RFN: A Random-Feature Based Newton Method for Empirical Risk Minimization in Reproducing Kernel Hilbert Spaces

Ting-Jui Chang and Shahin Shahrampour, Senior Member, IEEE

Abstract—In supervised learning using kernel methods, we often encounter a large-scale finite-sum minimization over a reproducing kernel Hilbert space (RKHS). Large-scale finite-sum problems can be solved using efficient variants of Newton method, where the Hessian is approximated via sub-samples of data. In RKHS, however, the dependence of the penalty function to kernel makes standard sub-sampling approaches inapplicable, since the gram matrix is not readily available in a low-rank form. In this paper, we observe that for this class of problems, one can naturally use kernel approximation to speed up the Newton method. Focusing on randomized features for kernel approximation, we provide a novel second-order algorithm that enjoys local superlinear convergence and global linear convergence (with high probability). We derive the theoretical lower bound for the number of random features required for the approximated Hessian to be close to the true Hessian in the norm sense. Our numerical experiments on real-world data verify the efficiency of our method compared to several benchmarks.

Index Terms—Newton method, optimization algorithms, risk minimization, Hessian approximation, random features.

I. INTRODUCTION

At the heart of many supervised machine learning problems, a learner must solve the following risk minimization

$$\min_{w \in \mathbb{R}^d} \left\{ F(w) \doteq \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(x_i; w)) + \lambda R(w) \right\}, \tag{1}$$

where \(\{(x_i, y_i)\}_{i=1}^{n}\) are input-output data samples generated independently from an unknown distribution, \(\ell\) is a task-dependent (convex) loss function, and \(R\) is a regularizer. Furthermore, \(f\) represents a certain function class, parameterized by \(w\), on which the learner wants to minimize the risk. As an example, for linear models we simply have \(f(x; w) = \mathbf{x}^\top w\).

First-order optimization algorithms have been widely used to solve large-scale optimization problems of form (1) (see [1] for a recent survey). Relying solely on the gradient information, these methods converge to (local) optima. However, second-order algorithms employ the curvature information to properly re-scale the gradient, resulting in more appropriate directions and much faster convergence rates. As an example, in the unconstrained optimization, Newton method pre-multiplies the gradient by the Hessian inverse at each iteration. It is quite well-known that under some technical assumptions, Newton method can achieve a locally superlinear convergence rate for strongly convex problems (see e.g., Theorem 1.2.5 in [2]). However, the cost of Hessian inversion is the major drawback of Newton method in practice.

To improve the (per iteration) time complexity, various approaches have been explored in the literature for approximately capturing the Hessian information. Popular methods in this direction include sub-sampling the Hessian matrix [3], [4], [5], sketching techniques [6], as well as quasi-Newton methods [7], [8], [9], [10] and its stochastic variants [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22].

Nevertheless, when the function class of \(f\) in (1) is a reproducing kernel Hilbert space (RKHS), due to the special structure of the problem, some of the aforementioned methods are not directly applicable and need adjustments.

A. Risk Minimization in RKHS

In this paper, we restrict our attention to risk minimization in the case that the function \(f\) in (1) belongs to a RKHS. In particular, consider a symmetric function \(k(\cdot, \cdot)\) such that

$$\sum_{i,j=1}^{n} \alpha_i \alpha_j k(x_i, x_j) \geq 0,$$

for \(\alpha = [\alpha_1, \ldots, \alpha_n]^\top \in \mathbb{R}^n\). Then, \(k(\cdot, \cdot)\) is called a positive (semi-)definite kernel and can define a Hilbert space \(\mathcal{H}\) where \(f(x; w) = \sum_{i=1}^{n} w_i k(x, x_i)\). This class of functions forms the basis of kernel methods that are powerful tools for data representation and are commonly used in machine learning and signal processing [23], [24]. In this scenario, the objective function in (1) takes the following form

$$F(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \sum_{j=1}^{n} w_j k(x_i, x_j)) + \frac{\lambda}{2} \|f\|^2_{\mathcal{H}}, \tag{2}$$

where the regularizer in (1) is the RKHS norm. Let us denote the kernel (gram) matrix as \([K]_{ij} = k(x_i, x_j)\). The definition
of inner product in RKHS immediately implies that \( \|f\|_H^2 = w^T Kw \) (see e.g., page 62 of [23]). Then, assuming that \( \ell \) is twice-differentiable, the Hessian of the objective function in (2) can be calculated as follows

\[ H(w) = \nabla^2 F(w) = \frac{1}{n} KD(w)K + \lambda K, \tag{3} \]

where \( D(w) \in \mathbb{R}^{n \times n} \) is a diagonal matrix defined as

\[ [D(w)]_{ii} = \ell'' \left( y_i, \sum_{j=1}^{n} w_j k(x_i, x_j) \right). \tag{4} \]

The inversion of the Hessian matrix requires an order of \( n^3 \) operations, which is costly. On the other hand, observe that in (3), the diagonal structure of \( D(w) \) and the symmetry of \( K \) together imply that \( KD(w)K \) can be trivially represented as a sum of rank-one matrices. However, \( K \), which appears as a result of the regularization term, may be dense and not readily available in a low-rank form. In other words, decomposing \( K \) to a low-rank matrix also requires effort, so we cannot directly apply sub-sampling Newton techniques, such as those in [3, 4, 5] to optimize the objective function (2). Furthermore, the Hessian is of size \( n \), which implies that even without the regularization term any sub-sampling scheme results in a singular matrix as the approximation. This naturally raises the following question, which we pursue in this paper:

**Problem 1:** Given the explicit connection of the Hessian (3) to the gram matrix \( K \), can we use kernel approximation techniques to improve the per iteration time complexity of the Newton method?

### B. Our Contributions

In this paper, we answer to Problem 1 in the affirmative by providing the following contributions:

- **Focusing on composite kernels, we apply the idea of randomized features for kernel approximation [25] to approximate the Hessian (3). Our algorithm is thus dubbed Random-Feature based Newton (RFN). The detailed derivation of RFN is explained in Section II.**

- **The key to our technical analysis is Lemma 1, which shows that when enough random features are sampled, the approximate Hessian is close to \( H(w) \) in the spectral norm sense. Our analysis relies on matrix concentration inequalities and explicitly derives the theoretical lower bound of the number of random features required to reach a target Hessian approximation error.**

- **We prove that RFN enjoys local superlinear convergence and global linear convergence in the high probability sense (Section III).**

- **Our numerical experiments on real-world datasets (Section IV) provide a performance comparison among RFN, classical Newton, L-BFGS, and a variation of sub-sampled Newton methods (as they are not directly applicable to (2)). We illustrate that RFN achieves a superior loss vs. run-time performance against its competitors.**

The omitted proofs are included in the Appendix (Section A).

### C. Related Literature

Inspired by the success of stochastic first-order algorithms for large-scale data analytics, the stochastic forms of second-order algorithms have received more attention in the recent literature. In this section, we review several stochastic second-order methods, and we split them into two categories: quasi-Newton methods and second-order Hessian-based methods.

- **Quasi-Newton Methods:** Instead of performing the expensive computation of the Newton step (which involves Hessian inversion), quasi-Newton methods approximate the Hessian by using the information obtained from gradient evaluations. BFGS algorithm [7, 8, 9, 10] is a seminal work of this type. Recent works [11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22] in this area have focused on stochastic quasi-Newton methods to obtain curvature information in a computationally inexpensive manner. Mokhtari and Ribeiro [12] propose a stochastic regularized version of BFGS algorithm to avoid the problem of singular curvature estimates. Byrd et al. [13] apply sub-sampled Hessian-vector products to stabilize the curvature estimation based on a stochastic limited memory BFGS (L-BFGS) algorithm. Another advancement in this line of work is provided by Moritz et al. [14] who propose an algorithm based on L-BFGS by incorporating ideas from stochastic variance-reduced gradient (SVRG) to achieve linear convergence to the optimum. Gower et al. [19] employ the idea of SVRG to propose a stochastic block BFGS algorithm using sketching techniques and show practical speed-ups for common machine learning problems. Based on the algorithm structure of [14], Zhao et al. [20] propose a coordinate transformation framework to analyze stochastic L-BFGS type algorithms and present improved convergence rates and computational complexities. Using momentum for L-BFGS Chang et al. [21] prove an accelerated linear convergence rate with better dependence on the condition number. For non-convex optimization, Zhang et al. [22] propose a novel stochastic quasi-Newton method with an improved stochastic first-order oracle complexity for reaching an \( \epsilon \) first-order stationary point.

