\right] = \mathbb{E}\left[\sum_{i=1}^{d} (1 - \mathbb{I}\{E_i\})\right] = d - \sum_{i=1}^{d} \mathbb{P}(E_i)\,. \tag{9}$$

Combining (8) and (9), we get that

$$\sum_{i=1}^{d} \mathbb{P}(E_i) \geq d - \frac{5B}{2d} - \sqrt{2B}\,.$$

To finish the lemma's proof, suppose on the contrary that $\sum_{i=1}^{d} \mathbb{P}(E_i) \leq \frac{d}{2}$. Then from the equation above, we would get that

$$\frac{d}{2} \geq d - \frac{5B}{2d} - \sqrt{2B}$$

which implies $B \geq \left(\frac{\sqrt{7}-\sqrt{2}}{5}\right)^2 d^2 > 0.06d^2$, contradicting the lemma's assumptions. Therefore, we must have $\sum_{i=1}^{d} \mathbb{P}(E_i) > \frac{d}{2}$ as required. ■

Lemma 12 *Suppose the kernel matrix K is sampled according to the distribution $\mathcal{D}$ as defined earlier (using a parameter $\boldsymbol{\sigma} \in \{0,1\}^d$). Let A_i be any event that, conditioned on N_i, depends only on $\beta_{i,1}$ and $\beta_{i,2}$, (as returned by a deterministic algorithm based on access to the kernel matrix), and let $g(N_i)$ be some non-negative function of N_i. Then*

$$\mathbb{E}\Big[g(N_i)\big(\mathbb{I}\{A_i, \sigma_i = 1\} + \mathbb{I}\{\neg A_i, \sigma_i = 0\}\big)\Big] \;\geq\; \frac{1}{2}\,\mathbb{E}\big[g(N_i)\,\mathbb{P}(E_i \mid N_i)\big]\,.$$

Proof We begin by noting that

$$\begin{aligned}
&\mathbb{E}\Big[g(N_i)\big(\mathbb{I}\{A_i, \sigma_i = 1\} + \mathbb{I}\{\neg A_i, \sigma_i = 0\}\big)\Big] \\
&= \mathbb{E}\Big[\mathbb{E}\big[g(N_i)\,(\mathbb{I}\{A_i, \sigma_i = 1\} + \mathbb{I}\{\neg A_i, \sigma_i = 0\}) \mid N_i\big]\Big] \\
&= \mathbb{E}\Big[g(N_i)\big(\mathbb{P}(A_i, \sigma_i = 1 \mid N_i) + \mathbb{P}(\neg A_i, \sigma_i = 0 \mid N_i)\big)\Big]\,.
\end{aligned} \tag{10}$$

We now continue by analyzing the probabilities in the expression:

$$\mathbb{P}(A_i, \sigma_i = 1 \mid N_i) + \mathbb{P}(\neg A_i, \sigma_i = 0 \mid N_i)\,. \tag{11}$$

We will need two auxiliary results. First, we argue that for all i,

$$\mathbb{P}(E_i \mid \sigma_i = 0, N_i) = \mathbb{P}(E_i \mid \sigma_i = 1, N_i) = \mathbb{P}(E_i \mid N_i)\,. \tag{12}$$

To prove this, we note that since the algorithm is deterministic, the occurrence of the event E_i is determined by the kernel matrix, and more specifically the entries of the kernel matrix observed by the algorithm. Therefore, if the kernel matrix is such that E_i occurs, then the algorithm's output would not change if we flip the value of σ_i as it only affects entries which were not touched by the algorithm. Therefore, $\mathbb{P}(E_i \mid \sigma_i = 0, N_i) = \mathbb{P}(E_i \mid \sigma_i = 1, N_i)$. Since σ_i is either 0 or 1, this means that these probabilities also equal $\mathbb{P}(E_i \mid N_i)$.

Second, we argue that

$$\mathbb{P}(A_i \mid E_i, \sigma_i = 0, N_i) = \mathbb{P}(A_i \mid E_i, \sigma_i = 1, N_i)\ . \tag{13}$$

This holds because if E_i occurs, then $\beta_{i,1}, \beta_{i,2}$ depend only on entries which are independent of σ_i. Moreover, N_i is also independent of σ_i. Therefore A_i, which is assumed to depend only on $\beta_{i,1}, \beta_{i,2}$ when conditioned on N_i, is also independent of σ_i when conditioned on E_i and N_i, from which (13) follows.

Using (12), (13), and the fact that σ_i is uniformly drawn from $\{0,1\}$ and independent of N_i, we have that (11) equals

