Revisiting Random Binning Feature: Fast Convergence and Strong Parallelizability

Lingfei Wu [*]
College of William and Mary
Williamsburg, VA 23185
lfwu@cs.wm.edu

Ian E.H. Yen [†]
University of Texas at Austin
Austin, TX 78712
ianyen@cs.utexas.edu

Jie Chen
IBM Research
Yorktown Heights, NY 10598
chenjie@us.ibm.com

Rui Yan
Baidu Inc.
Beijing 100085, China
yanrui02@baidu.com

ABSTRACT

Kernel method has been developed as one of the standard approaches for nonlinear learning, which however, does not scale to large data set due to its quadratic complexity in the number of samples. A number of kernel approximation methods have thus been proposed in the recent years, among which the random features method gains much popularity due to its simplicity and direct reduction of nonlinear problem to a linear one. Many different random basis functions have since been proposed to approximate different types of kernel. Among them the Random Binning (RB) features, proposed in the first random features paper [19], has drawn significantly less attention than that of Random Fourier (RF) features proposed also in [19]. However, in this work we observe the RB approach, with right choice of optimization solver, could be orders of magnitude faster than other random features and kernel approximation methods to achieve the same accuracy. We thus propose the first analysis of RB from the perspective of optimization, which by interpreting RB as a Randomized Block Coordinate Descent in the infinite-dimensional space, gives a faster convergence compared to that of other random features. In particular, we show that by drawing R grids with at least κ expected number of non-empty bins per grid, RB achieves a convergence rate of $O(1/\kappa R)$, which is not only better than the existing $O(1/\sqrt{R})$ rate from Monte Carlo analysis, but also shows a κ times speedup over other random features under the same analysis framework. In addtion, we demonstrate another advantage of RB in the L1-regularized setting, where unlike other random features, a RB-based Coordinate Descent solver can be parallelized with guaranteed speedup proportional to κ. Our extensive experiments demonstrate the superior performance of the RB features over other random features and kernel approximation methods.

[*]Both authors contributed equally to this manuscript

[†]Both authors contributed equally to this manuscript

KDD '16 August 13-17, San Francisco, CA, USA
© 2016 ACM. ISBN 123-4567-24-567/08/06...$15.00
DOI: 10.475/123_4

Keywords

Kernel approximation, Random Binning Features, large-scale machine learning, faster convergence, strong parallelizability

1. INTRODUCTION

Kernel methods have great promise for learning non-linear model from simple data input representations and have been demonstrated to be successful for solving various problems in machine learning and data mining, ranging from regression, classification, feature extraction, clustering and dimensionality reduction [24, 28]. However, they are typically not first choice for large-scale nonlinear learning problems since large amounts of modern datasets present significant challenges to both computation and memory consumptions for computing the dense kernel matrix $K \in \mathcal{R}^{N\times N}$. It requires $O(N^2)$ to store the matrix, and takes at least $O(N^2)$ or $O(N^3)$ computational costs depending on if the iterative solvers or direct solvers are employed. To scale up the kernel methods, there have been many great efforts to address this challenge, by applying advance knowledge from various areas such as numerical linear algebra and functional analysis [4].

From the perspective of numerical linear algebra, a line of research [29, 26, 7, 25] has been devoted to directly approximate the kernel matrix using low-rank factorizations, $K \approx Z^T Z$, where $Z \in \mathcal{R}^{N\times R}$ and $R \leq N$. Among them, the Nyström method and its fruitful variants [29, 5, 11, 8, 25] are probably one of the most popular methods. Depending on if the subsequent kernel algorithms to operate on K explicitly or implicitly through Z, the total computational costs are $O(NRd+NR^2+R^3)$ or $O(NRd+NRk)$ by using $O(NR)$ storage, where d and k are the input data dimensions and the number of the iterations using iterative solvers, respectively. However, in practice, the convergence of these low-rank kernels could be slow since the approximation error of the objective function is proportional to $O(1/\sqrt{R} + 1/\sqrt{N})$ [5, 31]. It implies that the rank R may need to be near-linear to the number of data points in order to achieve less performance loss compared to the vanilla kernel method. For very large-scale problems like spatial data, the low-rank approximation based methods become almost as expensive as the exact kernel method when R grows close to N in order to maintain a competitive performance [27].

