Decentralised Learning with Distributed Gradient Descent and Random Features

Dominic Richards [1] **Patrick Rebeschini** [1] **Lorenzo Rosasco** [2 3 4]

Abstract

We investigate the generalisation performance of Distributed Gradient Descent with Implicit Regularisation and Random Features in the homogenous setting where a network of agents are given data sampled independently from the same unknown distribution. Along with reducing the memory footprint, Random Features are particularly convenient in this setting as they provide a common parameterisation across agents that allows to overcome previous difficulties in implementing Decentralised Kernel Regression. Under standard source and capacity assumptions, we establish high probability bounds on the predictive performance for each agent as a function of the step size, number of iterations, inverse spectral gap of the communication matrix and number of Random Features. By tuning these parameters, we obtain statistical rates that are minimax optimal with respect to the total number of samples in the network. The algorithm provides a linear improvement over single machine Gradient Descent in memory cost and, when agents hold enough data with respect to the network size and inverse spectral gap, a linear speed-up in computational runtime for any network topology. We present simulations that show how the number of Random Features, iterations and samples impact predictive performance.

1. Introduction

In supervised learning, an agent is given a collection of training data to fit a model that can predict the outcome of new data points. Due to the growing size of modern data sets and complexity of many machine learning models, a popular approach is to incrementally improve the model with respect to a loss function that measures the performance on the training data. The complexity and stability of the resulting model is then controlled implicitly by algorithmic parameters, such as, in the case of Gradient Descent, the step size and number of iterations. An appealing collection of models in this case are those associated to the Reproducing Kernel Hilbert Space (RKHS) for some positive definite kernel, as the resulting optimisation problem (originally over the space of functions) admits a tractable form through the Kernel Trick and Representer Theorem, see for instance (Schölkopf et al., 2001).

Given the growing size of data, privacy concerns as well as the manner in which data is collected, distributed computation has become a requirement in many machine learning applications. Here training data is split across a number of agents which alternate between communicating model parameters to one another and performing computations on their local data. In centralised approaches (effective star topology), a single agent is typically responsible for collecting, processing and disseminating information to the agents. Meanwhile for many applications, including ad-hoc wireless and peer-to-peer networks, such centralised approaches are unfeasible. This motivates decentralised approaches where agents in a network only communicate locally within the network i.e. to neighbours at each iteration.

Many problems in decentralised multi-agent optimisation can be phrased as a form of consensus optimisation (Tsitsiklis et al., 1986; Tsitsiklis, 1984; Johansson et al., 2007; Nedic & Ozdaglar, 2009; Nedić et al., 2009; Johansson et al., 2009; Lobel & Ozdaglar, 2011; Matei & Baras, 2011; Boyd et al., 2011; Duchi et al., 2012; Shi et al., 2015; Mokhtari & Ribeiro, 2016). In this setting, a network of agents wish to minimise the average of functions held by individual agents, hence "reaching consensus" on the solution of the global problem. A standard approach is to augment the original optimisation problem to facilitate a decentralised algorithm. This typically introduces additional penalisation (or constraints) on the difference between neighbouring agents within the network, and yields a higher dimensional optimisation problem which decouples across the agents.

[1]Department of Statistics, University of Oxford, 24-29 St Giles', Oxford, OX1 3LB [2]MaLGa Center, Universitá degli Studi di Genova, Genova, Italy [3]Istituto Italiano di Tecnologia, Via Morego, 30, Genoa 16163, Italy [4]Massachusetts Institute of Technology, Cambridge, MA 02139, USA. Correspondence to: Patrick Rebeschini <patrick.rebeschini@stats.ox.ac.uk>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

This augmented problem can then often be solved using standard techniques whose updates can now be performed in a decentralised manner. While this approach is flexible and can be applied to many consensus optimisation problems, it often requires more complex algorithms which depend upon the tuning of additional hyper parameters, see for instance the Alternating Direction Method of Multiplers (ADMM) (Boyd et al., 2011).

Many distributed machine learning problems, in particular those involving empirical risk minimisation, can been framed in the context of consensus optimisation. As discussed in (Bouboulis et al., 2017; Koppel et al., 2018), for the case of Decentralised Kernel Regression it is not immediately clear how the objective ought to be augmented to facilitate both a decentralised algorithm and the Representer Theorem. Specifically, so the problem decouples across the network and agents have a common represention of the estimated function. Indeed, while distributed kernel regression can be performed in the one-shot Divide and Conquer setting (Star Topology) (Zhang et al., 2015; Lin et al., 2017; Guo et al., 2017; Mücke & Blanchard, 2018; Dobriban & Sheng, 2020) where there is a fusion center to combine the resulting estimators computed by each agent, in the decentralised setting there is no fusion center and agents must communicate for multiple rounds. A number of works have aimed to tackle this challenge (Forero et al., 2010; Mitra & Bhatia, 2014; Gao et al., 2015; Chouvardas & Draief, 2016; Bouboulis et al., 2017; Koppel et al., 2018), although these methods often include approximations whose impact on statistical performance is not clear[1]. Most relevant to our work is (Bouboulis et al., 2017) where Distributed Gradient Descent with Random Fourier Features is investigated in the online setting. In this case regret bounds are proven, but it is not clear how the number of Random Fourier Features or network topology impacts predictive performance in conjunction with non-parametric statistical assumptions[2]. For more details on the challenges of the developing a Decentralised Kernel Regression algorithm see Section 2.1.

1.1. Contributions

In this work we give statistical guarantees for a simple and practical Decentralised Kernel Regression algorithm. Specifically, we study the learning performance (Generalisation Error) of full-batch Distributed Gradient Descent (Nedic & Ozdaglar, 2009) with implicit regularisation (Richards & Patrick, 2020; Richards & Rebeschini, 2019) and Random Features (Rahimi & Recht, 2008; Rudi & Rosasco, 2017). Random Features can be viewed as a form of non-linear sketching or shallow neural networks with random initialisations, and have be utilised to facilitate the large scale application of kernel methods by overcoming the memory bottle-neck. In our case, they both decrease the memory cost and yield a simple Decentralised Kernel Regression algorithm. While previous approaches have viewed Decentralised Kernel Regression with explicit regularisation as an instance of consensus optimisation, where the speed-up in runtime depends on the network topology (Duchi et al., 2012; Scaman et al., 2017). We build upon (Richards & Rebeschini, 2019) and directly study the Generalisation Error of Distributed Gradient Descent with implicit regularisation. This allows linear speed-ups in runtime for *any* network topology to be achieved by leveraging the statistical concentration of quantities held by agents. Specifically, our analysis demonstrates how the number of Random Features, network topology, step size and number of iterations impact Generalisation Error, and thus, can be tuned to achieve minimax optimal statistical rates with respect to all of the samples within the network (Caponnetto & De Vito, 2007). When agents have sufficiently many samples with respect to the network size and topology, and the number of Random Features equal the number required by single machine Gradient Descent, a linear speed-up in runtime and linear decrease memory useage is achieved over single machine Gradient Descent. Previous guarantees given in consensus optimisation require the number of iterations to scale with the inverse spectral gap of the network (Duchi et al., 2012; Scaman et al., 2017), and thus, a linear speed-up in runtime is limited to well connected topologies. We now provide a summary of our contributions.