$$\begin{aligned}
&\mathbb{P}(A_i, \sigma_i = 1 \mid N_i) + \mathbb{P}(\neg A_i, \sigma_i = 0 \mid N_i) \\
&= \mathbb{P}(\sigma_i = 1)\mathbb{P}(A_i \mid \sigma_i = 1, N_i) + \mathbb{P}(\sigma_i = 0)\mathbb{P}(\neg A_i \mid \sigma_i = 0, N_i) \\
&= \frac{1}{2}\Big(\mathbb{P}(A_i \mid \sigma_i = 1, N_i) + \mathbb{P}(\neg A_i \mid \sigma_i = 0, N_i)\Big) \\
&= \frac{1}{2}\Big(1 - \mathbb{P}(A_i \mid \sigma_i = 0, N_i) + \mathbb{P}(A_i \mid \sigma_i = 1, N_i)\Big) \\
&= \frac{1}{2}\Big(1 - \mathbb{P}(E_i \mid \sigma_i = 0, N_i)\mathbb{P}(A_i \mid E_i, \sigma_i = 0, N_i) - \mathbb{P}(\neg E_i \mid \sigma_i = 0, N_i)\mathbb{P}(A_i \mid \neg E_i, \sigma_i = 0, N_i) \\
&\qquad + \mathbb{P}(E_i \mid \sigma_i = 1, N_i)\mathbb{P}(A_i \mid E_i, \sigma_i = 1, N_i) + \mathbb{P}(\neg E_i \mid \sigma_i = 1, N_i)\mathbb{P}(A_i \mid \neg E_i, \sigma_i = 1, N_i)\Big) \\
&= \frac{1}{2}\Big(1 + \mathbb{P}(E_i \mid N_i)\big(\mathbb{P}(A_i \mid E_i, \sigma_i = 1, N_i) - \mathbb{P}(A_i \mid E_i, \sigma_i = 0, N_i)\big) \\
&\qquad + \mathbb{P}(\neg E_i \mid N_i)\big(\mathbb{P}(A_i \mid \neg E_i, \sigma_i = 1, N_i) - \mathbb{P}(A_i \mid \neg E_i, \sigma_i = 0, N_i)\big)\Big) \\
&\geq \frac{1}{2}\Big(1 + \mathbb{P}(E_i \mid N_i) \times 0 + \mathbb{P}(\neg E_i \mid N_i) \times (-1)\Big) \;=\; \frac{1}{2}\big(1 - \mathbb{P}(\neg E_i \mid N_i)\big) \;=\; \frac{1}{2}\mathbb{P}(E_i \mid N_i)\ .
\end{aligned}$$

Plugging this lower bound on (11) back into (10), the result follows. ■

Finally, we will also require the following auxiliary result:

Lemma 13 *Let $N = \sum_{i=1}^{m/2} 2X_i$ where $X_1, \ldots, X_{m/2}$ are i.i.d. Bernoulli random variables with parameter $1/d$. Also, let $g(N)$ be a non-negative function of N. Then for any event Z,*

$$\mathbb{E}\big[g(N)\,\mathbb{P}(Z \mid N = n)\big] \geq \left(\min_{\frac{m}{2d} \leq n \leq \frac{2m}{d}} g(n)\right)\left(\mathbb{P}(Z) - 2\exp\left(-\frac{m}{8d}\right)\right).$$

Proof Let $S = \left\{\frac{m}{2d}, \frac{m}{2d}+1, \ldots, \frac{2m}{d}\right\}$. We can lower bound the expectation by

$$\begin{aligned}
&\sum_{n\in S} \mathbb{P}(N=n) g(n) \mathbb{P}(Z \mid N=n) \;\geq\; \left(\min_{n\in S} g(n)\right) \sum_{n\in S} \mathbb{P}(N=n)\mathbb{P}(Z\mid n) \\
&= \left(\min_{n\in S} g(n)\right)\left(\mathbb{P}(Z) - \sum_{n\notin S} \mathbb{P}(N=n)\mathbb{P}(Z\mid n)\right) \\
&\geq \left(\min_{n\in S} g(n)\right)\left(\mathbb{P}(Z) - \sum_{n\notin S} \mathbb{P}(N=n)\right) \\
&= \left(\min_{n\in S} g(n)\right)\Big(\mathbb{P}(Z) - \mathbb{P}(N\notin S)\Big)\,. \qquad (14)
\end{aligned}$$

Since N is distributed as twice the sum of $m/2$ i.i.d. Bernoulli random variables with parameter $1/d$, so its expectation is m/d, and by multiplicative Chernoff bounds and union bounds,

$$\mathbb{P}(N\notin S) = \mathbb{P}\left(N > \frac{2m}{d}\right) + \mathbb{P}\left(N < \frac{m}{2d}\right) \leq \exp\left(-\frac{m}{3d}\right) + \exp\left(-\frac{m}{8d}\right) \leq 2\exp\left(-\frac{m}{8d}\right)\,.$$

Substituting this back into (14), the result follows. ■

A.1.1. PROOF OF THEOREM 3

Suppose we pick $y_t = \frac{1}{\sqrt{d}}$ for all t. Using Yao's minimax principle, it is sufficient to prove a lower bound for

$$\mathbb{E}\left[\frac{1}{m}\sum_{t=1}^{m}\left|\boldsymbol{\alpha}^\top K\boldsymbol{e}_t - \frac{1}{\sqrt{d}}\right| - \min_{\boldsymbol{\alpha}\,:\,\boldsymbol{\alpha}^\top K\boldsymbol{\alpha}\leq 2} \frac{1}{m}\sum_{t=1}^{m}\left|\boldsymbol{\alpha}^\top K\boldsymbol{e}_t - \frac{1}{\sqrt{d}}\right|\right] \geq \frac{1}{70\sqrt{d}}\;, \qquad (15)$$

where the expectation is with respect to the kernel matrix K drawn according to the distribution $\mathcal{D}$ specified earlier, and $\boldsymbol{\alpha}$ is any deterministic function of K encoding the learning algorithm. This ensures that for any (possibly randomized) algorithm, there exists some K which satisfies the theorem statement.

First, we will show that for any $K \in \mathcal{K}_{2d,m}$, there exists some $\boldsymbol{\alpha}$ such that

$$\frac{1}{m}\sum_{t=1}^{m}\left|\boldsymbol{\alpha}^\top K\boldsymbol{e}_t - \frac{1}{\sqrt{d}}\right| = 0 \qquad \text{and} \qquad \boldsymbol{\alpha}^\top K\boldsymbol{\alpha} \leq 2\,. \qquad (16)$$

This implies that (15) can be re-written as

$$\mathbb{E}\left[\frac{1}{m}\sum_{t=1}^{m}\left|\boldsymbol{\alpha}^\top K\boldsymbol{e}_t - \frac{1}{\sqrt{d}}\right|\right]\,. \qquad (17)$$