Another popular approach for scaling up kernel methods is to using random features approximation [19, 20]. Unlike the previous approach to approximate kernel matrix, the method approximates the kernel function directly by using explicit feature maps. In particular, Random Fourier (RF) features has attracted considerable interests due to its easy implementation and fast execution time [20, 12, 30, 4]. The merit of this approach is that the total computational costs and storage requirements are $O(NRd + NRk)$ and $O(NR)$ respectively, for computing feature matrix Z and operating the subsequent kernel algorithms on Z. Fastfood and its extension [12, 30] improve the prediction time of random features from $O(Rd)$ to $O(R\log d)$ by leveraging the Hadamard basis functions instead of using the Fourier basis functions. Although RF features has been successfully applied to speech recognition and vision classifications for very large datasets [3, 10, 16], a drawback is that a significant large number of random features are needed to achieve a comparable performance. This is not surprising since the convergence of approximation error is the order of $O(1/\sqrt{R} + 1/\sqrt{N})$ [20, 4], which is the same as that of low-rank kernel approximations in the previous approach.

The core idea of Mercer's theorem [18] states that any positive definite kernel can be expressed as a decomposition of a set of basis functions for some feature maps. However, the decomposition is not unique and one may find different basis functions to composite the same kernel [31]. Therefore, we ask following question: can some of the basis functions converge faster than the others ? To answer this question, in this paper, we reconsider Random Binning (RB) features for scaling up the kernel methods. Our main contributions are fourfold.

First, we propose the first analysis of RB from the perspective of optimization. By interpreting RB as a *Randomized Block Coordinate Descent* (RBCD) in the Reproducing Kernel Hilbert Space (RKHS) induced from the kernel, we prove that RB enjoys faster convergence than other random features. Specifically, by drawing R grids with at least κ expected number of non-empty bins per grid, RB can achieve a solution comparable to the exact kernel method within $O(1/\kappa R)$ precision in terms of the objective function, which is not only better than the existing $O(1/\sqrt{R})$ rate from Monte Carlo analysis [19], but also shows a κ times speedup over other random features under the same analysis framework [31].

Second, we exploit the sparse structure of the feature matrix Z, which is the key to rapidly transform the data features into a very high-dimension feature space that is linearly separately by any regressors and classifiers. In addition, we discuss how to efficiently perform the computation for a large, sparse matrix by using state-of-the-art iterative solvers and advanced matrix storage techniques. As a result, the computational complexity and storage requirements in training are still $O(NRd + NRk)$ and $O(NR)$, respectively.

Third, we also propose to combine the RB features with L1-regularization to produce sparse nonlinear predictor that have faster prediction and more compact representation, yielding the Sparse Random Binning Features algorithm. We show that a state-of-the-art *Randomized Coordinate Descent* (RCD) solver with RB can be parallelized with guaranteed speedup proportional to κ.

Fourth, we provides extensive sets of experiments to demonstrate the efficiency and effectiveness of RB. Compared to other popular low-rank approximations, RB shows the superior performance for regression and classification due to its much faster convergence. In particular, for obtaining same performance, RB achieves between one and three orders of magnitude faster and memory savings than other approximated kernels. Finally, we demonstrate the strong parallel performance of RB over RF by using RCD solver as a result of the high sparsity in Z.