- **Second-Order Hessian-based Methods:** The appealing feature of Newton method is its local superlinear convergence rate. However, there are two main issues for the implementation of the classical Newton method: the cost of Hessian construction, and the cost of Hessian inversion. For example, in application to the family of generalized linear models (GLMs) involving an \( n \times d \) data matrix, the computation of the full Hessian costs \( O(n d^2) \) and the matrix inversion takes \( O(d^3) \) time. This high cost, especially for large-scale applications, has motivated researchers to apply randomization techniques, and thus sub-sampled Newton methods have gained a good deal of attention recently.

In [26, 27], the authors establish the convergence of the modified Newton method with sub-sampled Hessian. Under a similar setting to [26, 27], Wang et al. [28] provide modifications in order to get better estimated Hessian and time cost performance. Within the context of deep neural networks,
Martens [29] proposes a sub-sampled Gauss-Newton method for the training and studies the choice of the regularization parameter.

Erdogdu and Montanari [30] propose a Newton-like method, where the Hessian is approximated by sub-sampling the true Hessian and computing its truncated eigenvalue decomposition. Their work establishes non-asymptotic local convergence rates for the uniform sub-sampling of the Hessian. Pilanci and Wainwright [6] propose another Hessian approximation method, called Newton sketching, which approximates the true Hessian through random projection matrices. This method is applicable for the case where the square root of the full Hessian is available, and the best complexity results are achieved when the randomized Hadamard transform is used.

Authors of [3], [4] analyze the global and local convergence rates for sub-sampled Newton methods with different sampling rates for gradient and Hessian approximations. Bollapragada et al. [3] show the convergence results in expectation, whereas [4] provides high probability guarantees by applying matrix concentration inequalities [31], [32]. The work of [4] further relaxes a common assumption in the literature: though the objective function is assumed to be strongly convex, the individual functions are only weakly convex. Along this line of works, Xu et al. [5] build the approximated Hessian by applying non-uniform sampling based on the data matrix to get better dependence on problem specific quantities.

Agarwal et al. [33] propose a method to compute an unbiased estimator of the inverse Hessian based on the power expansion of the Hessian inverse. The method achieves a time complexity scaling linearly with the size of variables. This is followed by an improved and simplified convergence analysis in [34].

---

**First-Order Methods for Risk Minimization in RKHS:**

Though our work focuses on second-order optimization, it is worth noting that first-order methods have been explored for risk minimization in RKHS. In [35], the idea of functional stochastic gradient descent (FSGD) is applied and the convergence rates are quantified for models with and without truncation. Following the idea of FSGD, the authors of [36] control the computational cost by projecting the updated function to a sub-space selected using kernel orthogonal matching pursuit. Moreover, [37] provides a scalable approach, called doubly stochastic gradient descent, which further approximates the stochastic gradient using random features.

The main distinction of our work with the literature is that we consider risk minimization in RKHS with second-order methods, where there are explicit connections between the Hessian and the gram matrix. We leverage this fact to approximate the Hessian, and we provide theoretical guarantees for the RFN method. Our prior work [38] studied only the global convergence of RFN for simple kernels, but this work provides a comprehensive analysis on both global and local convergence rates for composite kernels explained in Section II-C. This allows us to achieve a sharper theoretical lower bound for the number of random features required for convergence. Furthermore, we illustrate the efficiency of RFN compared to other benchmarks on real-world datasets.

### Table I

| Method             | Complexity per iteration |
|--------------------|--------------------------|
| Newton             | $O(n^3)$                 |
| SSNCG (exact)      | $O(|I|^2 n + |I| n \sqrt{\kappa_{SSN}} \log \epsilon^{-1})$ |
| SSNCG (inexact)    | $O(m^2 n + m^3)$         |

($n$: # of data points. $I$: The sub-sampled index set. $m$: # of sampled random features).

## II. Random-Feature Based Newton Method

### A. Notation

We denote by $\text{Tr}[:]$ the trace operator, by $\langle \cdot, \cdot \rangle$ the standard inner product, by $\|\cdot\|$ the spectral (respectively, Euclidean) norm of a matrix (respectively, vector), by $O(\cdot)$ (respectively, $\Omega(\cdot)$) the Big O (respectively, Big Omega) notation in complexity theory, and by $E[\cdot]$ the expectation operator. Boldface lowercase variables (e.g., $a$) are used for vectors, and boldface uppercase variables (e.g., $A$) are used for matrices. $[A]_{ij}$ denotes the $i$-th entry of matrix $A$. $\lambda_{\min}(A)$ (respectively, $\lambda_{\max}(A)$) denotes the smallest (respectively, largest) eigenvalue of matrix $A$. The symbol “$\preceq$” is used for matrix inequality and $A \preceq B$ implies that the matrix $(B - A)$ is positive semi-definite. $|I|$ represents the cardinality of the set $I$. The vectors are all in column form.

### B. Background on Random Features for Kernel Approximation

As discussed in the introduction, the Hessian of the objective function in (2) can be written as $H(w) = \frac{1}{n} K D(w) K + \lambda K$, where $[K]_{ij} = k(x_i, x_j)$ is the gram matrix. The Hessian is a square matrix of size $n$, and a plain inversion of that in Newton method introduces a prohibitive cost of $O(n^3)$. As $n$ is the number of data points, this cost is specifically expensive for Big Data problems.

To find a low-rank representation of Hessian, the key is to approximate the gram matrix $K \in \mathbb{R}^{n \times n}$, which in general is dense. An elegant method for kernel approximation, called random Fourier features, was introduced by Rahim and Recht [25]. Let $p(\omega)$ be a probability density with support $\Omega \subseteq \mathbb{R}^d$. Consider any kernel function with the following integral form

$$k(x, x') = \int_{\Omega} \phi(x, \omega) \phi(x', \omega) p(\omega) d\omega,$$

where $\phi(x, \omega) : \mathbb{R}^d \rightarrow \mathbb{R}$ is a feature map. We can immediately see from (5) that the kernel function can be approximated via Monte-Carlo sampling as

$$k(x, x') \approx \frac{1}{m} \sum_{s=1}^{m} \phi(x, \omega_s) \phi(x', \omega_s),$$

where $\{\omega_s\}_{s=1}^{m}$ are independent samples from the density $p(\omega)$ and are called random features. There exist many kernels taking the form (5), including shift-invariant kernels [25] and dot product (e.g., polynomial) kernels [39] (see Table I in [40] for an exhaustive list). Gaussian kernel, for example, can be approximated using $\phi(x, \omega) = \sqrt{2} \cos(\omega^T x + b)$ where $\omega$
follows a Gaussian distribution and $b$ has a uniform distribution on $[0, 2\pi]$. It is common to assume that the feature map is uniformly bounded (as evident in the case of cosine). For simplicity, we assume $|\phi(x, \omega)| \leq 1$ for any $x, \omega \in \mathbb{R}^d$. Let us now define

$$z(\omega) \triangleq [\phi(x_1, \omega), \ldots, \phi(x_n, \omega)]^\top. \tag{7}$$

Then, based on (6), the gram matrix $K$ can be approximated with $ZZ^\top$, where $Z \in \mathbb{R}^{n \times m}$ is the following matrix

$$Z \triangleq \frac{1}{\sqrt{m}} [z(\omega_1), \ldots, z(\omega_m)]. \tag{8}$$

When $m < n$ in above, $ZZ^\top$ has a lower rank than $n$, saving computational cost when used to find the Newton direction. However, unlike the finite sum problem considered in [4], where $n$ (# of data points) can be much greater than $d$ (# of parameters to learn), in the optimization problem for RKHS (2) the decision variable is also of the size $n$. Therefore, only replacing $K$ by a low-rank form $ZZ^\top$ leads to an approximated Hessian that is singular. Thus, we only focus on positive definite kernels of the composite type, as described next.