To see this, we utilize Lemma 10 to rewrite (16) as

$$\begin{aligned}
\frac{1}{m}\sum_{t=1}^{m}\ell\left(\boldsymbol{\alpha}^\top K\boldsymbol{e}_i, y\right) &= \sum_{i=1}^{d}\frac{N_i}{2m}\left(\left|\beta_{i,1}+\sigma_i\beta_{i,2}-\frac{1}{\sqrt{d}}\right| + \left|\sigma_i\beta_{i,1}+\beta_{i,2}-\frac{1}{\sqrt{d}}\right|\right) = 0, \text{ and} \\
\boldsymbol{\alpha}^\top K\boldsymbol{\alpha} &= \sum_{i=1}^{d}\left(\beta_{i,1}^2+\beta_{i,2}^2+2\sigma_i\beta_{i,1}\beta_{i,2}\right) \leq 2\;,
\end{aligned}$$

where $\{\beta_{i,1}, \beta_{i,2}\}$ are the appropriate functions of $\boldsymbol{\alpha}$. Note that these constraints are indeed satisfied for any $\boldsymbol{\alpha}$ for which $\beta_{i,1} = \beta_{i,2} = \frac{1}{\sqrt{d}}$ if $\sigma_i = 0$, and $\beta_{i,1} = \frac{1}{\sqrt{d}}, \beta_{i,2} = 0$ if $\sigma_i = 1$. Again using Lemma 10, we can rewrite (17) as

$$\mathbb{E}\left[\sum_{i=1}^{d} \frac{N_i}{2m}\left(\left|\beta_{i,1} + \sigma_i\beta_{i,2} - \frac{1}{\sqrt{d}}\right| + \left|\sigma_i\beta_{i,1} + \beta_{i,2} - \frac{1}{\sqrt{d}}\right|\right)\right] . \tag{18}$$

Let us consider the expression $\left|\beta_{i,1} + \sigma_i\beta_{i,2} - \frac{1}{\sqrt{d}}\right| + \left|\sigma_i\beta_{i,1} + \beta_{i,2} - \frac{1}{\sqrt{d}}\right|$ for some fixed choice of the kernel matrix K. In particular:

- If $\sigma_i = 1$ and $\beta_{i,1} + \beta_{i,2} \geq \frac{3}{2\sqrt{d}}$, then

$$\left|\beta_{i,1} + \sigma_i\beta_{i,2} - \frac{1}{\sqrt{d}}\right| + \left|\sigma_i\beta_{i,1} + \beta_{i,2} - \frac{1}{\sqrt{d}}\right| \;=\; 2\left|\beta_{i,1} + \beta_{i,2} - \frac{1}{\sqrt{d}}\right| \;\geq\; \frac{1}{\sqrt{d}} .$$

- If $\sigma_i = 0$ and $\beta_{i,1} + \beta_{i,2} < \frac{3}{2\sqrt{d}}$, then either $\beta_{i,1}$ or $\beta_{i,2}$ must be less than $\frac{3}{4\sqrt{d}}$, and therefore

$$\begin{aligned}\left|\beta_{i,1} + \sigma_i\beta_{i,2} - \frac{1}{\sqrt{d}}\right| + \left|\sigma_i\beta_{i,1} + \beta_{i,2} - \frac{1}{\sqrt{d}}\right| &= \left|\beta_{i,1} - \frac{1}{\sqrt{d}}\right| + \left|\beta_{i,2} - \frac{1}{\sqrt{d}}\right| \\ &\geq \left|\frac{3}{4\sqrt{d}} - \frac{1}{\sqrt{d}}\right| = \frac{1}{4\sqrt{d}} .\end{aligned}$$

Let A_i be the event that $\beta_{i,1} + \beta_{i,2} \geq \frac{3}{2\sqrt{d}}$. Since the algorithm is deterministic, $\{\beta_{i,1}, \beta_{i,2}\}$ and hence A_i is determined by the kernel matrix K. By the analysis above, we can lower bound (18) by

$$\begin{aligned}&\mathbb{E}\left[\sum_{i=1}^{d} \frac{N_i}{2m}\left(\mathbb{I}\left\{A_i, \sigma_i = 1\right\}\frac{1}{\sqrt{d}} + \mathbb{I}\left\{\neg A_i, \sigma_i = 0\right\}\frac{1}{4\sqrt{d}}\right)\right] \\ &\geq \frac{1}{8m\sqrt{d}}\sum_{i=1}^{d}\mathbb{E}\Big[N_i\big(\mathbb{I}\left\{A_i, \sigma_i = 1\right\} + \mathbb{I}\left\{\neg A_i, \sigma_i = 0\right\}\big)\Big] .\end{aligned}$$

By Lemma 12 and Lemma 13 this is lower bounded by

$$\frac{1}{16m\sqrt{d}}\sum_{i=1}^{d}\mathbb{E}\big[N_i\,\mathbb{P}(E_i \mid N_i)\big] \;\geq\; \frac{1}{32\sqrt{d}}\left(\frac{1}{d}\sum_{i=1}^{d}\mathbb{P}(E_i) - 2\exp\left(-\frac{m}{8d}\right)\right) .$$

Since we assumed that $B < \frac{3}{50}d^2$ and $m \geq 2^7 d$, we can apply Lemma 11, which lower bounded this by

$$\frac{1}{32\sqrt{d}}\left(\frac{1}{2} - 2\exp\left(-\frac{m}{8d}\right)\right) \;\geq\; \frac{1}{70\sqrt{d}} ,$$

where we used again the assumption that $m \geq 2^7 d$. ■

A.1.2. PROOF OF THEOREM 5

The proof is broadly similar to the one of Theorem 3, but using a generic loss rather than the absolute loss.