2. RANDOM BINNING FEATURE AS KERNEL APPROXIMATION

In this work, we consider the problem of fitting a nonlinear prediction function $f : \mathcal{X} \rightarrow \mathcal{Y}$ in the RKHS $\mathcal{H}$ from training data $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ of pairs, where $\boldsymbol{x}_n \in \mathcal{X} \subseteq \mathbb{R}^d$ and $y_n \in \mathcal{Y}$ via approach of Empirical Risk Minimization (ERM)

$$f^* = \underset{f\in\mathcal{H}}{argmin} \quad \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 + \frac{1}{N}\sum_{n=1}^{N} L(f(\boldsymbol{x}_n), y_n), \tag{1}$$

where $L(z, y)$ is a convex loss function with Lipschitz-continuous derivative satisfying $|L'(z_1, y) - L'(z_2, y)| \leq \beta|z_1 - z_2|$, which includes several standard loss functions such as the *square-loss* $L(z, y) = \frac{1}{2}(z - y)^2$, *square-hinge loss* $L(z, y) = \max(1 - zy, 0)^2$ and *logistic loss* $L(z, y) = \log(1 + \exp(-yz))$.

2.1 Learning in Reproducing Kernel Hilbert Space

The space $\mathcal{H}$ can be specified through the choice of a positive-definite (PD) kernel function $k(\boldsymbol{x}_1, \boldsymbol{x}_2)$ that measures the similarity between samples by defining $\mathcal{H}$ as the the space spanned by:

$$\mathcal{H} = \left\{ f(\cdot) = \sum_{i=1}^{K} \alpha_i k(\boldsymbol{x}_i, \cdot) \mid \alpha_i \in \mathbb{R}, \boldsymbol{x}_i \in \mathcal{X} \right\}. \tag{2}$$

The other way is to find a possibly infinite-dimensional feature map $\{\bar{\phi}_h(\boldsymbol{x})\}_{h\in H}$ with each $h \in H$ that defines a basis function $\bar{\phi}_h(\boldsymbol{x}) : \mathcal{X} \rightarrow \mathbb{R}$ s.t. the space can be represented as

$$\mathcal{H} = \left\{ f(\cdot) = \int_{h\in H} w(h)\bar{\phi}_h(\cdot)dh = \langle \boldsymbol{w}, \bar{\boldsymbol{\phi}}(\cdot)\rangle_{\mathcal{H}} \mid \|f\|_{\mathcal{H}}^2 < \infty \right\}, \tag{3}$$

where $w(h)$ is a weighting function over the basis $\{\phi_h(\boldsymbol{x})\}_{h\in\mathcal{H}}$. The Mercer's theorem [18] connects the above two formulation of RKHS by stating that every PD kernel $k(\boldsymbol{x}_1, \boldsymbol{x}_2)$ can be expressed as an integration over basis functions for some feature map $\{\phi_h(.)\}_{h\in H}$ as

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \int_{h\in H} p(h)\phi_h(\boldsymbol{x}_1)\phi_h(\boldsymbol{x}_2)dh = \langle \bar{\boldsymbol{\phi}}(\boldsymbol{x}_1), \bar{\boldsymbol{\phi}}(\boldsymbol{x}_2)\rangle_{\mathcal{H}}, \tag{4}$$

However, the decomposition (4) is not unique, so one can find different feature maps $\{\phi_h(.)\}_{h\in H}$ with different distribution $p(h)$ for the same kernel $k(.,.)$ to let equality (4) hold. In particular, as an example used extensively in this work, the Laplacian Kernel

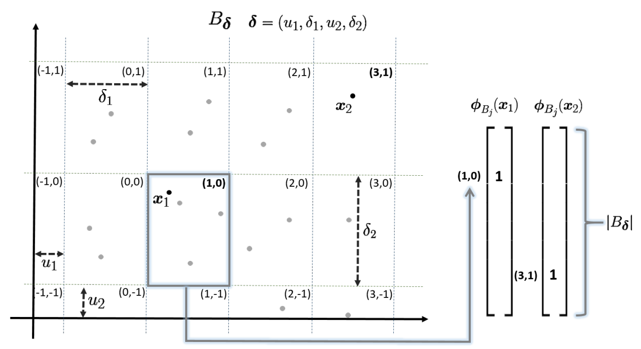
$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp\left(-\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_1}{\sigma}\right), \tag{5}$$

allows decomposition based on (i) Fourier basis map [19], (ii) RB map [19], and also (iii) map based on infinite number of decision trees [14] to name a few. On the other hand, the different kernels can be constructed from the same set of basis function through (4) with different distribution $p(h)$. For example, the RB feature map can be used to construct shift-invariant kernel of the form [19]