**C. Composite Kernel Function**

We consider gram matrices built on a composite kernel with the following form:

$$k(x, x') = k_1(x, x') + k_2(x, x'), \tag{9}$$

where $k_1(x, x')$ can be expressed as (5), and $k_2(x, x') = \mu \mathbb{I}(x = x')$, where $\mathbb{I}$ is the indicator function. From this definition, we have that $K = K_1 + \mu I$, and the random feature method is applied to approximate $K_1$. Let us discuss our current choice of composite kernels in (9). On one hand, we can only hope to approximate the gram matrix with a low-rank surrogate, when its eigen-decay is fast. On the other hand, if the eigenvalue sequence of $K$ decays to zero fast, the Hessian becomes singular (as evident in (3)), and the problem becomes ill-conditioned. The composite kernel structure in (9) allows us to control the smallest eigenvalue of the gram matrix by $\mu$ and keeps the Hessian positive definite. Notice that the positive definiteness assumption for Hessian is quite standard in the analysis of second-order methods (see e.g., [3], [4], [5], [28]). In (9), the matrix $K_1$ can be formed using any kernel that can be approximated with random features (e.g., Gaussian kernel, Laplace kernel, Cauchy kernel, dot product kernels, etc.).

**D. Algorithm: Random-Feature Based Newton (RFN)**

RFN leverages the random feature method for Hessian approximation to execute the Newton method. Following (3) and using the approximation $K = ZZ^\top + \mu I$, where $Z$ is given in (8), we derive the following approximated Hessian

$$\hat{H}(w) = \frac{1}{n} (ZZ^\top + \mu I) D(w) (ZZ^\top + \mu I) + \lambda (ZZ^\top + \mu I). \tag{10}$$

Define $C \triangleq (ZZ^\top + \mu I)$. By matrix inversion lemma, $\hat{H}^{-1}(w)$ can be written as

$$\hat{H}^{-1}(w) = C^{-1} - (\lambda n D^{-1}(w) + C)^{-1}, \tag{11}$$

where we can observe that both $C$ and $(\lambda n D^{-1}(w) + C)$ are expressed as the sum of a low-rank matrix and a diagonal matrix. Therefore, denoting the diagonal matrix $D_\mu(w) \triangleq (\lambda n D^{-1}(w) + \mu I)$ and substituting $C = (ZZ^\top + \mu I)$ in above, we have that

$$\hat{H}^{-1}(w) = \frac{1}{\lambda} \left[ (I - Z(\mu I + Z^\top Z)^{-1}Z^\top) - D_\mu^{-1}(w) + D_\mu^{-1}(w)Z(I + Z^\top D_\mu^{-1}(w)Z)^{-1}Z^\top D_\mu^{-1}(w) \right], \tag{12}$$

which can reduce the time complexity of computing $H^{-1}(w)\nabla F(w)$ from $O(n^3)$ to $O(m^2 n + m^3)$. Then, we can use $H^{-1}(w)$ to perform a computationally efficient Newton update, as described in Algorithm 1.

**E. Adjustment of Sub-Sampled Newton Methods**

We now revisit sub-sampled Newton methods to elaborate on the differences between RFN and these methods. Sub-sampled Newton algorithms often end up working with Hessians of the form

$$H(w) = \frac{1}{n} \sum_{i=1}^{n} a_i(w)a_i^\top(w) + Q(w),$$

where $a_i(w)$ is a vector and $Q(w)$ is a data-independent matrix (see e.g., [5]). Given this structure, if we randomly select a subset $I$ of data points, the Hessian can be easily approximated via

$$\tilde{H}(w) = \frac{1}{|I|} \sum_{i \in I} a_i(w)a_i^\top(w) + Q(w).$$

This would reduce the time cost of Hessian construction by a factor of $|I|/n$. Furthermore, since the approximated Hessian still consists of sum of rank-one matrices (in the data-related part), one can apply conjugate gradient (CG) method when computing the newton step to further speed up the process [3], [4], [5].

However, it turns out that the Hessian of the objective function (2), i.e., $\frac{1}{n} KD(w)K + \lambda K$, cannot be directly handled
by sub-sampling, since the regularizer in this case is indeed data-dependent, and writing it as a sum of rank-one matrices requires a low-rank decomposition of $K$. In this case, sub-sampled Newton techniques can be adjusted using the Nyström method for column sampling (see e.g., [41] for a review on Nyström method). Let $\mathcal{V} = \{1, \ldots, n\}$ and $\mathcal{I}$ denote a (random) subset of $\mathcal{V}$. Furthermore, denote by $K(\mathcal{A}, \mathcal{B})$ the sub-matrix of $K$ with rows in $\mathcal{A} \subseteq \mathcal{V}$ and columns in $\mathcal{B} \subseteq \mathcal{V}$. Then, the gram matrix $K$ can be approximated as

$$K \approx K_1(\mathcal{V}, I)K_1(\mathcal{I}, \mathcal{I})^\dagger K_1(\mathcal{I}, \mathcal{V}) + \mu I,$$  \hspace{1cm} (13)

where $\dagger$ denotes the pseudo-inverse. Notice that the first term of the Hessian can be trivially sub-sampled since

$$KD(w)K = \sum_{i=1}^n |D(w)|_i K(\mathcal{V}, i)K(i, \mathcal{V}).$$  \hspace{1cm} (14)

In the similar spirit as [3], [4], [5], we call this algorithm SSNCG, as it is a sub-sampled Newton method, where CG is used to find the Newton step.

**Remark 1:** Since the focus of this work is on random features, we only compare RFN to the case that $\mathcal{I}$ is chosen uniformly at random, i.e., the method would be a variant of uniform sub-sampled Newton [4]. Nonuniform sampling methods give better approximations of the kernel at the cost of modifying the uniform sampling distribution. We refer the reader to Table I in [41] for various guarantees on the approximation quality via different sampling schemes.

**F. Comparison of Time Complexity**

Before stating our main results, we present the time complexity of finding the Newton step for four methods: Newton method, sub-sampled Newton method solved with CG exactly, and sub-sampled Newton method solved with CG inexactly up to $\epsilon$ error, and RFN. The number of random features used by RFN is denoted by $m$. The size of the sub-sampled data points is $|\mathcal{I}|$, as discussed in Section II-E. $\kappa_{SSN}$ represents the upper bound of the condition number of the Hessian generated from sub-sampled data. Note that for each method, the reported complexity excludes the cost of obtaining the gradient, because that cost is the same for all methods.

**III. THEORETICAL RESULTS**

In this section, we study the convergence properties of RFN. In order to establish our results, we need to prove that the approximated Hessian mimics the original Hessian with high probability, which is shown in Lemma 1. Then, we can show the global and local convergence of RFN in Sections III-B and III-C, respectively.

**A. Norm Bound for Hessian Approximation**

Throughout the paper, we adhere to the following assumptions:

**Assumption 1 (Bounded Eigenvalues of Hessian):** The objective function $F$ is twice-differentiable, $\gamma$-strongly convex, and $L$-smooth. The smallest and largest eigenvalues of the Hessian are bounded as follows

$$\gamma I \preceq H(w) \preceq LI, \quad \forall w \in \mathbb{R}^n,$$  \hspace{1cm} (15)

and also $D(w)$ satisfies

$$\|D(w)\| \leq \lambda_1 < \infty, \quad \forall w \in \mathbb{R}^n.$$  \hspace{1cm} (16)

Condition (16) is satisfied for common loss functions (e.g., quadratic loss and logistic loss). Furthermore, if we solve the risk minimization (2) with a positive definite kernel, the gram matrix $K$ would be positive definite, and therefore, Assumption 1 is satisfied. We also define the condition number $\kappa$ of the Hessian matrix as follows:

$$\kappa \triangleq L \gamma.$$  \hspace{1cm} (17)

**Assumption 2 (Lipschitz Continuity of Hessian):** The Hessian of the objective function $F$ is Lipschitz continuous, i.e., there exists a constant $M > 0$, such that

$$\|H(w_1) - H(w_2)\| \leq M \|w_1 - w_2\|, \forall w_1, w_2 \in \mathbb{R}^n.$$  \hspace{1cm} (18)

Assumption 2 is commonly used in the literature to establish local superlinear convergence of sub-sampled Newton methods (see e.g., [3], [4]). The assumption is used to prove the local superlinear convergence of the original Newton method as well (see e.g., Theorem 1.2.5 in [2]).

We are now ready to show that with high probability the approximated Hessian is close enough to the original Hessian when large enough random features are sampled.