Suppose we pick $y_t = y \in \mathcal{Y}$ for all t, where $y \in \mathcal{Y}$ will be determined later. Using Yao's minimax principle, it is sufficient to prove a lower bound for

$$\mathbb{E}_K\left[\left(\frac{1}{m}\sum_{t=1}^{m}\ell\left(\boldsymbol{\alpha}^\top K\boldsymbol{e}_t, y\right)+\frac{\lambda}{2}\boldsymbol{\alpha}^\top K\boldsymbol{\alpha}\right)-\min_{\boldsymbol{\alpha}}\left(\frac{1}{m}\sum_{t=1}^{m}\ell\left(\boldsymbol{\alpha}^\top K\boldsymbol{e}_t, y\right)+\frac{\lambda}{2}\boldsymbol{\alpha}^\top K\boldsymbol{\alpha}\right)\right] \tag{19}$$

where —as in the proof of Theorem 3— the expectation is with respect to the kernel matrix K drawn according to the distribution $\mathcal{D}$, and $\boldsymbol{\alpha}$ is any deterministic function of K encoding the learning algorithm. This ensures that for any (possibly randomized) algorithm, there exists some K which satisfies the theorem statement.

Utilizing Lemma 10, we can rewrite (19) as

$$\begin{aligned}
&\mathbb{E}\left[\left(\sum_{i=1}^{d}\frac{N_i}{2m}\Big(\ell(\beta_{i,1}+\sigma_i\beta_{i,2},y)+\ell(\sigma_i\beta_{i,1}+\beta_{i,2},y)\Big)+\frac{\lambda}{2}\sum_{i=1}^{d}\left(\beta_{i,1}^2+\beta_{i,2}^2+2\sigma_i\beta_{i,1}\beta_{i,2}\right)\right)\right.\\
&\quad-\min_{\{\beta_{i,1},\beta_{i,2}\}}\left(\sum_{i=1}^{d}\frac{N_i}{2m}\Big(\ell(\beta_{i,1}+\sigma_i\beta_{i,2},y)+\ell(\sigma_i\beta_{i,1}+\beta_{i,2},y)\Big)\right.\\
&\quad\left.\left.+\frac{\lambda}{2}\sum_{i=1}^{d}\left(\beta_{i,1}^2+\beta_{i,2}^2+2\sigma_i\beta_{i,1}\beta_{i,2}\right)\right)\right]\\
&=\sum_{i=1}^{d}\mathbb{E}\left[\frac{N_i}{2m}\Big(\ell(\beta_{i,1}+\sigma_i\beta_{i,2},y)+\ell(\sigma_i\beta_{i,1}+\beta_{i,2},y)\Big)+\frac{\lambda}{2}\left(\beta_{i,1}^2+\beta_{i,2}^2+2\sigma_i\beta_{i,1}\beta_{i,2}\right)\right.\\
&\quad\left.-\min_{\beta_{i,1},\beta_{i,2}}\left(\frac{N_i}{2m}\Big(\ell(\beta_{i,1}+\sigma_i\beta_{i,2},y)+\ell(\sigma_i\beta_{i,1}+\beta_{i,2},y)\Big)+\frac{\lambda}{2}\left(\beta_{i,1}^2+\beta_{i,2}^2+2\sigma_i\beta_{i,1}\beta_{i,2}\right)\right)\right]\,.
\end{aligned}$$

This can be written in a simplified form as

$$\sum_{i=1}^{d}\mathbb{E}\left[g_{N_i}^{\sigma_i}(\beta_{i,1},\beta_{i,2})\right]\,, \tag{20}$$

where

$$\begin{aligned}
g_n^\sigma(u,v)&=f_n^\sigma(u,v)-\min_{u,v}f_n^\sigma(u,v),\ \text{and}\\
f_n^\sigma(u,v)&=\frac{n}{2m}\big(\ell(u+\sigma v,y)+\ell(\sigma u+v,y)\big)+\frac{\lambda}{2}\left(u^2+v^2+2\sigma uv\right)\,.
\end{aligned}$$

Now, let A_i be the event that $g_{N_i}^0(\beta_{i,1},\beta_{i,2})<g_{N_i}^1(\beta_{i,1},\beta_{i,2})$. We consider two cases:

- If $\sigma_i=1$ and A_i occurs, then

$$g_{N_i}^{\sigma_i}(\beta_{i,1},\beta_{i,2})=\max_{\sigma}g_{N_i}^{\sigma}(\beta_{i,1},\beta_{i,2})\geq\min_{u,v}\max_{\sigma}g_{N_i}^{\sigma}(u,v)\,.$$

- If $\sigma_i = 0$ and A_i does not occur, then

$$g^{\sigma_i}_{N_i}(\beta_{i,1}, \beta_{i,2}) = \max_{\sigma} g^{\sigma}_{N_i}(\beta_{i,1}, \beta_{i,2}) \geq \min_{u,v} \max_{\sigma} g^{\sigma}_{N_i}(u, v) \; .$$

Therefore, using the fact that g^{σ}_n is non-negative by definition, we have

$$g^{\sigma_i}_{N_i}(\beta_{i,1}, \beta_{i,2}) \geq \left(\mathbb{I}\left\{A_i, \sigma_i = 1\right\} + \mathbb{I}\left\{\neg A_i, \sigma_i = 0\right\}\right) \min_{u,v} \max_{\sigma} g^{\sigma}_{N_i}(u, v) \; .$$