- **Decentralised Kernel Regression Algorithm**: By leveraging Random Features we develop a simple, practical and theoretically justified algorithm for Decentralised Kernel Regression. It achieves a linear reduction in memory cost and, given sufficiently many samples, a linear speed-up in runtime for any graph topology (Theorem 1, 2). This required extending the theory of Random Features to the decentralised setting (Section 4).
- **Refined Statistical Assumptions**: Considering the attainable case in which the minimum error over the hypothesis class is achieved, we give guarantees that hold over a wider range of complexity and capacity assumptions. This is achieved through a refined analysis of the Residual Network Error term (Section 4.4).
- **Bounds in High Probability**: All guarantees hold in high probability, where previous results (Richards & Rebeschini, 2019) for the decentralised setting only held in expectation. This is achieved through refined analysis of the Population Network Error (Section 4.3).

[1] Additional details on some of these works have been included within Remark 2 in the Appendix

[2] We note the concurrent work (Xu et al., 2020) which also investigates Random Fourier Features for decentralised non-parametric learning. The differences from our work have been highlighted in Remark 3 in the Appendix.

This work is structured as follows. Section 2 introduces the notation and Random Features. Section 3 presents the main theoretical results. Section 4 provides the error decomposition and a sketch proof of the refined analysis. Section 5 presents simulation results. Section 6 gives the conclusion.

2. Setup

This section introduces the setting. Section 2.1 introduces Decentralised Kernel Regression and the challenges in developing a decentralised algorithm. Section 2.2 introduces the link between Random Features and kernel methods. Section 2.3 introduces Distributed Gradient Descent with Random Features.

2.1. Challenges of Decentralised Kernel Regression

We begin with the single machine case then go on to the decentralised case.

Single Machine Consider a standard supervised learning problem with squared loss. Given a probability distribution ρ over $X \times \mathbb{R}$, we wish to solve

$$\min_f \mathcal{E}(f), \quad \mathcal{E}(f) = \int (f(x) - y)^2 d\rho(x, y), \tag{1}$$

given a collection of independently and identically distributed (i.i.d.) samples drawn from ρ, here denoted $(x_i, y_i)_{i=1}^m \in (X \times \mathbb{R}^m)$. Kernel methods are non-parametric approaches defined by a kernel $k : X \times X \to \mathbb{R}$ which is symmetric and positive definite. The space of functions considered will be the Reproducing Kernel Hilbert Space associated to the kernel k, that is, the function space $\mathcal{H}$ defined as the completion of the linear span $\{K(x, \cdot) : x \in X\}$ with respect to the inner product $\langle K(x, \cdot), K(x', \cdot)\rangle_{\mathcal{H}} := K(x, x')$ (Aronszajn, 1950). When considering functions that minimise the empirical loss with explicit regularisation $\lambda \geq 0$

$$\min_{f \in \mathcal{H}} \left\{ \frac{1}{m} \sum_{i=1}^m (f(x_i) - y_i)^2 + \lambda \|f\|_{\mathcal{H}}^2 \right\} \tag{2}$$

we can appeal to the Representer Theorem (Schölkopf et al., 2001), and consider functions represented in terms of the data points, namely $\widehat{f}(x) = \sum_{i=1}^m \alpha_i k(x_i, x)$ where $\alpha = (\alpha_1, \ldots, \alpha_m) \in \mathbb{R}^m$ are a collection of weights. The weights are then often written in terms of the gram-matrix $K \in \mathbb{R}^{m \times m}$ whose i, jth entry is $K_{ij} = k(x_i, x_j)$.

Decentralised Consider a connected network of n agents $G = (V, E)$ $|V| = n$, joined by edges $E \subseteq V \times V$, that wish to solve (1). Each agent $v \in V$ has a collection of m i.i.d. training points $(x_{i,v}, y_{i,v})_{i=1}^m \in (X \times \mathbb{R})^m$ sampled from ρ. Following standard approaches in consensus optimisation we arrive at the optimisation problem

$$\min_{f_v \in \mathcal{H}, v \in V} \left\{ \frac{1}{nm} \sum_{v \in V} \sum_{i=1}^m (f_v(x_{i,v}) - y_{i,v})^2 + \lambda \|f_v\|_{\mathcal{H}}^2 \right\}$$
$$f_v = f_w \quad (v, w) \in E,$$

where a local function for each agent f_v is only evaluated at the data held by that agent $(x_{i,v}, y_{i,v})_{i=1}^m$, and a constraint ensures agents that share an edge are equal. This constrained problem is then often solved by considering the dual problem (Scaman et al., 2017) or introducing penalisation (Jakovetić et al., 2015). In either case, the objective decouples so that given $\{f_v\}_{v \in V}$ it can be evaluated and optimised in a decentralised manner. As discussed by (Bouboulis et al., 2017; Koppel et al., 2018), it is not immediately clear whether a representation for $\{f_v\}_{v \in V}$ exists in this case that respects the gram-matrices held by each agent. Recall, in the decentralised setting, only agent v can access the data $(x_{i,v}, y_{i,v})_{i=1}^m$ and the kernel evaluated at their data points $k(x_{i,v}, x_{j,v})$ for $i, j = 1, \ldots, m$.

2.2. Feature Maps and Kernel Methods

Consider functions parameterised by $\omega \in \mathbb{R}^M$ and written in the following form

$$f(x) = \langle \omega, \phi_M(x) \rangle, \quad \forall x \in X,$$

where $\phi_M : X \to \mathbb{R}^M, M \in \mathbb{N}$, denotes a family of finite dimensional feature maps that are identical and known across all of the agents. Feature maps in our case take a data point x to a (often higher dimensional) space where Euclidean inner products approximate the kernel. That is, informally, $k(x, x') \approx \langle \phi_M(x), \phi_M(x') \rangle$. One now classical example is Random Fourier Features (Rahimi & Recht, 2008) which approximate the Gaussian Kernel.

Random Fourier Features If $k(x, x') = G(x - x')$, where $G(z) = e^{-\frac{1}{2\sigma^2}\|z\|^2}$, for $\sigma > 0$ then we have

$$G(x - x') = \frac{1}{2\pi Z} \int \int_0^{2\pi}$$
$$\sqrt{2}\cos(\omega^\top x + b)\sqrt{2}\cos(\omega^\top x' + b) e^{-\frac{\sigma^2}{2}\|\omega\|^2} d\omega db$$

where Z is a normalizing factor. Then, for the Gaussian kernel, $\phi_M(x) = M^{-1/2}(\sqrt{2}\cos(\omega_1^\top x + b_1), \ldots, \sqrt{2}\cos(\omega_M^\top x + b_M))$, where $\omega_1, \ldots, \omega_M$ and $b_1, \ldots, b_M$ sampled independently from $\frac{1}{Z} e^{-\sigma^2\|\omega\|^2/2}$ and uniformly in $[0, 2\pi]$, respectively.