**Lemma 1 (Spectrum Preserving Inequality):** Suppose Assumption 1 holds. Define

$$\zeta \triangleq \lambda_1 \left\| \frac{ZZ^\top + \mu I}{n} \right\| + \lambda_1 \left\| \frac{K_1 + \mu I}{n} \right\|.$$  \hspace{1cm} \hspace{1cm} (19)

For $0 < \delta < 1$, if

$$m = \Omega\left(\frac{\zeta n \|K_1\|^2}{\epsilon^2 \gamma} \log \frac{\text{Tr}(K_1)}{\|K_1\| \delta}\right),$$

random features are sampled from $p(\omega)$, we have that

$$\text{Pr}\left(\left\| \hat{H}(w) - H(w) \right\| \leq \epsilon \gamma\right) \geq 1 - \delta.$$  \hspace{1cm} (20)

**Corollary 2:** Under assumptions of Lemma 1, with probability at least $1 - \delta$, we have

$$\left\|[1 - \epsilon \gamma] \cdot I \preceq \hat{H}(w) \preceq [(1 + \epsilon)L] \cdot I\right\|,$$  \hspace{1cm} (21)

where $\hat{H}(w)$ is ensured to be positive-definite when $\epsilon < 1$.

**Proof:** From Lemma 1, we have $\|\hat{H}(w) - H(w)\| \leq \epsilon \gamma$, which implies the following relationship

$$H(w) - \epsilon \gamma \cdot I \preceq \hat{H}(w) \preceq H(w) + \epsilon \gamma \cdot I \Rightarrow (1 - \epsilon)H(w) \preceq \hat{H}(w) \preceq (1 + \epsilon)H(w) \Rightarrow [(1 - \epsilon)] \cdot I \preceq \hat{H}(w) \preceq [(1 + \epsilon)L] \cdot I.$$  \hspace{1cm} (22)

**Discussion on efficiency of RFN vs. Newton:** Based on the result in Lemma 1, if $\mu = O(n^p)$ for any $0 < p \leq 1$, both $\|ZZ^\top + \mu I\|$ and $\|K_1 + \mu I\|$ are $O(1)$ since both $\|ZZ^\top\|$ and $\|K_1\|$
are at most $O(n)$. Therefore, disregarding the log factor, the lower bound for random features (with respect to problem-dependent quantities) is $\Omega\left(\frac{n\|K_1\|}{\gamma^2}\right)$. Notice that the Hessian $H(w)$ is lower bounded as follows

$$\lambda_\mu I \preceq \lambda K \preceq H(w),$$

so letting $\gamma = \lambda_\mu$, we can get

$$\frac{n\|K_1\|}{\gamma^2} = \frac{n\|K_1\|}{\lambda^2\mu^2} \leq \frac{n^2}{\lambda^2\mu^2}.$$ 

Therefore, if $\mu = O(n^p)$ for any $\frac{1}{2} < p \leq 1$, the required number of random features is $o(n)$, and based on Table I, RFN has a smaller time complexity compared to Newton method. Notice that the relationship above only provides a sufficient condition on $\mu$. Otherwise, the computational advantage can be guaranteed anytime that we have $\frac{n\|K_1\|}{\gamma^2} = o(n)$, where the choice of $\mu$ affects $\gamma$.

### B. Global Linear Convergence

In this section, we establish the global convergence of our method using the approximation inequality derived in Lemma 1. Recall that the RFN update is written as

$$w_{t+1} = w_t + \alpha_t p_t,$$

(21)

where $p_t = -[\breve{H}(w_t)]^{-1} \nabla F(w_t)$ and $\alpha_t$ is selected by Armijo-type line search such that

$$F(w_t + \alpha_t p_t) \leq F(w_t) + \alpha_t \beta p_t^T \nabla F(w_t),$$

(22)

for some $\beta \in (0, 1)$. In what follows, we denote

$$w^* = \arg\min_{w \in \mathbb{R}^n} F(w).$$

**Theorem 3 (Global Convergence):** Let Assumption 1 hold. If we update $w_t \in \mathbb{R}^n$ using RFN algorithm, where $H(w_t)$ is constructed by sampling

$$m = \Omega\left(\frac{\zeta^2 n\|K_1\|}{\epsilon^2} \log \frac{\lambda}{\|K_1\| \delta}\right),$$

random features, we have

$$F(w_{t+1}) - F(w^*) \leq (1 - \rho_t)(F(w_t) - F(w^*)), $$

(23)

with probability at least $1 - \delta$, where

$$\rho_t \triangleq \frac{2\alpha_t \beta}{(1 + \epsilon) \kappa}.$$ 

Moreover, the step size $\alpha_t \leq 2(1 - \beta)(1 - \epsilon)/\kappa$ is sufficient to pass the line search.

Theorem 3 draws a connection between the precision of approximated Hessian and the convergence speed of RFN. If the precision parameter $\epsilon$ is set to a small number (high precision), the step size $\alpha_t$ is upper bounded by a larger number, which implies a more aggressive update. Also, the parameter $\rho_t$ tends to be larger (i.e., faster convergence) if the approximated Hessian is closer to the original Hessian.

### C. Local Superlinear Convergence

The Newton method is particularly appealing for its local convergence property, resulting in quadratic rates for strongly convex and smooth problems. In this section, we discuss the local convergence behavior of RFN. We use the unit step size, i.e., $\alpha_t = 1$. In the following lemma, we provide an error recursion for the update $w_t$ using Lemma 1.

**Lemma 4 (Error Recursion):** Let Assumptions 1–2 hold. If we update $w_t \in \mathbb{R}^n$ using RFN algorithm (with $\alpha_t = 1$), where $H(w_t)$ is constructed by sampling

$$m = \Omega\left(\frac{\zeta^2 n\|K_1\|}{\epsilon^2} \log \frac{\lambda}{\|K_1\| \delta}\right)$$

random features, we have

$$\|w_{t+1} - w^*\| \leq \nu \|w_t - w^*\| + \eta \|w_t - w^*\|^2,$$

(24)

with probability at least $1 - \delta$, where

$$\nu \triangleq \frac{\epsilon}{(1 - \epsilon)} \quad \text{and} \quad \eta \triangleq \frac{M}{2(1 - \epsilon)^2\gamma}.$$ 

The above lemma helps with establishing the linear local rate, as discussed in the following theorem.

**Theorem 5 (Local Linear Convergence):** Let Assumptions 1–2 hold. Suppose $\epsilon$ is chosen such that $0 < \nu < 1$ and $\rho$ is selected as

$$\nu < \rho < 1.$$ 

Assume the initial point $w_0$ satisfies

$$\|w_0 - w^*\| \leq \frac{\rho - \nu}{\eta},$$

(25)

where $\eta$ is defined in Lemma 4. If we update $w_t \in \mathbb{R}^n$ using RFN algorithm with $\alpha_t = 1$, where $H(w_t)$ is constructed by sampling

$$m = \Omega\left(\frac{\zeta^2 n\|K_1\|}{\epsilon^2} \log \frac{\lambda}{\|K_1\| \delta}\right)$$

random features, we have the following linear convergence

$$\|w_t - w^*\| \leq \rho \|w_{t-1} - w^*\|, \quad t = 1, \ldots, t_0$$

(26)

with probability at least $(1 - t_0 \delta)$.

While Theorem 5 establishes the local linearity rate, we are more interested in stronger local convergence guarantees. If the precision of the Hessian approximation increases through iterations, it is expected that the algorithm can converge faster than the linear rate. In the next theorem, we show that if the Hessian approximation error decreases geometrically, RFN converges superlinearly and asymptotically achieves the same local rate as the Newton method.

**Theorem 6 (Local Superlinear Convergence):** Let the assumptions of Theorem 5 hold. Also, for each iteration $t = 0, 1, \ldots, t_0$ define the quantities $\epsilon_t$, $\nu_t$ and $\eta_t$ as follows

$$\epsilon_t = \rho^t \epsilon, \quad \nu_t = \frac{\epsilon_t}{(1 - \epsilon_t)} \quad \text{and} \quad \eta_t = \frac{M}{2(1 - \epsilon_t)^2\gamma},$$

where $\epsilon_t \leq 0.5$ and $\rho > 1$. If the Hessian approximation error decreases geometrically, i.e., $\epsilon_t \leq 0.5$ and $\rho > 1$, then the RFN algorithm converges superlinearly.
Assume that \( w_0 \) satisfies (25) with \( \rho, \nu_0 \) and \( \eta_0 \). Then, RFN algorithm (with \( \alpha_t = 1 \)) achieves the following superlinear convergence

\[
\| w_t - w^* \| \leq \rho^t \| w_{t-1} - w^* \|, \quad t = 1, \ldots, t_0, \tag{27}
\]

with probability at least \((1 - t_0 \delta)\).