Substituting this back into (20), and using Lemma 12, Lemma 13 and Lemma 11 in order, we get a lower bound of the form

$$\begin{aligned}
&\sum_{i=1}^{d} \mathbb{E}\left[\left(\min_{u,v} \max_{\sigma} g^{\sigma}_{N_i}(u, v)\right)\left(\mathbb{I}\left\{A_i, \sigma_i = 1\right\} + \mathbb{I}\left\{\neg A_i, \sigma_i = 0\right\}\right)\right] \\
&\quad \geq \frac{1}{2} \sum_{i=1}^{d} \mathbb{E}\left[\left(\min_{u,v} \max_{\sigma} g^{\sigma}_{N_i}(u, v)\right) \mathbb{P}(E_i \mid N_i)\right] \\
&\quad \geq \frac{1}{2} \sum_{i=1}^{d} \left(\min_{\frac{m}{2d} \leq n_i \leq \frac{2m}{d}} \min_{u,v} \max_{\sigma} g^{\sigma}_{n_i}(u, v)\right)\left(\mathbb{P}(E_i) - 2\exp\left(-\frac{m}{8d}\right)\right) \\
&\quad = \frac{1}{2} \left(\min_{\frac{m}{2d} \leq n \leq \frac{2m}{d}} \min_{u,v} \max_{\sigma} g^{\sigma}_{n}(u, v)\right) \sum_{i=1}^{d} \left(\mathbb{P}(E_i) - 2\exp\left(-\frac{m}{8d}\right)\right) \\
&\quad \geq \frac{d}{2} \left(\min_{\frac{m}{2d} \leq n \leq \frac{2m}{d}} \min_{u,v} \max_{\sigma} g^{\sigma}_{n}(u, v)\right)\left(\frac{1}{2} - 2\exp\left(-\frac{m}{8d}\right)\right) \\
&\quad = d\left(\frac{1}{4} - \exp\left(-\frac{m}{8d}\right)\right)\left(\min_{\frac{m}{2d} \leq n \leq \frac{2m}{d}} \min_{u,v} \max_{\sigma} g^{\sigma}_{n}(u, v)\right)
\end{aligned}$$

where the first inequality follows from Lemma 12, the second inequality is from Lemma 13, and the third inequality is by Lemma 11. Since we assume $m \geq 2^7 d$, this is at least

$$\frac{d}{5}\left(\min_{\frac{m}{2d} \leq n \leq \frac{2m}{d}} \min_{u,v} \max_{\sigma} g^{\sigma}_{n}(u, v)\right) \; . \tag{21}$$

We now turn to analyze $\min_{u,v} \max_{\sigma} g^{\sigma}_n(u, v)$. By definition of $g^{\sigma}_n(u, v)$, we have that

$$g^0_n(u, v) = f^0_n(u, v) - \min_{u,v} f^0_n(u, v) = \frac{n}{2m}\big(\ell(u, y) + \ell(v, y)\big) + \frac{\lambda}{2}(u^2 + v^2) - \min_{u,v} f^0_n(u, v) \; .$$

It is readily seen that this function is λ-strongly convex in (u, v), and attains a minimal value of 0 at some (u^*_1, u^*_1), where

$$u^*_1 = \operatorname*{argmin}_u \frac{n}{m}\ell(u, y) + \lambda u^2 \; .$$

Using the property of λ-strong convexity, we have for all u, v that

$$g^0_n(u, v) = g^0_n(u, v) - g^0_n(u^*_1, u^*_1) \geq \frac{\lambda}{2}\|(u, v) - (u^*_1, u^*_1)\|^2 = \frac{\lambda}{2}\big((u - u^*_1)^2 + (v - u^*_1)^2\big) \; . \tag{22}$$

Also, by definition,

$$g_n^1(u,v) = f_n^1(u,v) - \min_{u,v} f_n^1(u,v) = \frac{n}{m}\ell(u+v,y) + \frac{\lambda}{2}(u+v)^2 - \min_{u,v} f_n^1(u,v)$$

which is a λ strongly-convex function in $u+v$, and attains a minimal value of 0 at any u_2, v_2 such that $u_2^* = u_2 + v_2$, where

$$u_2^* \;=\; \operatorname*{argmin}_u \frac{n}{m}\ell(u,y) + \frac{\lambda}{2}u^2.$$

Using the property of λ-strong convexity, we have for all u, v that

$$g_n^1(u,v) = g^1(u,v) - g^1(u_2,v_2) \geq \frac{\lambda}{2}\big((u+v)-(u_2+v_2)\big)^2 = \frac{\lambda}{2}(u+v-u_2^*)^2\,. \tag{23}$$

Combining (22) and (23), and using the fact that the maximum is lower bounded by the average, this implies that

$$\begin{aligned}\min_{u,v}\max_{\sigma} g_n^\sigma(u,v) &\geq \min_{u,v}\max\left\{\frac{\lambda}{2}\big((u-u_1^*)^2+(v-u_1^*)^2\big),\ \frac{\lambda}{2}(u+v-u_2^*)^2\right\}\\ &\geq \min_{u,v}\frac{\lambda}{4}\Big((u-u_1^*)^2+(v-u_1^*)^2+(u+v-u_2^*)^2\Big)\,.\end{aligned}$$

A straightforward calculation reveals that this expression is minimized at $u = v = \frac{1}{3}(u_1^* + u_2^*)$, leading to a value of $\frac{\lambda}{12}(2u_1^* - u_2^*)^2$. To summarize, we showed that

$$\min_{u,v}\max_{\sigma} g_n^\sigma(u,v) \geq \frac{\lambda}{12}(2u_1^* - u_2^*)^2$$

where

$$u_1^* = \arg\min_u \frac{n}{m}\ell(u,y) + \lambda u^2 \qquad \text{and} \qquad u_2^* \;=\; \arg\min_u \frac{n}{m}\ell(u,y) + \frac{\lambda}{2}u^2\,.$$

This computation holds for any value of y, and therefore we have

$$\min_{u,v}\max_{\sigma} g_n^\sigma(u,v) \geq \max_{y\in\mathcal{Y}} \frac{\lambda}{12}(2u_1^* - u_2^*)^2$$

where u_1^*, u_2^* are as defined above. Substituting this back into (21), we get

$$\frac{1}{60}\lambda d\left(\min_{\frac{m}{2d}\leq n\leq\frac{2m}{d}} \max_{y\in\mathcal{Y}} (2u_1^* - u_2^*)^2\right)\,. \tag{24}$$