More generally, this motivates the strategy in which we assume the kernel k can be expressed as

$$k(x, x') = \int \psi(x, \omega)\psi(x', \omega) d\pi(\omega), \quad \forall x, x' \in X, \tag{3}$$

where (Ω, π) is a probability space and $\psi : X \times \Omega \to \mathbb{R}$ (Reed, 2012). Random Features can then be seen as Monte Carlo approximations of the above integral.

2.3. Distributed Gradient Descent and Random Features

Since the functions are now linearly parameterised by $\omega \in \mathbb{R}^M$, agents can consider the simple primal method Distributed Gradient Descent (Nedic & Ozdaglar, 2009). Initialised at $\widehat{\omega}_{1,v} = 0$; for $v \in V$, agents update their iterates for $t \geq 1$

$$\widehat{\omega}_{t+1,v} = \sum_{w \in V} P_{vw} \times \Big(\widehat{\omega}_{t,w} - \frac{\eta}{m} \sum_{i=1}^{m} \big(\langle \widehat{\omega}_{t,w}, \phi_M(x_{i,w}) \rangle - y_{i,w} \big) \phi_M(x_{i,w}) \Big), \tag{4}$$

where $P \in \mathbb{R}^{n \times n}$ is a doubly stochastic matrix supported on the network i.e. $P_{ij} \neq 0$ only if $(i,j) \in E$, and η is a fixed stepsize. The above iterates are a combination of two steps. Each agent performing a local Gradient Descent step with respect to their own data i.e. $\widehat{\omega}_{t,w} - \frac{\eta}{m} \sum_{i=1}^{m} \big(\langle \widehat{\omega}_{t,w}, \phi_M(x_{i,w}) \rangle - y_{i,w} \big) \phi_M(x_{i,w})$ for agent $w \in V$. And a communication step where agents average with their neighbours as encoded by the summation $\sum_{w \in V} P_{vw} a_w$, where a_w is the quantity held by agent $w \in V$. The performance of Distributed Gradient Descent naturally depends on the connectivity of the network. In our case it is encoded by the second largest eigenvalue of P in absolute value, denoted $\sigma_2 \in [0, 1)$. In particular, it arises through the inverse spectral gap $1/(1-\sigma_2)$, which is known to scale with the network size for particular topologies, that is $O\big(1/(1-\sigma_2)\big) = O(n^\beta)$ where $\beta = 2$ for a cycle, $\beta = 1$ for a grid and $\beta = 0$ for an expander, see for instance (Duchi et al., 2012). Naturally, more "connected" topologies have larger spectral gaps, and thus, smaller inverses.

Notation For $a, b \in \mathbb{R}$ we denote $a \vee b$ as the maximum between a and b and $a \wedge b$ the minimum. We say $a \simeq b$ if there exists a constant c independent of $n, m, M, (1-\sigma_2)^{-1}$ up-to logarithmic factors such that $a = cb$. Similarly we write $a \lesssim b$ if $a \leq bc$ and $a \gtrsim b$ if $a \geq cb$.

3. Main Results

This section presents the main results of this work. Section 3.1 provides the results under basic assumptions. Section 3.2 provides the results under more refined assumptions.

3.1. Basic Result

We begin by introducing the following assumption related to the feature map.

Assumption 1 *Let (Ω, π) be a probability space and define the feature map $\psi : X \times \Omega \to \mathbb{R}$ for all $x \in X$ such that (3) holds. Define the family of feature maps for $M > 0$*

$$\phi_M(x) := \frac{1}{\sqrt{M}} (\psi(x, \omega_1), \dots, \psi(x, \omega_M))$$

where $(\omega_j)_{j=1}^M \in \Omega$ are sampled independently from π.

The above assumption states that the feature map is made of M independent features $\psi(x, \omega_i)$ for $i = 1, \dots, M$. This is satisfied for a wide range of kernels, see for instance Appendix E of (Rudi & Rosasco, 2017). The next assumption introduces some regularity to the feature maps.

Assumption 2 *The function ψ is continuous and there exists $\kappa \geq 1$ such that $|\psi(x, \omega)| \leq \kappa$ for any $x \in X, \omega \in \Omega$.*

This implies that the kernel considered is bounded $|k(x, x')| \leq \kappa^2$ which is a common assumption in statistical learning theory (Cucker & Zhou, 2007; Steinwart & Christmann, 2008). The following assumption is related to the optimal predictor.

Assumption 3 *Let $\mathcal{H}$ be the RKHS with kernel k. Suppose there exists $f_{\mathcal{H}} \in \mathcal{H}$ such that $\mathcal{E}(f_{\mathcal{H}}) = \inf_{f \in \mathcal{H}} \mathcal{E}(f)$.*

It states that the optimal predictor is within the interior of $\mathcal{H}$. Moving beyond this assumption requires considering the non-attainable case, see for instance (Dieuleveut et al., 2016), which is left to future work. Finally, the following assumption is on the response moments.

Assumption 4 *For any $x \in X$*

$$\int y^{2\ell} d\rho(y|x) \leq \ell! B^\ell p, \quad \forall \ell \in \mathbb{N}$$

for constants $B \in (0, \infty)$ and $p \in (1, \infty)$, ρ_X-almost surely.

This assumption is satisfied if the response is bounded or generated from a model with independent zero mean Gaussian noise.

Given an estimator $\widehat{f}$, its excess risk is defined as $\mathcal{E}(\widehat{f}) - \mathcal{E}(f_{\mathcal{H}})$. Let the estimator held by agent $v \in V$ be denoted by $\widehat{f}_{t,v} = \langle \widehat{\omega}_{t,v}, \phi_M(\cdot) \rangle$, where $\widehat{\omega}_{t,v}$ is the output of Distributed Gradient Descent (4) for agent v. Given this basic setup, we state the prediction bound prescribed by our theory.

Theorem 1 (Basic Case) *Let $n, m, M \in \mathbb{N}_+$, $\delta \in (0, 1)$, $t \geq 4$, $\eta\kappa^2 \leq 1$ and $\eta \simeq 1$. Under assumptions 1 to 4, the following holds with high probability for any $v \in V$*

$$\mathcal{E}(\widehat{f}_{t+1,v}) - \mathcal{E}(f_{\mathcal{H}}) \lesssim \frac{1}{\sqrt{nm}}$$

when

$$m \gtrsim \frac{n^3}{(1-\sigma_2)^4}, \quad M \simeq \sqrt{nm}, \quad \text{and } t = \sqrt{nm}. \tag{5}$$

Theorem 1 demonstrates that Distributed Gradient Descent with Random Features achieves optimal statistical rates, in the minimax sense (Caponnetto & De Vito, 2007; Blanchard & Mücke, 2018), with respect to all nm samples when three conditions are met. The first $m \gtrsim n^3/(1-\sigma_2)^4$ ensures that the network errors, due to agents communicating locally on the network, are sufficiently small from the phenomena of concentration. The second $M \simeq \sqrt{nm}$ ensures that the agents have sufficiently many Random Features to control the kernel approximation. It aligns with the number required by single machine Gradient Descent with all nm samples (Carratino et al., 2018). Finally $t = \sqrt{nm}$ is the number of iterations required to trade off the bias and variance error terms. This is the number of iterations required by single machine Gradient Descent with all nm samples, and thus, due to considering a distributed algorithm, gives a linear speed-up in runtime. We now discuss the runtime and space complexity of Distributed Gradient Descent with Random Features when the covariates take values in $\mathbb{R}^D$ for some $D > 0$. Remark 1 in Appendix A shows how, with linear features, Random Features can yield communication savings when $D > M$.