\textbf{Proof:} The proof follows a similar argument to that of Theorem 7 in [4]. Based on Lemma 4, for each iteration \( t \), sampling \( m \) random features such that the Hessian approximation error is at most \( \gamma \varepsilon_t \), we have

\[
\| w_{t+1} - w^* \| \leq \nu_t \| w_t - w^* \| + \eta_t \| w_t - w^* \|^2.
\]

Note that, by \( \varepsilon_t = \rho^t \varepsilon \), it follows that

\[
\nu_t \leq \rho^t \nu_0, \quad \eta_t \leq \eta_t - 1.
\]

Define \( \Delta_t \equiv w_t - w^* \). For \( t = 0 \), by assumption on \( \rho, \nu_0, \) and \( \eta_0 \) (Theorem 5), we have

\[
\| \Delta_1 \| \leq \nu_0 \| \Delta_0 \| + \eta_0 \| \Delta_0 \|^2 \leq \rho \| \Delta_0 \|.
\]

Assume (27) holds up to iteration \( t \). For \( t + 1 \), we get

\[
\| \Delta_{t+1} \| \leq \nu_t \| \Delta_t \| + \eta_t \| \Delta_t \|^2 \leq \rho \nu_0 \| \Delta_t \| + \eta_0 \| \Delta_t \|^2 \leq \rho \nu_0 \| \Delta_t \| + \eta_0 \| \Delta_t \|^2.
\]

By induction hypothesis, we have \( \| \Delta_{t-1} \| \leq \| \Delta_0 \| \), and

\[
\| \Delta_t \| \leq \rho^t \| \Delta_{t-1} \| \leq \rho^t \left( \frac{\rho - \nu_0}{\eta_0} \right).
\]

Therefore, it follows that \( \| \Delta_{t+1} \| \leq \rho^{t+1} \| \Delta_t \| \). \hfill \Box

\textbf{Remark 2:} We remark that in Theorem 6, the precision \( \varepsilon_t \) varies over iterations, so re-sampling random features in line 4 of RFN is necessary. However, to derive results of Theorems 3 and 5, since the number of random features \( m \) is iteration-independent, we can execute line 4 outside the iteration loop, which saves on computation. This can be seen from (12), where the first term only depends on data and can be re-used for all iterations.

IV. NUMERICAL EXPERIMENTS

We now provide numerical experiments to illustrate the performance of the random-feature based Newton method. We consider regularized kernel logistic regression

\[
F(w) = \frac{1}{n} \sum_{i=1}^{n} \log \left( 1 + \exp \left( -y_i \sum_{j=1}^{n} k(x_i, x_j) w_j \right) \right) + \frac{\lambda}{2} \| w \|_K^2,
\]

where \( K = K_1 + \mu I \) is a composite positive-definite kernel, and \( \| w \|_K^2 = w^T K w \) is the norm with respect to the gram matrix. We choose \( k_1(x, x') = \exp(-\| x - x' \|^2 / 2\sigma^2) \) to be a Gaussian kernel. We apply the random feature technique to approximate the Gaussian kernel.

- \textbf{Benchmark algorithms:} We compare RFN (as described in Algorithm 1) with the algorithms listed below:

1) \textbf{Newton:} The original Newton method \( w_{t+1} = w_t + \alpha_t p_t \), where \( p_t \) is the exact solution of the linear system \( \mathbf{H}(w_t) p_t = -\nabla F(w_t) \).

2) \textbf{SSNCG:} The sub-sampled Newton method with conjugate gradient update \( w_{t+1} = w_t + \alpha_t p_t \), where \( p_t \) is computed by solving the linear system \( \mathbf{H}_{SSN}(w_t) p_t = -\nabla F(w_t) \) up to a high precision using the CG method. The approximated Hessian \( \mathbf{H}_{SSN}(w_t) \) is generated based on the description in Section II-E, and the Hessian is approximated as follows

\[
\mathbf{H}_{SSN}(w) = \frac{1}{|I|} \sum_{i \in I} [D(w)]_{ii} K(V, i) K(i, V)
+ \lambda (K_1(V, V) K_1(I, I) + K_1(I, V) + \mu I),
\]

where \( I \) is a random subset of \( \{1, \ldots, n\} \). SSNCG is the adaption of algorithms in [3], [4] as described in Section II-E.

3) \textbf{L-BFGS:} Limited-memory BFGS which approximates the Newton direction using the first-order information. Here, we implement BFGS using the history of the past 50 updates of \( w_t \) and \( \nabla F(w_t) \).

4) \textbf{GD:} The plain gradient descent method where \( p_t = -\nabla F(w_t) \).

For all these methods, the step size \( \alpha_t \) is determined by Armijo backtracking line search, and the full gradient \( \nabla F(w_t) \) is used. For SSNCG, the linear system is solved approximately to achieve \( 10^{-6} \) relative error.

- \textbf{Hyper-parameters of the empirical risk:} There are three hyper-parameters in the regularized kernel logistic regression with a composite kernel: \( \sigma, \mu \) and \( \lambda \). \( \lambda \) determines the rate by which we impose the RKHS norm as a penalty function. When \( \lambda \) is small, the model fits the training data. On the other hand, \( \sigma \), known as the bandwidth of a Gaussian kernel, determines the kernel generalization characteristics. When \( \sigma \) is very small, the gram matrix approaches the identity matrix, and the corresponding distribution of random features has a larger variance, which requires more random feature samples to approximate the kernel function. However, for a Gaussian kernel with a large bandwidth, the gram matrix eigenvalues decay faster. \( \mu \) controls the relative importance of \( k_1(x, x') \) and as \( \mu \) grows larger, the Gaussian kernel plays a less important role for the composite kernel.

- \textbf{Datasets:} We apply all of the methods on three datasets from the UCI Machine Learning Repository and Kaggle (Table II). For each dataset, we randomly sample \( n = 3000 \) data points for training to run the optimization.

Based on the data distribution of each dataset, the hyperparameters are set as follows:

- \textbf{Covertype}: \( \sigma^{-2} = 5 \), \( \lambda = 2 \times 10^{-15} \), \( \mu = 1000 \), \( \alpha = 0.3 \) and \( \beta = 0.5 \).
Fig. 1. The plot of relative empirical risk (training error) vs. iteration shows that RFN enjoys a similar iteration complexity compared to the Newton method. In this sense, RFN outperforms L-BFGS, SSNCG, and GD.

Fig. 2. The plot of relative empirical risk (training error) vs. time cost shows that RFN outperforms nearly all other methods. Only L-BFGS has a similar performance to RFN.

**TABLE II**

| Dataset | # of points | # of features | Reference |
|---------|-------------|---------------|-----------|
| Covertype | 581012 | 54 | [42] |
| Cardio | 70000 | 11 | [43] |
| Rice | 18185 | 10 | [44] |

Cardio: $\sigma^{-2} = 100$, $\lambda = 2 \times 10^{-15}$, $\mu = 1000$, $\alpha = 0.3$ and $\beta = 0.5$.

Rice: $\sigma^{-2} = 50$, $\lambda = 2 \times 10^{-15}$, $\mu = 1000$, $\alpha = 0.3$ and $\beta = 0.5$.

For the number of random features for RFN ($m$) and the number of sub-sampled data points for SSNCG ($|I|$), we set the ratio as 10% of the training data (i.e., $m = |I| = 300$).

- **Performance**: We record the loss value and the time cost of each iteration for all methods. The time cost includes the time of finding the Newton direction (computing the Hessian and solving the linear system) and determining the step size. The run time is based on a desktop with a 6-core, AMD Ryzen 5 5600 G CPU and 15.5 G of RAM (3600 Mhz). The initial point $w_0$ is the all zero vector. For randomized methods RFN and SSNCG, we run the experiment 30 times and report the average both for the loss value and time cost.

Basing on Fig. 1, we see that in terms of iteration complexity, RFN and SSNCG outperform GD and L-BFGS, which use first-order information. From Fig. 2, we observe that RFN and L-BFGS have the most competitive loss vs. run time performance. The two figures together imply that RFN achieves an identical iteration complexity to the Newton method with a mild computational cost. In this simulation, for RFN we compute $D(w)$ and the gradient based on the approximated gram matrix, which leads to the saturation observed in Fig. 1.

- **Impact of $m$**: One factor affecting the convergence performance of RFN is the number of random features. We run another experiment on Cardio dataset with the same hyper-parameter set up except that we choose the ratio ($m/n$) from the set $\{10\%, 40\%, 70\%\}$. Figs. 3–4 verify that with more random features, we can achieve a smaller error, but this comes at the cost of increased run time. The reason is that using more
random features improves the Hessian approximation, but it also increases the time of computing the newton step based on \( (12) \).