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = K(\boldsymbol{x}_1 - \boldsymbol{x}_2) = \prod_{j=1}^{d} k_j(x_{1j} - x_{2j}), \tag{6}$$

by sampling the "width" of bins δ_j for each feature j from a distribution proportional to $\delta k''(\delta)$, where $k''(\delta)$ is the second derivative of $k(\delta)$ w.r.t. δ, assuming the kernel has a non-negative second derivative.

2.2 Random Binning Features



$B_{\boldsymbol{\delta}}$ $\boldsymbol{\delta} = (u_1, \delta_1, u_2, \delta_2)$
(-1,1)
(0,1)
(1,1)
(2,1)
(3,1)
δ_1
$\boldsymbol{x}_2$
(-1,0)
(0,0)
(1,0)
(2,0)
(3,0)
$\boldsymbol{x}_1$
δ_2
u_1
(-1,-1)
(0,-1)
(1,-1)
(2,-1)
(3,-1)
u_2
$\phi_{B_j}(\boldsymbol{x}_1)$
$\phi_{B_j}(\boldsymbol{x}_2)$
(1,0) 1
(3,1) 1
$|B_{\boldsymbol{\delta}}|$

Figure 1: Generating process of RB features.

In this section, we describe the RB features [19], which considers feature map of the form

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \int_{\boldsymbol{\delta}} p(\boldsymbol{\delta}) \boldsymbol{\phi}_{B_{\boldsymbol{\delta}}}(\boldsymbol{x}_1)^T \boldsymbol{\phi}_{B_{\boldsymbol{\delta}}}(\boldsymbol{x}_2)\, d\boldsymbol{\delta} \tag{7}$$

where $B_{\boldsymbol{\delta}}$ is a grid parameterized by $\boldsymbol{\delta} = (\delta_1, u_1, ..., \delta_d, u_d)$ that specifies the *width* and *bias* of the grid w.r.t. the d dimensions, and $\boldsymbol{\phi}_{B_{\boldsymbol{\delta}}}(\boldsymbol{x})$ is a vector which has, for $b \in B_{\boldsymbol{\delta}}$,

$$\phi_b(\boldsymbol{x}) = 1, \; \textit{if}\; b = (\lfloor \frac{x_{n1} - u_1}{\delta_1} \rfloor, ..., \lfloor \frac{x_{nd} - u_d}{\delta_d} \rfloor),$$

that is, when $\boldsymbol{x}$ lies in the bin $b \in B_{\boldsymbol{\delta}}$, and $\phi_b(\boldsymbol{x}) = 0$ for any other bin $b \in B_{\boldsymbol{\delta}}$. Note for each grid $B_{\boldsymbol{\delta}}$ the number of bins $|B_{\boldsymbol{\delta}}|$ is countably infinite, so $\boldsymbol{\phi}_{B_{\boldsymbol{\delta}}}(\boldsymbol{x})$ has infinite dimensions but only 1 non-zero entry (at the bin $\boldsymbol{x}$ lies in). Figure 1 illustrates an example when the raw dimension $d = 2$. The kernel $K(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is thus interpreted as the *collision probability* that two data points $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ fall in the same bin, when the grid is generated from distribution $p(\boldsymbol{\delta})$. In [19], it is pointed out for any kernel of form (6) with *nonnegative second derivative* $k_j''(\delta)$, one can derive distribution $p(\boldsymbol{\delta}) = \prod_{j=1}^d p_j(\delta_j) U(u_j; 0, \delta_j)$, where $p_j(\delta_j) \propto \delta k_j''(\delta_j)$ and $U(\cdot, a, b)$ is uniform distribution in the range $[a, b]$.