Finally, to write this in a simpler form, let $p = \frac{m}{nd}$. Then we can equivalently write u_1^*, u_2^* as

$$\begin{aligned}u_1^* &= \operatorname*{argmin}_u \ell(u,y) + \frac{m}{n}\lambda u^2 = \operatorname*{argmin}_u \ell(u,y) + p\lambda d u^2\\ u_2^* &= \operatorname*{argmin}_u \ell(u,y) + \frac{m\lambda}{2n}u^2 = \operatorname*{argmin}_u \ell(u,y) + \frac{p\lambda d}{2}u^2\,.\end{aligned}$$

Moreover, the constraint $\frac{m}{2d} \leq n \leq \frac{2m}{d}$ implies that $p \in \left[\frac{1}{2}, 2\right]$, so we can lower bound (24) by

$$\frac{1}{60}\lambda d\left(\min_{p\in\left[\frac{1}{2},2\right]} \max_{y\in\mathcal{Y}} (2u_1^* - u_2^*)^2\right)$$

as desired. ■

A.2. Proof of Theorem 9

Recall that we assume that m is divisible by $2d$. Given $t = 1, \ldots, m$, define

$$i(t) = 1 + \left\lfloor \frac{t-1}{m/2d} \right\rfloor$$

to be the partition function of $\{1, \ldots, m\}$ into $2d$ equal-sized blocks:

$$i(1) = i(2) = \cdots = i(m/2d) = 1$$
$$i(m/2d+1) = \cdots = i(2m/2d) = 2$$

and so on, until $i(m) = 2d$.

Suppose we choose the target values $y_1, \ldots, y_m$ according to $y_t = z_{i(t)}$, where $\boldsymbol{z} = (z_1, \ldots, z_{2d})$ is to be chosen later, and let $\boldsymbol{x}_t = \boldsymbol{v}_{i(t)}$ for $t = 1, \ldots, m$. Recall that $k(\boldsymbol{v}_i, \boldsymbol{v}_j) = \mathbb{I}\{\boldsymbol{v}_i = \boldsymbol{v}_j\}$. It is easily seen that these instances induce a block-diagonal kernel matrix $K \in \mathcal{K}_{2d,m}$, composed of $2d$ all-one blocks of equal size $m/(2d)$. Moreover, any low-rank matrix K' used by the algorithm will also have a block-wise structure (with possibly different values for the entries), where $K'_{t,t'} = \left\langle \phi(\boldsymbol{v}_{i(t)}), \phi(\boldsymbol{v}_{i(t')}) \right\rangle$.

Given any such block-wise kernel matrix K, composed of $2d$ uniform blocks of size $m/2d$, let G^K be the $2d \times 2d$ matrix defined as $G_{i(t),i(t')} = K_{t,t'}$. Note that since K is symmetric, G^K is symmetric as well. Finally, given some coefficient vector $\boldsymbol{\alpha}$, define $\boldsymbol{\beta}$ as

$$\forall i = 1, \ldots, d, \quad \beta_i = \sum_{t\,:\,i(t)=i} \alpha_t \,.$$

With this notation, we can re-write the objective function and resulting solution using the following lemma.

Lemma 14 *For any block matrix K, where $K_{t,t'} = K_{r,r'}$ if $i(t) = i(r)$ and $i(t') = i(r')$, and any coefficient vector $\boldsymbol{\alpha}$ with corresponding $\boldsymbol{\beta}$, we have*

$$\begin{aligned}\frac{1}{m}\sum_{t=1}^{m}\left(\boldsymbol{\alpha}^\top K\boldsymbol{e}_t - y_t\right)^2 + \frac{\lambda}{2}\boldsymbol{\alpha}^\top K\boldsymbol{\alpha} &= \frac{1}{m}\left(\boldsymbol{\alpha}^\top\left(K + \frac{m\lambda}{2}I\right)K\boldsymbol{\alpha} - 2\boldsymbol{y}^\top K\boldsymbol{\alpha} + \|\boldsymbol{y}\|^2\right) \\ &= \frac{1}{2d}\left(\boldsymbol{\beta}^\top\left(G^K + d\lambda I\right)G^K\boldsymbol{\beta} - 2\boldsymbol{z}^\top G^K\boldsymbol{\beta} + \|\boldsymbol{z}\|^2\right).\end{aligned}$$

Moreover, if $\boldsymbol{\alpha} = \left(K + \frac{\lambda m}{2}I\right)^{-1}\boldsymbol{y}$, then $\boldsymbol{\beta} = (G^K + d\lambda I)^{-1}\boldsymbol{z}$.

The proof is a technical exercise, and appears separately in Subsection A.3.

In our case, we chose the training instances so that K is a block-diagonal matrix composed of $2d$ equal-sized all-ones block. Therefore, G^K in our case is simply the $d \times d$ identity matrix. By Lemma 14, we can write the objective function as

$$\frac{1}{m}\sum_{t=1}^{m}\left(\boldsymbol{\alpha}^\top K\boldsymbol{e}_t - y_t\right)^2 + \frac{\lambda}{2}\boldsymbol{\alpha}^\top K\boldsymbol{\alpha} \;=\; \frac{1}{2d}\left((1+d\lambda)\|\boldsymbol{\beta}\|^2 - 2\boldsymbol{z}^\top\boldsymbol{\beta} + \|\boldsymbol{z}\|^2\right).$$

This function is $\frac{1+d\lambda}{d}$-strongly convex in $\boldsymbol{\beta}$, and is minimized at $\boldsymbol{\beta}^* = \frac{1}{1+d\lambda}\boldsymbol{z}$. Therefore, the error obtained by any other solution $\boldsymbol{\beta}$ is at least

$$\frac{1+d\lambda}{2d}\left\|\boldsymbol{\beta} - \frac{1}{1+d\lambda}\boldsymbol{z}\right\|^2. \tag{25}$$