- Impact of \( \mu \): Another factor affecting the convergence performance of RFN is the choice of \( \mu \). When \( \mu \) is very small, the approximated Hessian suffers from the singularity issue. On the other hand, when \( \mu \) is large, the composite kernel behaves approximately like \( \mu I \). In general, there is no monotonic relationship for the convergence performance in terms of \( \mu \). The main reason is that, as we can see in Theorem 3, \( \rho_d \) depends on multiple parameters that can vary with respect to \( \mu \). Here, we run an experiment on Covertype dataset with the same hyper-parameter setup except that \( \mu \) varies from \( 10^{-2} \) to \( 10^{5} \). In Fig. 5, we can observe that \( \mu = 10^{3} \) achieves the lowest final error. Of course, the optimal performance depends on the kernel choice and the dataset, and it is different across various problem settings.

- Performance on the testing dataset: To illustrate the practical performance of RFN on unseen data, we compare the test accuracy of the models learned by applying Newton method versus RFN. In this experiment, we perform a grid search over \( \sigma \) and \( \mu \), which are the main two hyper-parameters. Both \( \sigma \) and \( \mu \) can take values in the set \( \{1, 10, 10^2, \ldots, 10^5\} \), i.e., for each dataset, we cross validate over 36 possible combinations of hyper-parameters and select the one with the best accuracy on the validation dataset of 6000 points. Then, we report the test error for 6000 test points, using the models trained with Newton and RFN. In Table III, we observe that the prediction power of the models learned by RFN are similar (or quite competitive) compared to Newton method. While Table III presents only the best models, we observed in experimental results that for any given \( (\mu, \sigma) \) pair, the difference between the test accuracy of models learned by RFN and Newton is negligible.

### V. Conclusion

In this paper, we proposed a random-feature based Newton method (RFN) for risk minimization over RKHS. We drew explicit connections between the Hessian and the gram matrix and observed that sub-sampled Hessian suffers from the singularity issue. The theoretical point of view, we proved that the approximate Hessian is close to the original Hessian in terms of spectral norm when enough random features are sampled, which in turn ensures the local and global convergence with high probability. From the practical point of view, we applied RFN to three real-world datasets, compared it with other benchmarks, and showed that RFN enjoys a faster run time under certain conditions. Future directions include the development of distributed/decentralized variants of RFN as well as data-dependent sampling schemes for random features [45].

### Appendix

We make use of the following matrix concentration inequality [32] for the proof of Lemma 1, but we adopt the version used in [46] and present it for real matrices.

**Lemma 7 (Matrix Concentration Inequality):** Let \( B \) be a fixed \( d_1 \times d_2 \) matrix. Consider a \( d_1 \times d_2 \) random matrix \( R \) that satisfies

\[
\mathbb{E}[R] = B \quad \text{and} \quad ||R|| \leq U.
\]

Let \( M_1 \) and \( M_2 \) be semi-definite upper bounds such that

\[
\mathbb{E}[RR^\top] \preceq M_1 \quad \text{and} \quad \mathbb{E}[R^\top R] \preceq M_2.
\]

Define the quantities

\[
c \triangleq \max(\|M_1\|_1, \|M_2\|_1) \quad \text{and} \quad b \triangleq (\text{Tr}(M_1) + \text{Tr}(M_2))/c.
\]

Form the matrix sampling estimator

\[
R_h = \frac{1}{h} \sum_{i=1}^{h} R_i,
\]

where each \( R_i \) is an independent copy of \( R \). Then, for all \( \epsilon \geq \sqrt{c/h} + 2U/3h \),

\[
\Pr(\|R_h - B\| \geq \epsilon) \leq 4b \exp \left( -\frac{\epsilon^2/2}{c + 2U\epsilon/3} \right).
\]
A. Proof of Lemma 1

In the proof of this lemma, we disregard the dependence on \( w \) and denote \( \hat{\mathbf{H}}(w), \mathbf{H}(w) \) and \( \mathbf{D}(w) \) by \( \hat{\mathbf{H}}, \mathbf{H} \) and \( \mathbf{D} \), respectively. This is just for the sake of the presentation clarity.

First, we focus on \( \|\mathbf{Z}\mathbf{Z}^\top - \mathbf{K}_1\| \), which can be written as \( \frac{1}{m} \sum_{i=1}^{m} z_i z_i^\top - \mathbf{K}_1 \), where \( z_i \) is the \( i \)-th column of \( \mathbf{Z} \). Notice that \( \mathbb{E}[z_i z_i^\top] = \mathbf{K}_1 \). To apply Lemma 7, we observe that

\[
\|z_i z_i^\top\| \leq n \triangleq U,
\]

and

\[
\mathbb{E}[z_i z_i^\top z_i z_i^\top] \preceq n \mathbf{K}_1 \triangleq \mathbf{M}_1.
\]

By symmetry \( \mathbf{M}_1 = \mathbf{M}_2 \), and we also have that

\[
c = n \|\mathbf{K}_1\|
\]

\[
b = \frac{2 \text{Tr}(\mathbf{M}_1)}{c} = \frac{2 \text{Tr}(\mathbf{K}_1)}{\|\mathbf{K}_1\|}.
\]

With the above quantities, based on Lemma 7, we derive

\[
\Pr \left( \| \frac{1}{m} \sum_{i=1}^{m} z_i z_i^\top - \mathbf{K}_1 \| \geq \epsilon \right) \leq 4b \exp \left( \frac{-mc^2/2}{n \|\mathbf{K}_1\| + \frac{3}{4} ne} \right).
\]

To have the right hand side smaller than \( \delta \), we should solve the inequality

\[
\log \frac{4b}{\delta} \leq \frac{mc^2/2}{n \|\mathbf{K}_1\| + \frac{3}{4} ne},
\]

for \( m \), which implies

\[
m \geq \frac{2n \|\mathbf{K}_1\| + \frac{3}{4} ne}{c^2} \log \frac{4b}{\delta}.
\]

Since we further assume that \( \epsilon \leq \frac{3}{4} \|\mathbf{K}_1\| \), the sufficient lower bound on \( m \) can be simplified to

\[
m \geq \frac{4n \|\mathbf{K}_1\| + \frac{3}{8} ne}{c^2} \log \frac{4b}{\delta}.
\]

(29)

Therefore, by sampling \( m = \Omega \left( \frac{n \|\mathbf{K}_1\| \log \frac{n \|\mathbf{K}_1\|}{\epsilon \|\mathbf{K}_1\|} \right) \), with probability at least \( (1 - \delta) \), we have

\[
\|\mathbf{Z}\mathbf{Z}^\top - \mathbf{K}_1\| \leq \epsilon.
\]

(30)

Denoting \( (\mathbf{Z}\mathbf{Z}^\top + \mu \mathbf{I}) \) as \( \tilde{\mathbf{K}} \), based on (30) the approximation error of Hessian is upper-bounded as follows:

\[
\|\hat{\mathbf{H}} - \mathbf{H}\| = \left\| \frac{\tilde{\mathbf{K}} \mathbf{D} \mathbf{K}}{n} + \lambda \mathbf{K} - \frac{\mathbf{D} \mathbf{K}}{n} - \lambda \mathbf{K} \right\|
\]

\[
\leq \left\| \frac{\tilde{\mathbf{K}} \mathbf{D} \mathbf{K}}{n} - \frac{\mathbf{D} \mathbf{K}}{n} \right\| + \left\| \lambda (\mathbf{Z}\mathbf{Z}^\top - \mathbf{K}_1) \right\|
\]

\[
\leq \left\| \frac{\tilde{\mathbf{K}} \mathbf{D} \mathbf{K} - \mathbf{D} \mathbf{K}}{n} \right\| + \left\| \lambda (\mathbf{Z}\mathbf{Z}^\top - \mathbf{K}_1) \right\|
\]

\[
\leq \left( \lambda + \left\| \frac{\tilde{\mathbf{K}} \mathbf{D} \mathbf{K}}{n} \right\| + \left\| \frac{\mathbf{D} \mathbf{K}}{n} \right\| \right) \|\mathbf{Z}\mathbf{Z}^\top - \mathbf{K}_1\|
\]

\[
\leq \left( \lambda + \lambda_1 \left\| \frac{\tilde{\mathbf{K}} \mathbf{D} \mathbf{K}}{n} \right\| + \lambda_1 \left\| \frac{\mathbf{D} \mathbf{K}}{n} \right\| \right) \epsilon.
\]

(31)

Define \( \zeta \triangleq \lambda + \lambda_1 \left\| \frac{\tilde{\mathbf{K}} \mathbf{D} \mathbf{K}}{n} \right\| + \lambda_1 \left\| \frac{\mathbf{D} \mathbf{K}}{n} \right\| \). Based on (31), by replacing \( \epsilon \) with \( (\epsilon \gamma)/\zeta \) in (29), the result is proved.