To obtain a kernel approximation scheme from the feature map (7), a simple Monte Carlo method can be used to approximate (7) by averaging over R grids $\{B_{\boldsymbol{\delta}_r}\}_{r=1}^R$ with each grid's parameter $\boldsymbol{\delta}_r$ drawn from $p(\boldsymbol{\delta})$. The procedure for generating R RB features from raw data $\{\boldsymbol{x}_n\}_{n=1}^N$ is given in Algorithm 1.

Using a Hoefding bound, one can show the Monte-Carlo approximation to (7) yields an $O(1/\sqrt{R})$ approximation error. From the Representer theorem, one can further bound error of the learned predictor

$$\left| \boldsymbol{w}_{RF}^T \boldsymbol{z}(\boldsymbol{x}) - f^*(\boldsymbol{x}) \right| = \left| \sum_{n=1}^N \alpha_n^{RF} \boldsymbol{z}(\boldsymbol{x}_n)^T \boldsymbol{z}(\boldsymbol{x}) - \sum_{n=1}^N \alpha_n^* k(\boldsymbol{x}_n, \boldsymbol{x}) \right|$$

as shown in the appendix of [19]. Unfortunately, the rate of convergence suggests that to achieve small approximation error ϵ, one needs significant amount of random features $\Omega(1/\epsilon^2)$, and furthermore, the Monte-Carlo analysis does not explain why empirically RB feature achieves faster convergence than other random feature map like Fourier basis, sometimes by orders of magnitude.

3. FASTER CONVERGENCE OF RANDOM BINNING

In this section, we firstly illustrate the sparse structure of the feature matrix Z of RB and then discuss how to make efficient computation and storage format of Z. By interpreting the RB features as RBCD in the infinite-dimensional space, we prove that RB has a faster convergence rate than other random features. We illustrate them accordingly in the following sections.

Algorithm 1 Random Binning Features

```
Given a kernel function $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \prod_{j=1}^d k_j(|x_{1j} - x_{2j}|)$. Let $p_j(\delta) \propto \delta k_j''(\delta)$ be a distribution over $\delta$.
for $r = 1...R$ do
    1. Draw $\delta_{rj} \sim p_j(\delta), \forall j \in [d]$. $u_{rj} \in [0, \delta_{rj}], \forall j \in [d]$.
    2. Compute feature $\boldsymbol{z}_r(\boldsymbol{x}_n)$ as the the indicator vector of bin index $(\lfloor \frac{x_{n1} - u_1}{\delta_1} \rfloor, ..., \lfloor \frac{x_{nd} - u_d}{\delta_d} \rfloor)$, for $\forall n \in [N]$.
end for.
Return $\boldsymbol{z}(\boldsymbol{x}_n) = \frac{1}{\sqrt{D}}[\boldsymbol{z}_1(\boldsymbol{x}_n); ...; \boldsymbol{z}_D(\boldsymbol{x}_n)]$ $\forall n \in [N]$ as the data with RB Features.
```

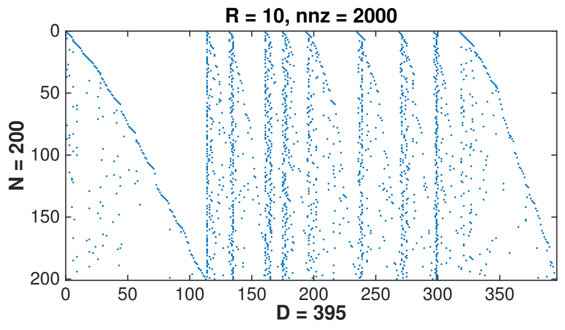


Figure 2: Example of the sparse feature matrix $Z_{N \times D}$ generated by RB. In this special case, Z has the number of rows $N = 200$ and columns $D = 395$, respectively. The number of grids $R = 10$ and the $nnz(Z) = 2000$. Note that for ith row of Z, $nnz(Z(i,:)) = R$ and $R \leq D \leq NR$.