Pre-processing + Space Complexity After a pre-processing step which costs $O(DMm) = O(Dm^{3/2}\sqrt{n})$, Distributed Gradient Descent has each agent store a $m \times M = m \times \sqrt{nm}$ matrix. Single machine Gradient Descent performs a $O(DMnm) = O(D(nm)^{3/2})$ pre-processing step and stores a $nm \times M = nm \times \sqrt{nm}$ matrix. Distributed Gradient Descent thus gives a linear order n improvement in pre-processing time and memory cost.

Time Complexity Suppose one gradient computation costs 1 unit of time and communicating with neighbours costs τ. Given sufficiently many samples $m \gtrsim n^3/(1-\sigma_2)^4$ then *Single Machine Iterations = Distributed Iterations* and the speed-up in runtime for Distributed Gradient Descent over single machine Gradient Descent is

$$\begin{aligned}\text{Speed-up} &:= \frac{\text{Single Machine Runtime}}{\text{Distributed Runtime}} \\ &= \frac{\text{Single Machine Iteration Time}}{\text{Distributed Iteration Time}} \underbrace{\frac{\text{Single Machine Iters.}}{\text{Distributed Iters.}}}_{=1} \\ &= \frac{nm}{m + \tau + M\text{Deg}(P)} \simeq n\end{aligned}$$

where the final equality holds when the communication delay and cost of aggregating the neighbours solutions is bounded $\tau + M\text{Deg}(P) \lesssim m$. This observation demonstrates a linear speed-up in runtime can be achieved for *any* network topology. This is in contrast to results in decentralised consensus optimisation where the speed-up in runtime usually depends on the network topology, with a linear improvement only occurring for well connected topologies i.e. expander and complete, see for instance (Duchi et al., 2012; Scaman et al., 2017).

3.2. Refined Result

Let us introduce two standard statistical assumptions related to the underlying learning problem. With the marginal distribution on covariates $\rho_X(x) := \int_{\mathbb{R}} \rho(x,y)dy$ and the space of square integrable functions $L^2(X, \rho_X) = \{f : X \to \mathbb{R} : \|f\|_\rho^2 = \int |f|^2 d\rho_X < \infty\}$, let $L : L^2(X,\rho_X) \to L^2(X,\rho_X)$ be the integral operator defined for $x \in X$ as $Lf(x) = \int k(x,x')f(x')d\rho_X(x')$, $\forall f \in L^2(X,\rho_X)$. The above operator is symmetric and positive definite. The assumptions are then as follows.

Assumption 5 *For any $\lambda > 0$, define the effective dimension as $\mathcal{N}(\lambda) := \text{Tr}\left((L+\lambda I)^{-1}L\right)$, and assume there exists $Q > 0$ and $\gamma \in [0,1]$ such that $\mathcal{N}(\lambda) \leq Q^2\lambda^{-\gamma}$. Moreover, assume there exists $1 \geq r \geq 1/2$ and $g \in L^2(X,\rho_X)$ such that $f_{\mathcal{H}}(x) = (L^r g)(x)$.*

The above assumptions will allow more refined bounds on the Generalisation Error to be given. The quantity $\mathcal{N}(\lambda)$ is the effective dimension of the hypothesis space, and Assumption 5 holds for $\gamma > 0$ when the ith eigenvalue of L is of the order $i^{-1/\gamma}$, for instance. Meanwhile, the second condition for $1 \geq r \geq 1/2$ determines which subspace the optimal predictor is in. Here larger r indicates a smaller sub-space and a stronger condition. The refined result is then as follows.

Theorem 2 (Refined) *Let $n, m, M \in \mathbb{N}_+$, $\delta \in (0,1)$, $t \geq 2t^\star \geq 4$, $\eta\kappa^2 \leq 1$ and $\eta \simeq 1$. Under assumptions 1 to 5 with $r + \gamma > 1$, the following holds with high probability for any $v \in V$*

$$\mathcal{E}(\widehat{\omega}_{t+1,v}) - \mathcal{E}(f_{\mathcal{H}}) \lesssim (nm)^{\frac{-2r}{2r+\gamma}}$$

when we let $t^\star \simeq 1/(1-\sigma_2)$ and have

$$\underbrace{m \gtrsim \left((t^\star)^{\frac{(1+\gamma)(2r+\gamma)}{2(r+\gamma-1)}} n^{\frac{r+1}{r+\gamma-1}}\right) \vee \left((t^\star)^{2\vee(2r+\gamma)} n^{\frac{2r}{\gamma}}\right)}_{\textit{Sufficiently Many Samples}}$$

$$\underbrace{M \simeq (nm)^{\frac{1+\gamma(2r-1)}{2r+\gamma}}}_{\textit{Single Machine Random Features}} \qquad \underbrace{t = (nm)^{\frac{1}{2r+\gamma}}}_{\textit{Single Machine Iterations}}$$

Once again, the statistical rate achieved $(nm)^{-\frac{2r}{2r+\gamma}}$ is the minimax optimal rate with respect to all of the samples within the network (Caponnetto & De Vito, 2007), and both the number of Random Features as well as the number of iterations match the number required by single machine Gradient Descent when given *sufficiently many samples* m. When $r = 1/2$ and $\gamma = 1$ we recover the basic result given in Theorem 1, with the bounds now adapting to complexity

of the predictor as well as capacity through r and γ, respectively. In the low dimensional setting when $\gamma = 0$, we note our guarantees do not offer computational speed-ups over single machine Gradient Descent. While counter-intuitive, this observation aligns with (Richards & Rebeschini, 2019), which found the easier the problem (larger r, smaller γ) the more samples required to achieve a speed-up. This is due to network error concentrating at fixed rate of $1/m$ while the optimal statistical rate is $(nm)^{-\frac{2r}{2r+\gamma}}$. An open question is then how to modify the algorithm to exploit regularity and achieve a speed-up runtime, similar to how Leverage Score Sampling exploits additional regularity (Bach, 2013; Avron et al., 2017; Rudi et al., 2018; Li et al., 2019).

To provide insight into how the conditions in Theorem 2 arise, the following theorem gives the leading order error terms which contribute to the conditions in Theorem 2.