According to Lemma 14 and the definition of the algorithm, the $\boldsymbol{\beta}$ corresponding to the coefficient vector $\boldsymbol{\alpha}$ returned by the learning algorithm (using a kernel matrix K') satisfies $\boldsymbol{\beta} = (G^{K'} + d\lambda I)^{-1}\boldsymbol{z}$. Plugging this back into (25), we get an error lower bound of

$$\frac{1+d\lambda}{2d}\left\|(G^{K'} + d\lambda I)^{-1}\boldsymbol{z} - \frac{1}{1+d\lambda}\boldsymbol{z}\right\|^2 = \frac{1+d\lambda}{2d}\left\|\left((G^{K'} + d\lambda I)^{-1} - \frac{1}{1+d\lambda}I\right)\boldsymbol{z}\right\|^2. \tag{26}$$

Let $USU^\top$ be the spectral decomposition of $G^{K'}$, where $U = \left[\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{2d}\right] \in \mathbb{R}^{2d\times 2d}$ is an orthonormal matrix, and S is a diagonal matrix with eigenvalues $s_1 \leq s_2 \leq \ldots s_{2d}$ on the diagonal. Moreover, since K' is a matrix of rank at most d, it follows that $G^{K'}$ is also of rank at most d, hence $s_1 = \cdots = s_d = 0$. We can therefore re-write (26) as

$$\begin{aligned}
&\frac{1+d\lambda}{2d}\left\|\left(U(S+d\lambda I)^{-1}U^\top - \frac{1}{1+d\lambda}I\right)\boldsymbol{z}\right\|^2 \\
&= \frac{1+d\lambda}{2d}\left\|U\left((S+d\lambda I)^{-1} - \frac{1}{1+d\lambda}I\right)U^\top\boldsymbol{z}\right\|^2 \\
&= \frac{1+d\lambda}{2d}\left\|\left((S+d\lambda I)^{-1} - \frac{1}{1+d\lambda}I\right)U^\top\boldsymbol{z}\right\|^2 \\
&= \frac{1+d\lambda}{2d}\sum_{i=1}^{2d}\left(\left(\frac{1}{s_i+d\lambda} - \frac{1}{1+d\lambda}\right)\boldsymbol{u}_i^\top\boldsymbol{z}\right)^2 \\
&\geq \frac{1+d\lambda}{2d}\sum_{i=1}^{d}\left(\left(\frac{1}{s_i+d\lambda} - \frac{1}{1+d\lambda}\right)\boldsymbol{u}_i^\top\boldsymbol{z}\right)^2 \\
&= \frac{1+d\lambda}{2d}\sum_{i=1}^{d}\left(\left(\frac{1}{d\lambda} - \frac{1}{1+d\lambda}\right)\boldsymbol{u}_i^\top\boldsymbol{z}\right)^2 \\
&= \frac{1+d\lambda}{2d}\frac{1}{\left(d\lambda(1+d\lambda)\right)^2}\sum_{i=1}^{d}\left(\boldsymbol{u}_i^\top\boldsymbol{z}\right)^2 \\
&= \frac{1}{2(d\lambda)^2(1+d\lambda)d}\sum_{i=1}^{d}\left(\boldsymbol{u}_i^\top\boldsymbol{z}\right)^2.
\end{aligned}$$

We are now free to choose $\boldsymbol{z} = (z_1, \ldots, z_{2d})$, which induces some choice of the target values $y_1, \ldots, y_m$, to get the final bound. In particular, we argue that there exist some $\boldsymbol{z} \in \{-1, +1\}^{2d}$ such that

$$\sum_{i=1}^{d}\left(\boldsymbol{u}_i^\top\boldsymbol{z}\right)^2 \geq d$$

from which the result follows. To show this, we use the probabilistic method: Suppose that $\boldsymbol{z}$ is chosen uniformly at random from $\{-1,+1\}^{2d}$. Then

$$\mathbb{E}\left[\sum_{i=1}^{d}\left(\boldsymbol{u}_i^\top \boldsymbol{z}\right)^2\right] = \mathbb{E}\left[\sum_{i=1}^{d}\boldsymbol{u}_i^\top \boldsymbol{z}\boldsymbol{z}^\top \boldsymbol{u}_i\right] = \sum_{i=1}^{d}\boldsymbol{u}_i^\top \mathbb{E}[\boldsymbol{z}\boldsymbol{z}^\top]\boldsymbol{u}_i$$
$$= \sum_{i=1}^{d}\boldsymbol{u}_i^\top I \boldsymbol{u}_i = \sum_{i=1}^{d}\|\boldsymbol{u}_i\|^2 = d\,.$$

This means that there must exist some $\boldsymbol{z} \in \{-1,+1\}^{2d}$ such that $\sum_{i=1}^{d}\left(\boldsymbol{u}_i^\top \boldsymbol{z}\right)^2 \geq d$ as required. ■

A.3. Proof of Lemma 14

To simplify the notation, let us introduce an auxiliary $m\times 2d$ matrix A, defined as $A_{t,j} = \mathbb{I}\{i(t)=j\}$. In other words, its transpose has the form

$$A^\top = \begin{pmatrix} 1 & \cdots & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots \\ 0 & \cdots & 0 & 1 & \cdots & 1 & 0 & \cdots & 0 & 0 & \cdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 1 & \cdots & 1 & 0 & \cdots \\ & & & & & \cdots & & & & & \end{pmatrix}$$

We will now collect a few useful identities. First, since we defined $\boldsymbol{\beta}$ according to $\beta_i = \sum_{t:i(t)=i}\alpha_i$ and $\boldsymbol{y}$ according to $y_t = z_{i(t)}$, it is easily verified that

$$\boldsymbol{\beta} = A^\top\boldsymbol{\alpha} \quad , \quad \boldsymbol{z} = \frac{2d}{m}A^\top\boldsymbol{y} \quad , \quad \|\boldsymbol{y}\|^2 = \frac{m}{2d}\|\boldsymbol{z}\|^2. \tag{27}$$