3.1 Sparse Feature Matrix & Iterative Solvers

The first special feature of RB compared to other low-rank approximations is probably the fact that the feature matrix generated by RB could be a large, sparse binary matrix $Z \in \mathbb{R}^{N \times D}$, where the value of D is determined by both number of grids R and kernel parameter (ex. σ in the case of Laplacian Kernel). Different from other random features, D, rather than R, is the actual number of columns of Z. A direct connection between D and R is that the matrix has each row i satisfying $nnz(Z(i,:)) = R$ and therefore $R \leq D \leq NR$. Intuitively speaking, RB has more expressive power than RF since it generates a large yet sparse feature matrix to rapidly transform the data space to a very high dimension space, where data could become linearly separable by classifiers. Fig. 2 gives an example to illustrate the sparse structure of Z.

In the case of *Kernel Ridge Regression* (L2-regularization with square loss), if using RB feature to approximate the RKHS, one can solve (1) directly in its primal form. The weighting vector is simply the solution of the linear system:

$$(Z^T Z + \lambda I) \boldsymbol{w}_{RB} = Z^T y. \tag{8}$$

Note since Z is a large sparse matrix, there is no need to explicitly compute the covariance matrix $Z^T Z$, which is much denser than Z itself. One can apply state-of-the-art sparse iterative solvers such as Conjugate Gradient (CG) and GMRES to directly operate on Z [23]. The main computation in CG or GMRES is the sparse matrix-vector products. Let k be the number of iterations, then the total computational complexity of iterative solver is $O(k\, nnz(Z))$, which is also $O(kNR)$. In addition, since most of elements in Z are zeros, the Compressed Sparse Row type matrix storage format should be employed for economically storing Z [9]. Therefore, one can easily derive that the computational costs and memory requirements in training are still $O(kNR)$ and $O(NR)$ respectively, which are similar as other low-rank approximations [19, 29]. In testing, each testing point produces a new sparse feature vector $z(x) \in \mathcal{R}^D$ based on the hash-table of partitions in training. Note that, $z(x)$ is a sparse vector $nnz(z(x))) = R$. Then we can compute the decision function by $z(x)^T \boldsymbol{w}_{RB}$. Therefore, the computational costs in testing are still $O(dR + R)$. In our experiments, we leverage kernel ridge regression to train both regressors and classifiers since they are very efficient.

When the ERM is smooth but not quadratic, a *Newton-CG* method that solves smooth problem via a series of local quadratic approximation gives the same complexity per CG iteration [13], and note that most of state-of-the-art linear classification algorithms have complexity linear to $nnz(Z)$, the number of nonzeros in design matrix [6]. As an example, in section 4, we introduce a RCD algorithm that has cost per iteration linear to $nnz(Z)$ for L1-regularized ERM with RB features.

3.2 Random Binning Feature as Block Coordinate Descent

In [31], a new approach of analysis was proposed, which interpreted RF as a RCD method in the infinite dimensional space, and gives a better $O(1/R)$ rate in the convergence of objective function. In this section, we extend the approach of [31] to show that, RB Feature can be interpreted as RBCD in the infinite-dimensional space, which by drawing *a block of features* at a time, produces a number of features D significantly more than the number of blocks R, resulting a provably faster convergence rate than other RF. While at the same time, by exploiting state-of-the-art iterative solvers introduced in section 3.1, the computational complexity of RB does not increase with number of features D but only the number of blocks R. Consequently, to achieve the same testing accuracy, RB requires significantly less training and prediction time compared to other RF.