Theorem 3 (Leading Order Terms) *Let $n, m, M \in \mathbb{N}_+$, $\delta \in (0,1)$, $t \geq 2t^\star \geq 4$, $\eta\kappa^2 \leq 1$ and $\eta \simeq 1$. Under assumptions 1 to 5 with $r + \gamma > 1$, the following holds with high probability when $t^\star \simeq \frac{1}{1-\sigma_2}$ for any $v \in V$*

$$\mathcal{E}(\widehat{f}_{t+1,v}) - \mathcal{E}(f_{\mathcal{H}}) \lesssim \underbrace{\frac{\eta^\gamma}{m(1-\sigma_2)^\gamma} + \frac{(\eta t)^2(\eta t^\star)^{1+\gamma}}{m^2}}_{\textit{Network Error}}$$
$$\underbrace{\Big(\frac{\eta t}{M}+1\Big)\frac{(\eta t)^\gamma}{nm} + \frac{1}{M(\eta t)^{(1-\gamma)(2r-1)}} + \Big(\frac{1}{\eta t}\Big)^{2r}}_{\textit{Statistical Error}} + \textit{H.O.T.}$$

where H.O.T. denotes Higher Order Terms.

Theorem 3 decomposes the Generalisation Error into two terms. The *Statistical Error* matches the Generalisation Error of Gradient Descent with Random Features (Carratino et al., 2018) and consists of Sample Variance, Random Feature and Bias errors. The *Network Error* arises from tracking the difference between the Distributed Gradient Descent $\widehat{\omega}_{t+1,v}$ and single machine Gradient Descent iterates. The primary technical contribution of our work is in the analysis of this term, in particular, building on (Richards & Rebeschini, 2019) in two directions. Firstly, bounds are given in high probability instead of expectation. Secondly, we give a tighter analysis of the Residual Network Error, here denoted in the second half of the *Network Error* as $(\eta t)^2(\eta t^\star)^{1+\gamma}/m^2$. Previously this term was of the order $(\eta t)^{2+\gamma}/m^2$ and gave rise to the condition of $r + \gamma/2 \geq 1$, whereas we now require $r + \gamma \geq 1$. Our analysis can ensure it is decreasing with the step size η, and thus, be controlled by taking a smaller step size. While not explored in this work, we believe our approach would be useful for analysing the Stochastic Gradient Descent variant (Lin & Rosasco, 2017) where a smaller step size is often chosen.

4. Error Decomposition and Proof Sketch

In this section we give a more detailed error decomposition as well as a sketch of the proof. Section 4.1 gives the error decomposition into statistical and network terms. Section 4.2 decomposes the network term into a population and a residual part. Section 4.3 and 4.4 give sketch proofs for bounding the population and residual parts respectively.

4.1. Error Decomposition

We begin by introducing the iterates produced by a single machine Gradient Descent with nm samples as well as an auxiliary sequence associated to the population. Initialised at $\widehat{v}_1 = \widetilde{v}_1 = 0$, we define, for $t \geq 1$

$$\widehat{v}_{t+1} = \widehat{v}_t - \frac{\eta}{nm}\sum_{w\in V}\sum_{i=1}^{m}\big(\langle \widehat{v}_{t,w}, \phi_M(x_{i,w})\rangle - y_{i,w}\big)\phi_M(x_{i,w}),$$
$$\widetilde{v}_{t+1} = \widetilde{v}_t - \eta\int_X \big(\langle \widetilde{v}_t, \phi_M(x)\rangle - y\big)\phi_M(x)d\rho(x,y).$$

We work with functions in $L^2(X, \rho_X)$, thus we define $\widehat{g}_t = \langle \widehat{v}_t, \phi_M(\cdot)\rangle$, $\widetilde{g}_t = \langle \widetilde{v}_t, \phi_M(\cdot)\rangle$. Since the prediction error can be written in terms of the $L^2(X, \rho_X)$ as follows $\mathcal{E}(\widehat{f}_{t,v}) - \mathcal{E}(f_{\mathcal{H}}) = \|\widehat{f}_{t,v} - f_{\mathcal{H}}\|_\rho^2$ we have the decomposition $\widehat{f}_{t,v} - f_{\mathcal{H}} = \widehat{f}_{t,v} - \widehat{g}_t + \widehat{g}_t - f_{\mathcal{H}}$. The term $\widehat{g}_t - f_{\mathcal{H}}$ that we call the *Statistical Error* is studied within (Carratino et al., 2018). The primary contribution of our work is in the analysis of $\widehat{f}_{t,v} - \widehat{g}_t$ which we call the *Network Error*, and go on to describe in more detail next.

4.2. Network Error

To accurately describe the analysis for the network error we introduce some notation. Begin by defining the operator $S_M : \mathbb{R}^M \to L^2(X, \rho_X)$ so that $(S_M\omega)(\cdot) = \langle \omega, \phi_M(\cdot)\rangle$ as well as the covariance $C_M : \mathbb{R}^M \to \mathbb{R}^M$ defined as $C_M = S_M^\star S_M$, where $S_M^\star$ is the adjoint of S_M in $L^2(X, \rho_X)$. Utilising an isometry property (see (7) in the Appendix) we have for $\omega \in \mathbb{R}^M$ the following $\|S_M\omega\|_\rho = \|C_M^{1/2}\omega\|$, that is going from a norm in $L^2(X, \rho_X)$ to Euclidean norm. The empirical covariance operator of the covariates held by agent $v \in V$ is denoted $\widehat{C}_M^{(v)} : \mathbb{R}^M \to \mathbb{R}^M$. For $t \geq 1$ and a path $w_{t:1} = (w_t, w_{t-1}, \dots, w_1) \in V^t$ denote the collection of contractions

$$\Pi(w_{t:1}) = (I - \eta\widehat{C}_M^{(w_t)})(I - \eta\widehat{C}_M^{(w_{t-1})})\dots(I - \eta\widehat{C}_M^{(w_1)})$$

as well as the centered product $\Pi^\Delta(w_{t:1}) = \Pi(w_{t:1}) - (I - \eta C_M)^t$. For $w \in V$ $k \geq 1$ let $N_{k,w} \in \mathbb{R}^M$ denote a collection of zero mean random variables that are independent across agents $w \in V$ but not index $k \geq 1$.

For $v, w \in V$ and $s \geq 1$ define the difference $\Delta^s(v,w) := P_{vw}^s - \frac{1}{n}$, where we apply the power then index i.e.

$(P^s)_{vw} = P^s_{vw}$. For $w_{t:k} \in V^{t-k}$ denote the deviation along a path $\Delta(w_{t:k}) = P_{vw_{t:k}} - \frac{1}{n^{t-k}}$ where we have written the probability for a path $P_{vw_{t:k}} = P_{vw_t}P_{w_tw_{t-1}}\dots P_{w_{k+1}w_k}$.

Following (Richards & Rebeschini, 2019), center the distributed $\omega_{t+1,v}$ and the single machine iterates $\widehat{v}_{t+1}$ around the population iterates $\widetilde{v}_t$. Apply the isometry property to $\|\widehat{f}_{t,v} - \widehat{g}_t\|_\rho = \|C_M^{1/2}(\widehat{\omega}_{t+1,v} - \widehat{v}_t)\|$ and following the steps in Appendix D.1 we arrive at

$$\|C_M^{1/2}(\widehat{\omega}_{t+1,v} - \widehat{v}_{t+1})\| \leq \underbrace{\sum_{k=1}^{t}\eta\sum_{w\in V}|\Delta^{t-k}(v,w)|\|C_M^{1/2}(I-\eta C_M)^{t-k}N_{k,w}\|}_{\text{Population Network Error}}$$
$$\underbrace{\sum_{k=1}^{t}\eta\Big\|\sum_{w_{t:k}\in V^{t-k+1}}\Delta(w_{t:k})C_M^{1/2}\Pi^\Delta(w_{t:k+1})N_{k,w_k}\Big\|}_{\text{Residual Network Error}}.$$

The two terms above can be associated to the two terms in the network error of Theorem 3, with the Population Network Error decreasing as $1/m$ and the Residual Network Error as $1/m^2$. We now analyse each of these terms separately.