B. Proof of Theorem 3

Denote the minimum eigenvalue of \( \hat{\mathbf{H}}(w_t) \) by \( \lambda_{\min}(\hat{\mathbf{H}}(w_t)) \), and recall that \( p_t \triangleq -[\hat{\mathbf{H}}(w_t)]^{-1} \nabla F(w_t) \). Since

\[
p_t^\top \hat{\mathbf{H}}(w_t) p_t \geq \lambda_{\min}(\hat{\mathbf{H}}(w_t)) \|p_t\|^2,
\]

by Corollary 2 we have that

\[
p_t^\top \hat{\mathbf{H}}(w_t) p_t \geq (1 - \epsilon) \gamma \|p_t\|^2.
\]

For any \( \alpha > 0 \), define \( w_\alpha \triangleq w_t + \alpha p_t \). With \( L \)-smoothness of the objective function \( F \), we have

\[
F(w_\alpha) - F(w_t) \leq (w_\alpha - w_t)^\top \nabla F(w_t) + \frac{L}{2} \|w_\alpha - w_t\|^2
\]

\[
= \alpha p_t^\top \nabla F(w_t) + \frac{\alpha^2 L}{2} \|p_t\|^2.
\]

To pass the Armijo line search condition, we need an \( \alpha \) which makes the following inequality hold

\[
\alpha p_t^\top \nabla F(w_t) + \frac{\alpha^2 L}{2} \|p_t\|^2 \leq \alpha \beta p_t^\top \nabla F(w_t).
\]

Since \( p_t^\top \nabla F(w_t) = -p_t^\top \hat{\mathbf{H}}(w_t) p_t \), the above inequality can be written as

\[
\alpha L \|p_t\|^2 \leq 2(1 - \beta) p_t^\top \hat{\mathbf{H}}(w_t) p_t.
\]

Therefore, if

\[
\alpha \leq 2(1 - \beta)(1 - \epsilon) \gamma / L,
\]

(given that \( p_t^\top \hat{\mathbf{H}}(w_t) p_t \geq (1 - \epsilon) \gamma \|p_t\|^2 \)), the Armijo line search is satisfied. This upper bound is iteration-independent. Now, \( w_{t+1} = w_t + \alpha_i p_t \), and based on (22), we have that

\[
F(w_{t+1}) \leq F(w_t) + \alpha_i \beta p_t^\top \nabla F(w_t)
\]

\[
= F(w_t) - \alpha_i \beta \nabla F(w_t)^\top [\hat{\mathbf{H}}(w_t)]^{-1} \nabla F(w_t)
\]

\[
\leq F(w_t) - \alpha_i \beta \|\nabla F(w_t)\|^2 / (1 + \epsilon) L,
\]

where the last inequality comes from (19). By subtracting \( F(w^*) \) from both sides and noting that \( F(w_t) - F(w^*) \leq \|\nabla F(w_t)\|^2 / 2 \gamma \) due to the strong convexity of \( F \), the result is proved.

Lemma 8 (Error Recursion): Let Assumption 2 hold. Assume that \( \alpha_t = 1 \) and \( \hat{\mathbf{H}}(w) \) is positive definite. We then have

\[
\|w_{t+1} - w^*\| \leq \nu \|w_t - w^*\| + \eta \|w_t - w^*\|^2,
\]

where \( \eta \triangleq \frac{M}{2 \lambda_{\min}(\hat{\mathbf{H}}(w_t))} \) and \( \nu \triangleq \frac{\|\hat{\mathbf{H}}(w_t) - \hat{\mathbf{H}}(w^*)\|}{\lambda_{\min}(\hat{\mathbf{H}}(w_t))} \).
C. Proof of Lemma 8

Define \( \Delta_t \triangleq w_t - w^* \). Since
\[
\nabla F(w_t) = \nabla F(w_{t+1}) = \nabla F(w_t) + (w_t - w_{t+1})^\top \hat{H}(w_t)(w_t - w_{t+1}).
\]

By setting \( w = w^* \) and noting that \( w_{t+1} - w_t = \Delta_{t+1} - \Delta_t \), we have
\[
\Delta_{t+1}^\top \hat{H}(w_t) \Delta_{t+1} = \Delta_{t+1}^\top \hat{H}(w_t) \Delta_t - \Delta_{t+1}^\top \nabla F(w_t) + \Delta_{t+1}^\top \nabla F(w^*),
\]
as \( \nabla F(w^*) = 0 \) due to the optimality of \( w^* \). We also have that
\[
\nabla F(w_t) - \nabla F(w^*) = \left( \int_0^t \nabla^2 F(w^* + \tau(w_t - w^*)) d\tau \right) (w_t - w^*).
\]

Therefore,
\[
\Delta_{t+1}^\top \hat{H}(w_t) \Delta_{t+1} = \Delta_{t+1}^\top \hat{H}(w_t) \Delta_t - \Delta_{t+1}^\top \nabla F(w_t) + \Delta_{t+1}^\top \nabla F(w^*)
\]
\[
\leq \| \Delta_{t+1} \| \| \hat{H}(w_t) - \nabla^2 F(w_t) \| \| \Delta_t \| + \| \Delta_t \| + \| \Delta_{t+1} \|
\]
\[
\leq \left( \int_0^t \| \nabla^2 F(w_t) - \nabla^2 F(w^* + \tau(w_t - w^*)) \| d\tau \right) \| \Delta_t \|
\]
\[
\leq \| \hat{H}(w_t) - \nabla^2 F(w_t) \| \| \Delta_t \| + \frac{M}{2} \| \Delta_t \|^2 \| \Delta_{t+1} \|.
\]

Since \( \Delta_{t+1}^\top \hat{H}(w_t) \Delta_{t+1} \geq \lambda_{\min}(\hat{H}(w_t)) \| \Delta_{t+1} \|^2 \) and \( \hat{H}(w_t) \) is positive definite, the result follows.

D. Proof of Lemma 4

In view of Lemma 1 and Corollary 2, we know that \( \| \hat{H}(w_t) - \nabla^2 F(w_t) \| \leq \epsilon \gamma \) and \( \lambda_{\min}(\hat{H}(w_t)) \geq (1 - \epsilon) \gamma \). The result follows by applying these to Lemma 8.

E. Proof of Theorem 5

According to Lemma 4, we have \( \| w_{t+1} - w^* \| \leq \nu \| w_t - w^* \| + \eta \| w_t - w^* \|^2 \) for every \( t \). The choice of \( w_0 \) guarantees
\[
\nu \| w_0 - w^* \| + \eta \| w_0 - w^* \|^2 \leq \rho \| w_0 - w^* \|, 
\]
and the proof follows by induction.

The probability that all iterations are successful is the complement of the probability that at least one iteration fails, which is bounded by \( t_0 \delta \). Therefore, the probability of a successful process is at least \( 1 - t_0 \delta \).