Second, we have

$$A^\top K = \frac{m}{2d}G^K A^\top, \tag{28}$$

which holds by the following chain of inequalities (using the definition of G^K, A) for all $j \in \{1,\ldots,2d$ and $t \in \{1,\ldots,m\}$:

$$(A^\top K)_{j,t} = \sum_{p=1}^{m}A_{p,j}K_{p,t} = \sum_{p=1}^{m}\mathbb{I}\{i(p)=j\}\,G^K_{i(p),i(t)}$$
$$= \frac{m}{2d}G^K_{j,i(t)} = \frac{m}{2d}\sum_{p=1}^{2d}G^K_{j,p}\mathbb{I}\{i(t)=p\}$$
$$= \frac{m}{2d}\sum_{p=1}^{2d}G^K_{j,p}A_{t,p} = \frac{m}{2d}(G^K A^\top)_{j,t}.$$

Third, we have

$$K = AG^K A^\top, \tag{29}$$

since for any $t, t' \in \{1, \ldots, m\}$,

$$\begin{aligned}\left(AG^K A^\top\right)_{t,t'} &= \sum_{p,q=1}^{2d} A_{t,p} G^K_{p,q} A_{t',q} \\ &= \sum_{p,q=1}^{2d} \mathbb{I}\left\{i(t) = p, i(t') = q\right\} G^K_{p,q} \;=\; G^k_{i(t),i(t')} \;=\; K_{t,t'}.\end{aligned}$$

We now turn to prove the required lemma. The fact that

$$\frac{1}{m}\sum_{t=1}^{m}\left(\boldsymbol{\alpha}^\top K\boldsymbol{e}_t - y_t\right)^2 + \frac{\lambda}{2}\boldsymbol{\alpha}^\top K\boldsymbol{\alpha} \;=\; \frac{1}{m}\left(\boldsymbol{\alpha}^\top\left(K + \frac{m\lambda}{2}I\right)K\boldsymbol{\alpha} - 2\boldsymbol{y}^\top K\boldsymbol{\alpha} + \|\boldsymbol{y}\|^2\right)$$

is a straightforward exercise. Moreover, using (27), (28) and (29), the above equals

$$\begin{aligned}&\frac{1}{m}\left(\boldsymbol{\alpha}^\top\left(KK + \frac{m\lambda}{2}K\right)\boldsymbol{\alpha} - 2\boldsymbol{y}^\top K\boldsymbol{\alpha} + \|\boldsymbol{y}\|^2\right) \\ &= \frac{1}{m}\left(\boldsymbol{\alpha}^\top\left(AG^KA^\top K + \frac{m\lambda}{2}AG^KA^\top\right)\boldsymbol{\alpha} - 2\boldsymbol{y}^\top AG^KA^\top\boldsymbol{\alpha} + \frac{m}{2d}\|\boldsymbol{z}\|^2\right) \\ &= \frac{1}{m}\left(\boldsymbol{\alpha}^\top A\left(G^KA^\top K + \frac{m\lambda}{2}G^KA^\top\right)\boldsymbol{\alpha} - \frac{m}{d}\boldsymbol{z}^\top G^K\boldsymbol{\beta} + \frac{m}{2d}\|\boldsymbol{z}\|^2\right) \\ &= \frac{1}{2d}\left(\boldsymbol{\alpha}^\top A\left(\frac{2d}{m}G^KA^\top K + d\lambda G^KA^\top\right)\boldsymbol{\alpha} - 2\boldsymbol{z}^\top G^K\boldsymbol{\beta} + \|\boldsymbol{z}\|^2\right) \\ &= \frac{1}{2d}\left(\boldsymbol{\alpha}^\top A\left(G^KG^KA^\top + d\lambda G^KA^\top\right)\boldsymbol{\alpha} - 2\boldsymbol{z}^\top G^K\boldsymbol{\beta} + \|\boldsymbol{z}\|^2\right) \\ &= \frac{1}{2d}\left(\boldsymbol{\alpha}^\top A\left(G^K + d\lambda I\right)G^KA^\top\boldsymbol{\alpha} - 2\boldsymbol{z}^\top G^K\boldsymbol{\beta} + \|\boldsymbol{z}\|^2\right) \\ &= \frac{1}{2d}\left(\boldsymbol{\beta}^\top\left(G^K + d\lambda I\right)G^K\boldsymbol{\beta} - 2\boldsymbol{z}^\top G^K\boldsymbol{\beta} + \|\boldsymbol{z}\|^2\right),\end{aligned}$$

proving the first part of the lemma. To prove the second part, it is enough to show that $(G^K + d\lambda I)\boldsymbol{\beta} = \boldsymbol{z}$. This follows from the following chain of equalities, using (27), (29), and the assumption that $\boldsymbol{\alpha} = \left(K + \frac{\lambda m}{2}I\right)^{-1}\boldsymbol{y}$:

$$\begin{aligned}(G^K + d\lambda I)\boldsymbol{\beta} = (G^K + d\lambda I)A^\top\boldsymbol{\alpha} \;&=\; G^KA^\top\boldsymbol{\alpha} + d\lambda A^\top\boldsymbol{\alpha} \\ &= \frac{2d}{m}A^\top K\boldsymbol{\alpha} + d\lambda A^\top\boldsymbol{\alpha} \;=\; \frac{2d}{m}A^\top\left(K + \frac{\lambda m}{2}I\right)\boldsymbol{\alpha} \\ &= \frac{2d}{m}A^\top\boldsymbol{y} \;=\; \boldsymbol{z}.\end{aligned}$$