4.3. Network Error: Population

Our contribution for analysing the *Population Network Error* is to give bounds it in high probability, where as (Richards & Rebeschini, 2019) only gave bounds in expectation. Choosing some $t \geq 2t^\star \geq 2$ and splitting the series at $k = t - t^\star$ we are left with two terms. For $1 \leq k \leq t - t^\star$ we utilise that the sum over the difference $|\Delta^s(v,w)|$ can be written in terms of euclidean ℓ_1 norm and this is bounded by the second largest eigenvalue of P in absolute value i.e. $\sum_{w\in V}|\Delta^{t-k}(v,w)| = \|e_v^\top P^{t-k} - \frac{1}{n}\mathbf{1}\|_1 \leq \sqrt{n}\sigma_2^{t-k} \leq \sqrt{n}\sigma_2^{t^\star}$, where e_v is the standard basis vector in $\mathbb{R}^n$ with a 1 aligning with agent $v \in V$ and $\mathbf{1}$ is a vector of all 1's. Meanwhile for $t \geq k \geq t - t^\star$, we follow (Richards & Rebeschini, 2019) and utilise the contraction of the gradient updates i.e. $C_M^{1/2}(I - \eta C_M)^{t-k}$ alongside that N_{k,w_k} is an average of m i.i.d. random variables, and thus, concentrate at $1/\sqrt{m}$ in high probability. This leads to the bound in high probability

$$\text{Population Network Error} \lesssim \underbrace{\frac{\sqrt{n}\sigma_2^{t^\star}t}{\sqrt{m}}}_{\text{Well Mixed Terms}} + \underbrace{\frac{(\eta t^\star)^{\gamma/2}}{\sqrt{m}}}_{\text{Poorly Mixed Terms}}.$$

The first term *Well Mixed*, decays exponentially with the second largest eigenvalue of P in absolute value, and represents the information from past iterates that has now fully propagated around the network. The term *Poorly Mixed* represents error from the most recent iterates that is yet to fully propagate through the network. It grows at the rate $(t^\star)^{\gamma/2}$ due to utilising the contractions of the gradients as well as the assumptions 5. The quantity $t^\star$ is now chosen to trade off these terms. Note by writing $\sigma_2^{t^\star} = e^{-t^\star\log(1/\sigma_2)}$ that, up to logarithmic factors, the first can be made small by taking $t^\star \gtrsim \frac{1}{1-\sigma_2} \geq \frac{1}{-\log(\sigma_2)}$.

4.4. Network Error: Residual

The primary technical contribution of our work is in the analysis of this term. The analysis builds on insights from (Richards & Rebeschini, 2019), specifically that $\Pi^\Delta(w_{t:1})$ is a product of empirical operators minus the population, and thus, can be written in terms of the differences $\widehat{C}_M^{(w)} - C_M$ which concentrate at $1/\sqrt{m}$. Specifically, for $N \in \mathbb{R}^M$, the bound within (Richards & Rebeschini, 2019) was of the following order with high probability for any $w_{t:1} \in V^t$

$$\|C_M^{1/2}\Pi^\Delta(w_{t:1})N\| \lesssim \|N\|\frac{(\eta t)^{\gamma/2}}{\sqrt{m}}. \tag{6}$$

The bound for Residual Network Error within (Richards & Rebeschini, 2019) is arrived at by applying triangle inequality over the series $\sum_{w_{t:k}\in V^{t-k+1}}$, plugging in (6) for $\|C_M^{1/2}\Pi^\Delta(w_{t:k+1})N_{k,w_k}\|$ alongside $\|N_{k,w_k}\| \lesssim 1/\sqrt{m}$ see Lemma 7 in Appendix. Summing over $1 \leq k \leq t$ yields the bound of order $(\eta t)^{1+\gamma/2}/m$ in high probability. The two key insights of our analysis are as follows. Firstly, noting that the error for bounding the contraction $\Pi^\Delta(w_{t:1})$ grows with the length of the path, and as such, we should aim to apply the bound (6) to short paths. Secondly, note for $N \in \mathbb{R}^M$ quantities of the form $\|C_M^{1/2}\sum_{w_{t:1}\in V^t}\Delta(w_{t:1})\Pi^\Delta(w_{t:1})N\|$ concentrate quickly (Lemma 13 in Appendix).

To apply the insights outlined previously, we decompose the deviation $\Pi^\Delta(w_{t:2})$ into two terms that only replace the final $t^\star$ operators with the population, that is

$$\Pi^\Delta(w_{t:2}) = \Pi(w_{t:t^\star+2})\Pi^\Delta(w_{t^\star+1:1}) + \Pi^\Delta(w_{t:t^\star+2})(I-\eta C_M)^{t^\star}.$$

Plugging in the above then yields, for the case $k = 1$,

$$\sum_{w_{t:1}\in V^t}\Delta(w_{t:1})C_M^{1/2}\Pi^\Delta(w_{t:2})N_{k,w_1}$$
$$= \sum_{w_{t:1}\in V^t}\Delta(w_{t:1})C_M^{1/2}\Pi(w_{t:t^\star+2})\underbrace{\Pi^\Delta(w_{t^\star+1:1})}_{t^\star\text{contraction}}N_{k,w_1}$$
$$+\sum_{w_{t:1}\in V^t}\Delta(w_{t:1})\underbrace{C_M^{1/2}\Pi^\Delta(w_{t:t^\star+2})(I-\eta C_M)^{t^\star}N_{k,w_1}}_{\text{Independent of } w_{t^\star+1:1}}$$

Note that the first term above only contains a contraction $\Pi^\Delta(w_{t^\star+1:1})$ of length $t^\star$, and as such, when applying a variant of (6) will only grow at length $(\eta t^\star)^{(1+\gamma)/2}/\sqrt{m}$.

When summing over $1 \leq k \leq t$ this will result in the leading order term for the residual error of $(\eta t)(\eta t^\star)^{(1+\gamma)/2}/m$. For the second term, note the highlighted section is independent of the final $t^\star$ steps of the path $w_{t:1}$, namely $w_{t^\star+1:1}$. Therefore we can sum the deviation $\Delta(w_{t:1})$ over path $w_{t^\star+1:1}$ and, if $t^\star \gtrsim \frac{1}{1-\sigma_2}$, replace N_{k,w_1} by the average $\frac{1}{n}\sum_{w\in V} N_{k,w}$. This has impact of decoupling the summation over the remainder of the path $w_{t:t^\star}$ allowing the second insight from previously to be used. For details on this step we point the reader to Appendix Section D.1.