References

[1] L. Bottou, F. E. Curtis, and J. Nocedal, “Optimization methods for large-scale machine learning,” SIAM Rev., vol. 60, no. 2, pp. 223–311, 2018.
[2] Y. Nesterov, “Introductory lectures on convex programming volume I: Basic course,” Lecture Notes, vol. 3, no. 4, 1998, Art. no. 5.
[3] R. Bollapragada, R. H. Byrd, and J. Nocedal, “Exact and inexact subsampled Newton methods for optimization,” IMA J. Numer. Anal., vol. 39, no. 2, pp. 545–578, 2018.
[4] F. Roosta-Khorasani and M. W. Mahoney, “Sub-sampled Newton methods,” Math. Program., vol. 174, no. 1/2, pp. 293–326, 2019.
[5] P. Xu, J. Yang, F. Roosta-Khorasani, C. Ré, and M. W. Mahoney, “Subsampled Newton methods with non-uniform sampling,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2016, pp. 3000–3008.
[6] M. Pilanci and M. J. Wainwright, “Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence,” SIAM J. Optim., vol. 27, no. 1, pp. 205–245, 2017.
[7] C. G. Broyden, “The convergence of a class of double-rank minimization algorithms: 2 The new algorithm,” IMA J. Appl. Math., vol. 6, no. 3, pp. 222–231, 1970.
[8] R. Fletcher, “A new approach to variable metric algorithms,” Comput. J., vol. 13, no. 3, pp. 317–322, 1970.
[9] D. Goldfarb, “A family of variable-memory methods derived by variational means,” Math. Comput., vol. 24, no. 109, pp. 23–26, 1970.
[10] D. F. Shanno, “Conditioning of quasi-Newton methods for function minimization,” Math. Computation, vol. 24, no. 111, pp. 647–656, 1970.
[11] N. N. Schraudolph, J. Yu, and S. Günter, “A stochastic quasi-Newton method for online convex optimization,” in Proc. 11th Int. Conf. Artif. Intell. Statist., 2007, pp. 436–443.
[12] A. Mokhtari and A. Ribeiro, “RES: Regularized stochastic BFGS algorithm,” IEEE Trans. Signal Process., vol. 62, no. 23, pp. 6089–6104, Dec. 2014.
[13] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer, “A stochastic quasi-Newton method for large-scale optimization,” SIAM J. Optim., vol. 26, no. 2, pp. 1008–1031, 2016.
[14] P. Moritz, R. Nishihara, and M. Jordan, “A linearly-convergent stochastic L-BFGS algorithm,” in Proc. 19th Int. Conf. Artif. Intell. Statist., 2016, pp. 249–258.
[15] A. Bordes, L. Bottou, and P. Gallinari, “SGD-QN: Careful quasi-Newton stochastic gradient descent,” J. Mach. Learn. Res., vol. 10, no. 59 pp. 1737–1754, 2009.
[16] J. Söld-Dickstein, B. Boole, and S. Ganguli, “Fast large-scale optimization by unifying stochastic gradient and quasi-Newton methods,” in Proc. Int. Conf. Mach. Learn., 2014, pp. 604–612.
[17] A. Mokhtari and A. Ribeiro, “Global convergence of online limited memory BFGS,” J. Mach. Learn. Res., vol. 16, no. 1, pp. 3151–3181, 2015.
[18] A. Mokhtari, M. Eisen, and A. Ribeiro, “IQN: An incremental quasi-Newton method with local superlinear convergence rate,” SIAM J. Optim., vol. 28, no. 2, pp. 1670–1698, 2018.
[19] R. Gower, D. Goldfarb, and P. Richtárik, “Stochastic block BFGS: Squeezing more curvature out of data,” in Proc. Int. Conf. Mach. Learn., 2016, pp. 1869–1878.
[20] R. Zhao, W. B. Haskell, and Y. Y. F. Tan, “Stochastic L-BFGS: Improved convergence rates and practical acceleration strategies,” IEEE Trans. Signal Process., vol. 66, no. 5, pp. 1155–1169, Mar. 2018.
[21] D. Chang, S. Sun, and C. Zhang, “An accelerated linearly convergent stochastic L-BFGS algorithm,” IEEE Trans. Neural Netw. Learn. Syst., vol. 30, no. 11, pp. 3338–3346, Nov. 2019.
[22] Q. Zhang, F. Huang, C. Deng, and H. Huang, “Faster stochastic quasi-Newton methods,” IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 9, pp. 4388–4397, Sep. 2022.
[23] J. Shawe-Taylor et al., Kernel Methods for Pattern Analysis. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[24] F. Pérez-Cruz and O. Bousquet, “Kernel methods and their potential use in signal processing,” IEEE Signal Process. Mag., vol. 21, no. 3, pp. 57–65, May 2004.

[25] A. Rahimi and B. Recht, “Random features for large-scale kernel machines,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2008, pp. 1177–1184.

[26] R. H. Byrd, G. M. Chin, W. Neveitt, and J. Nocedal, “On the use of stochastic Hessian information in optimization methods for machine learning,” SIAM J. Optim., vol. 21, no. 3, pp. 977–995, 2011.

[27] R. H. Byrd, G. M. Chin, J. Nocedal, and Y. Wu, “Sample size selection in optimization methods for machine learning,” Math. Program., vol. 134, no. 1, pp. 127–155, 2012.

[28] C.-C. Wang, C.-H. Huang, and C.-J. Lin, “Subsampled Hessian Newton methods for supervised learning,” Neural Comput., vol. 27, no. 8, pp. 1766–1795, 2015.

[29] J. Martens, “Deep learning via Hessian-free optimization,” in Proc. Int. Conf. Mach. Learn., 2010, pp. 735–742.

[30] M. A. Erdogdu and A. Montanari, “Convergence rates of sub-sampled Newton methods,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2015, pp. 3052–3060.

[31] J. A. Tropp and S. J. Wright, “Computational methods for sparse solution of linear inverse problems,” Proc. IEEE, vol. 98, no. 6, pp. 948–958, Jun. 2010.

[32] J. A. Tropp et al., “An introduction to matrix concentration inequalities,” Found. Trends Mach. Learn., vol. 8, no. 1/2, pp. 1–230, 2015.

[33] N. Agarwal, B. Bullins, and E. Hazan, “Second-order stochastic optimization for machine learning in linear time,” J. Mach. Learn. Res., vol. 18, no. 1, pp. 4148–4187, 2017.

[34] M. Mutny, “Stochastic second-order optimization via von Neumann series,” 2016, arXiv:1612.04694.

[35] J. Kivinen, A. J. Smola, and R. C. Williamson, “Online learning with kernels,” IEEE Trans. Signal Process., vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

[36] A. Koppel, G. Warnell, E. Stump, and A. Ribeiro, “Parsimonious online learning with kernels via sparse projections in function space,” J. Mach. Learn. Res., vol. 20, no. 1, pp. 83–126, 2019.

[37] B. Dai et al., “Scalable kernel methods via doubly stochastic gradients,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2014, pp. 3041–3049.

[38] T.-J. Chang and S. Shahrampour, “Global convergence of Newton method for empirical risk minimization in reproducing kernel hilbert space,” in Proc. Asilomar Conf. Signals Syst. Comput., 2020, pp. 1222–1226.

[39] P. Kar and H. Karnick, “Random feature maps for dot product kernels,” in Proc. Int. Conf. Artif. Intell. Statist., 2012, pp. 583–591.

[40] J. Yang, V. Sindhwani, Q. Fan, H. Avron, and M. W. Mahoney, “Random laplace feature maps for semigroup kernels on histograms,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2014, pp. 971–978.

[41] A. Gittens and M. W. Mahoney, “Revisiting the Nyström method for improved large-scale machine learning,” J. Mach. Learn. Res., vol. 17, no. 1, pp. 3977–4041, 2016.

[42] J. A. Blackard and D. J. Dean, “Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover using cartographic variables,” Comput. Electron. Agriculture, vol. 24, no. 3, pp. 131–151, 1999.

[43] “Cardiovascular disease data,” data retrieved from Kaggle. [Online]. Available: https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset

[44] “Rice type classification,” data retrieved from Kaggle. [Online]. Available: https://www.kaggle.com/datasets/mssmartypants/rice-type-classification

[45] Y. Wang and S. Shahrampour, “ORCCA: Optimal randomized canonical correlation analysis,” IEEE Trans. Neural Netw. Learn. Syst., early access, Nov. 15, 2021, doi: 10.1109/TNNLS.2021.3124868.

[46] H. Avron, M. Kapralov, C. Musco, C. Musco, A. Velingker, and A. Zandieh, “Random fourier features for kernel ridge regression: Approximation bounds and statistical guarantees,” in Proc. Int. Conf. Mach. Learn., 2017, pp. 253–262.

Ting-Jui Chang received the B.S. degree in electrical and computer engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2016, and the M.S. degree in computer engineering from Texas A&M University, College Station, TX, USA, in 2018. He is currently working toward the Ph.D. degree in industrial engineering with Northeastern University, Boston, MA, USA. His research interests include distributed learning and optimization, and decentralized and online control.

Shahin Shahrampour (Senior Member, IEEE) received the Ph.D. degree in electrical and systems engineering, the M.A. degree in statistics (The Wharton School), and the M.S.E. degree in electrical engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 2015, 2014, and 2012, respectively. He is currently an Assistant Professor with the Department of Mechanical and Industrial Engineering, Northeastern University, Boston, MA, USA. His research interests include machine learning, optimization, sequential decision-making, and distributed learning, with a focus on developing computationally efficient methods for data analytics.