A key quantity to our analysis is an upper bound on the *collision probability* $\nu_{\boldsymbol{\delta}}$ which specifies how likely data points will fall into the same bin, and its inverse $\kappa_{\boldsymbol{\delta}} := 1/\nu_{\boldsymbol{\delta}}$ which is a lower bound estimation of the number of bins of at least one data point. We define them as follows.

Definition 1. *Let the collision probability of data $\mathcal{D}$ on bin $b \in B_{\boldsymbol{\delta}}$:*

$$\nu_b := \frac{|\{n \in [N] \mid \phi_b = 1\}|}{N}. \tag{9}$$

Let $\nu_{\boldsymbol{\delta}} := \max_{b \in B_{\boldsymbol{\delta}}} \nu_b$ be an upper bound on (9), and $\kappa_{\boldsymbol{\delta}} := 1/\nu_{\boldsymbol{\delta}}$ be a lower bound on the number of bins of non-zero data point.

$$\kappa := E_{\boldsymbol{\delta}}[\kappa_{\boldsymbol{\delta}}] = E_{\boldsymbol{\delta}}[1/\nu_{\boldsymbol{\delta}}] \tag{10}$$

is denoted as the lower bound on the expected of number of (used) bins w.r.t the distribution $p(\boldsymbol{\delta})$.

In the RB matrix, the expected collision probability is simply the average number of non-zeros per column, divided by N, a number much smaller than 1 as in the example of Fig. 2. Our analysis assumes a smooth loss function satisfying the following criteria.

Assumption 1. *The loss function $L(z, y)$ is smooth w.r.t. response z so difference between function difference and its linear approximation can be bounded as*

$$L(z_2, .) - L(z_1, .) \leq \nabla L(z_1, .)(z_2 - z_1) + \frac{\beta}{2}(z_2 - z_2)^2.$$

for some constant $0 \leq \beta \leq \infty$.

This assumption is satisfied for a wide range of loss such as square loss ($\beta = 1$), logistic loss ($\beta = 1/4$) or L2-hinge loss ($\beta = 1$).

We interpret RB as a *Fully Corrective Randomized Block Coordinate Descent (FC-RBCD)* on the objective function

$$\min_{\bar{\boldsymbol{w}}} \quad F(\bar{\boldsymbol{w}}) := \mathcal{R}(\bar{\boldsymbol{w}}) + Loss(\bar{\boldsymbol{w}}; \boldsymbol{\phi}) \tag{11}$$

where $Loss(\bar{\boldsymbol{w}}; \phi) = \frac{1}{N}\sum_{n=1}^{N} L(\langle \bar{\boldsymbol{w}}, \boldsymbol{\phi}(\boldsymbol{x}_n)\rangle, y_n)$ and

$$\bar{\boldsymbol{\phi}} := \sqrt{p} \circ \boldsymbol{\phi} = (\sqrt{p(\boldsymbol{\delta})}\boldsymbol{\phi}_{B_{\boldsymbol{\delta}}}(.))_{\boldsymbol{\delta} \in H}$$

with "$\circ$" denoting the component-wise product. The goal is by performing R steps of FC-RBCD on (11), one can obtain a model $\bar{\boldsymbol{w}}^R$ with comparable regularized loss to that from optimal solution of (1). Note one advantage of analysis from this optimization perspective is: it does not rely on Representer theorem, and thus $R(\boldsymbol{w})$ can be L2 regularizer $\frac{\lambda}{2}\|\boldsymbol{w}\|^2$ or L1 regularizer $\lambda\|\boldsymbol{w}\|_1$, where the latter has advantage of giving sparse predictor of faster prediction [31]. The FC-RBCD algorithm maintains an active set of blocks $\mathcal{A}^{(r)}$. At each iteration r, the FC-RBCD does the following:

1. Draw $\boldsymbol{\delta}$ from $p(\boldsymbol{\delta})$ (derived from the kernel $k(.,.)$).
2. Expand active set $\mathcal{A}^{(r+1)} := \mathcal{A}^{(r)} \cup B_{\boldsymbol{\delta}}$.
3. Minimize (11) subject to a limited support $supp(\bar{\boldsymbol{w}}) \subseteq \mathcal{A}^{(r+1)}$.