5. Experiments

For our experiments we consider subsets of the SUSY data set (Baldi et al., 2014), as well as single machine and Distributed Gradient Descent with a fixed step size $\eta = 1$. Cycle and grid network topologies are studied, with the matrix P being a simple random walk. Random Fourier Features are used $\psi(x, \omega) = \cos(\xi \times w^\top x + q)$, with $\omega := (w, q)$, w sampled according to the normal distribution, q sampled uniformly at random between 0 and 2π, and ξ is a tuning parameter associated to the bandwidth (fixed to $\xi = 10^{-1/2}$). For any given sample size, topology or network size we repeated the experiment 5 times. Test size of 10^4 was used and classification error is minimum over iterations and maximum over agents i.e. $\min_t \max_{v\in V} \mathcal{E}_{\text{Approx}}(\widehat{\omega}_{t,v})$, where $\mathcal{E}_{\text{Approx}}$ is approximated test error. With the response of the data being either 1 or 0 and the predicted response $\widehat{y}$, the predicted classification is the indicator function of $\widehat{y} > 1/2$. The classification error is the proportion of mis-classified samples.

We begin by investigating the number of Random Features required with Distributed Gradient Descent to match the single machine performance. Looking to Figure 1, observe that for a grid topology, as well as small cycles ($n = 9, 25$), that the classification error aligns with a single machine beyond approximately $\sqrt{nm}$ Random Features. For larger more poorly connected topologies, in particular a cycle with $n = 49$ agents, we see that the error does not fully decrease down that of single machine Gradient Descent.

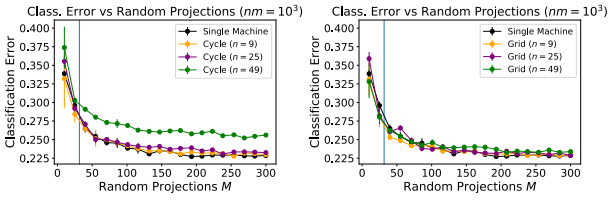

Figure 1. Classification Error (if y and $\widehat{y}$ are the true and predicted response respectively, error calculated is 0-1 loss) against number of Random Features M, with total sample size and maximum number of iterations $t = nm = 10^3$. Vertical line in plots indicates $\sqrt{nm}$. *Left*: Cycle topology, *Right*: Grid Topology.

Our theory predicts that the sub-optimality of more poorly connected networks decreases as the number of samples held by each agent increases. To investigate this, we repeat the above experiment for cycles and grids of sizes $n = 25, 49, 100$ while varying the dataset size. Looking to Figure 2, we see that approximately $nm \approx 10^3$ samples are sufficient for a cycle topology of size $n = 49$ to align with a single machine, meanwhile 10^4 samples are required for a larger $n = 100$ cycle. For a grid we see a similar phenomena, although with fewer samples required due to being better connected topology.

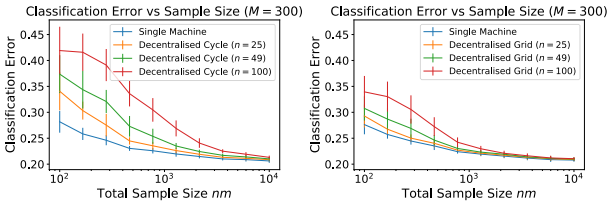

Figure 2. Plots of Classification Error (computed as in Figure 1) against total number of samples nm, with $M = 300$. Run for at most $t = 10^4$ iterations, each point is an average of 20 sub-subsets of the SUSY, which Distributed Gradient Descent with Random Features is run on 5 times.

Our theory predicts that, given sufficiently many samples, the number of iterations for any network topology scales as those of single machine Gradient Descent. We look to Figure 3 where the number of iterations required to achieve the minimum classification error (optimal stopping time) is plotted against the sample size. Observe that beyond approximately 10^3 samples both grid and cycles of sizes $n = 25, 49, 100$ have iterates that scale at the same order as a single machine. Observe that the number of iterations required by the grid and cycle topologies is initially decreasing with the sample size up to 10^3. While not supported by our theory for a constant step size, this suggests quantities held by agents become similar as agents hold more data, reducing the need for additional iterations in order to propagate information around the network. An analysis of this phenomena we leave to future work.

6. Conclusion

In this work we considered the performance of Distributed Gradient Descent with Random Features on the Generalisation Error, this being different from previous works which focused on training loss. Our analysis allowed us to understand the role of different parameters on the Generalisation error, and, when agents have sufficiently many samples with respect to the network size, achieve a linear speed-up in runtime time for any network topology.

Moving forward, it would be natural to extend our analysis to stochastic gradients (Lin & Rosasco, 2017) or stochastic

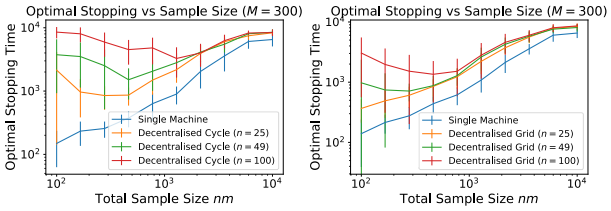

Figure 3. Optimal Stopping Time (Number of iterations required) against sample size nm ($\log - \log$ axis), with $M = 300$. *Left*: Cycle Topology, *Right*: Grid topology. Each point is averaged over 20 sub-subsets of the SUSY. Distributed Gradient Descent with Random Features was repeated 5 times, with at most 10^4 iterations.

communication at each iteration (Shah, 2009).

Acknowledgements

D.R. is supported by the EPSRC and MRC through the OxWaSP CDT programme (EP/L016710/1). Part of this work has been carried out at the Machine Learning Genoa (MaLGa) center, Università di Genova (IT). L.R. acknowledges the financial support of the European Research Council (grant SLING 819789), the AFOSR projects FA9550-17-1-0390 and BAA-AFRL-AFOSR-2016-0007 (European Office of Aerospace Research and Development), and the EU H2020-MSCA-RISE project NoMADS - DLV-777826.

References

Aronszajn, N. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

Avron, H., Kapralov, M., Musco, C., Musco, C., Velingker, A., and Zandieh, A. Random Fourier features for kernel ridge regression: Approximation bounds and statistical guarantees. In *34th International Conference on Machine Learning*, pp. 253–262. PMLR, 2017.

Bach, F. Sharp analysis of low-rank kernel matrix approximations. In *Conference on Learning Theory*, pp. 185–209, 2013.

Baldi, P., Sadowski, P., and Whiteson, D. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5:4308, 2014.

Blanchard, G. and Mücke, N. Optimal rates for regularization of statistical inverse learning problems. *Foundations of Computational Mathematics*, 18(4):971–1013, 2018.

Bouboulis, P., Chouvardas, S., and Theodoridis, S. Online distributed learning over networks in rkh spaces using random fourier features. *IEEE Transactions on Signal Processing*, 66(7):1920–1932, 2017.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, January 2011.

Caponnetto, A. and De Vito, E. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, 7(3):331–368, 2007.

Carratino, L., Rudi, A., and Rosasco, L. Learning with sgd and random features. In *Advances in Neural Information Processing Systems*, pp. 10192–10203, 2018.

Chouvardas, S. and Draief, M. A diffusion kernel lms algorithm for nonlinear adaptive networks. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4164–4168. IEEE, 2016.

Cucker, F. and Zhou, D. X. *Learning theory: an approximation theory viewpoint*, volume 24. Cambridge University Press, 2007.

Dieuleveut, A., Bach, F., et al. Nonparametric stochastic approximation with large step-sizes. *The Annals of Statistics*, 44(4):1363–1399, 2016.

Dobriban, E. and Sheng, Y. Wonder: Weighted one-shot distributed ridge regression in high dimensions. *Journal of Machine Learning Research*, 21(66):1–52, 2020.

Duchi, J. C., Agarwal, A., and Wainwright, M. J. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic Control*, 57(3):592–606, 2012.

Forero, P. A., Cano, A., and Giannakis, G. B. Consensus-based distributed support vector machines. *Journal of Machine Learning Research*, 11(May):1663–1707, 2010.

Gao, W., Chen, J., Richard, C., and Huang, J. Diffusion adaptation over networks with kernel least-mean-square. In *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pp. 217–220. IEEE, 2015.

Guo, Z.-C., Lin, S.-B., and Zhou, D.-X. Learning theory of distributed spectral algorithms. *Inverse Problems*, 33(7): 074009, 2017.

Jakovetić, D., Moura, J. M., and Xavier, J. Linear convergence rate of a class of distributed augmented lagrangian algorithms. *IEEE Transactions on Automatic Control*, 60 (4):922–936, 2015.

Johansson, B., Rabi, M., and Johansson, M. A simple peer-to-peer algorithm for distributed optimization in sensor networks. In *Decision and Control, 2007 46th IEEE Conference on*, pp. 4705–4710. IEEE, 2007.

Johansson, B., Rabi, M., and Johansson, M. A randomized incremental subgradient method for distributed optimization in networked systems. *SIAM Journal on Optimization*, 20(3):1157–1170, 2009.

Koppel, A., Paternain, S., Richard, C., and Ribeiro, A. Decentralized online learning with kernels. *IEEE Transactions on Signal Processing*, 66(12):3240–3255, 2018.

Le, Q., Sarlós, T., and Smola, A. Fastfood-computing hilbert space expansions in loglinear time. In *International Conference on Machine Learning*, pp. 244–252, 2013.

Li, Z., Ton, J.-F., Oglic, D., and Sejdinovic, D. Towards a unified analysis of random fourier features. In *International Conference on Machine Learning*, pp. 3905–3914, 2019.

Lin, J. and Cevher, V. Optimal convergence for distributed learning with stochastic gradient methods and spectral-regularization algorithms. *arXiv preprint arXiv:1801.07226*, 2018.

Lin, J. and Rosasco, L. Optimal rates for multi-pass stochastic gradient methods. *Journal of Machine Learning Research*, 18(97):1–47, 2017.

Lin, S.-B., Guo, X., and Zhou, D.-X. Distributed learning with regularized least squares. *The Journal of Machine Learning Research*, 18(1):3202–3232, 2017.

Lobel, I. and Ozdaglar, A. Distributed subgradient methods for convex optimization over random networks. *IEEE Transactions on Automatic Control*, 56(6):1291–1306, 2011.

Matei, I. and Baras, J. S. Performance evaluation of the consensus-based distributed subgradient method under random communication topologies. *IEEE Journal of Selected Topics in Signal Processing*, 5(4):754–771, 2011.

Mitra, R. and Bhatia, V. The diffusion-klms algorithm. In *2014 International Conference on Information Technology*, pp. 256–259. IEEE, 2014.

Mokhtari, A. and Ribeiro, A. Dsa: Decentralized double stochastic averaging gradient algorithm. *Journal of Machine Learning Research*, 17(61):1–35, 2016.

Mücke, N. and Blanchard, G. Parallelizing spectrally regularized kernel algorithms. *The Journal of Machine Learning Research*, 19(1):1069–1097, 2018.

Nedic, A. and Ozdaglar, A. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

Nedić, A., Olshevsky, A., Ozdaglar, A., and Tsitsiklis, J. N. On distributed averaging algorithms and quantization effects. *IEEE Transactions on Automatic Control*, 54(11): 2506–2517, 2009.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pp. 1177–1184, 2008.

Reed, M. *Methods of modern mathematical physics: Functional analysis*. Elsevier, 2012.

Richards, D. and Patrick, R. Graph-dependent implicit regularisation for distributed stochastic subgradient descent. *Journal of Machine Learning Research*, 21(2020):1–44, 2020.

Richards, D. and Rebeschini, P. Optimal statistical rates for decentralised non-parametric regression with linear speed-up. In *Advances in Neural Information Processing Systems*, pp. 1216–1227, 2019.

Rudi, A. and Rosasco, L. Generalization properties of learning with random features. In *Advances in Neural Information Processing Systems*, pp. 3215–3225, 2017.

Rudi, A., Calandriello, D., Carratino, L., and Rosasco, L. On fast leverage score sampling and optimal learning. In *Advances in Neural Information Processing Systems*, pp. 5672–5682, 2018.

Scaman, K., Bach, F., Bubeck, S., Lee, Y. T., and Massoulié, L. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In *34th International Conference on Machine Learning*, pp. 3027–3036. PMLR, 2017.

Schölkopf, B., Herbrich, R., and Smola, A. J. A generalized representer theorem. In *International conference on computational learning theory*, pp. 416–426. Springer, 2001.

Shah, D. Gossip algorithms. *Foundations and Trends® in Networking*, 3(1):1–125, 2009.

Shi, W., Ling, Q., Yuan, K., Wu, G., and Yin, W. On the linear convergence of the admm in decentralized consensus optimization. *IEEE Trans. Signal Processing*, 62(7): 1750–1761, 2014.

Shi, W., Ling, Q., Wu, G., and Yin, W. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

Steinwart, I. and Christmann, A. *Support vector machines*. Springer Science & Business Media, 2008.

Tsitsiklis, J., Bertsekas, D., and Athans, M. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9):803–812, 1986.

Tsitsiklis, J. N. Problems in decentralized decision making and computation. Technical report, Massachusetts Inst Of Tech Cambridge Lab For Information And Decision Systems, 1984.

Xu, P., Wang, Y., Chen, X., and Zhi, T. Coke: Communication-censored kernel learning for decentralized non-parametric learning. *arXiv preprint arXiv:2001.10133*, 2020.

Yu, F. X. X., Suresh, A. T., Choromanski, K. M., Holtmann-Rice, D. N., and Kumar, S. Orthogonal random features. In *Advances in Neural Information Processing Systems*, pp. 1975–1983, 2016.

Zhang, J., May, A., Dao, T., and Ré, C. Low-precision random fourier features for memory-constrained kernel approximation. *Proceedings of machine learning research*, 89:1264, 2019.

Zhang, Y., Duchi, J., and Wainwright, M. Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates. *The Journal of Machine Learning Research*, 16(1):3299–3340, 2015.