Note this algorithm is only used for analysis. In practice, one can draw $r = 1...R$ blocks of features at a time, and solve (11) by any optimization algorithm such as those mentioned in section 3.1 or the CD method we will introduce in section 4.

Due to space limit, here we prove the more difficult case when $\mathcal{R}(.)$ is the non-smooth L1 regularizer $\lambda\|\bar{\boldsymbol{w}}\|_1$. The smooth case for $\mathcal{R}(\boldsymbol{w}) = \frac{\lambda}{2}\|\bar{\boldsymbol{w}}\|^2$ can be shown in a similar but simpler way. Note the objective function (11) can be written as

$$\bar{F}(\boldsymbol{w}) := F(\sqrt{\boldsymbol{p}} \circ \boldsymbol{w}) = \mathcal{R}(\sqrt{\boldsymbol{p}} \circ \boldsymbol{w}) + Loss(\boldsymbol{w}, \bar{\boldsymbol{\phi}}). \tag{12}$$

by a scaling of variable $\bar{\boldsymbol{w}} = \sqrt{p} \circ \boldsymbol{w}$.

The below theorem states that, running FC-RBCD for R iterations, it generates a solution $\bar{\boldsymbol{w}}^R$ close to any reference solution $\boldsymbol{w}^*$ in terms of objective (12) with their difference bounded by $O(\frac{1}{\kappa R})$.

Theorem 1. *Let R be the number of blocks (grids) generated by FC-RBCD, and $\boldsymbol{w}^*$ be any reference solution, we have*

$$E[\bar{F}(\boldsymbol{w}^{(R)})] - \bar{F}(\boldsymbol{w}^*) \leq \frac{\beta\|\boldsymbol{w}^*\|^2}{\kappa R'} \tag{13}$$

for $R' := R - c > 0$, where $c = \lceil \frac{2\kappa(\bar{F}(\mathbf{0}) - \bar{F}(\boldsymbol{w}^*))}{\beta\|\boldsymbol{w}^*\|^2} \rceil$.

PROOF. Firstly, we obtain an expression for the progress made by each iteration of FC-RBCD. Let $B := B_{\boldsymbol{\delta}^{(r)}}$ be the block drawn at step 1 of FC-RBCD, and $\bar{\boldsymbol{w}}^{(r+1)}$ be the minimizer of (11) subject to support $supp(\bar{\boldsymbol{w}}) \subseteq \mathcal{A}^{(r+1)}$ given by the step 3. Since $B \subseteq \mathcal{A}^{(r+1)}$, we have

$$F(\bar{\boldsymbol{w}}^{(r+1)}) - F(\bar{\boldsymbol{w}}^{(r)}) \leq F(\bar{\boldsymbol{w}}^{(r)} + \boldsymbol{\eta}_B) - F(\bar{\boldsymbol{w}}^{(r)}) \tag{14}$$

for any $\boldsymbol{\eta}_B : supp(\boldsymbol{\eta}) \subseteq B$. Then denote b_i as the bin $\boldsymbol{x}_i$ falling in and $L'_i = \nabla L(\bar{\boldsymbol{w}}^{(r)T}\boldsymbol{\phi}(\boldsymbol{x}_i), y_i)$, by smoothness of the loss (Assumption 1), we have

$$Loss(\bar{\boldsymbol{w}}^{(r)} + \boldsymbol{\eta}_B) - Loss(\bar{\boldsymbol{w}}^{(r)}) \leq \frac{1}{N}\sum_{i=1}^{N} L'_i \phi_{b_i}\eta + \frac{\beta}{2}(\eta_{b_i}\phi_{b_i})^2$$
$$\leq \langle \boldsymbol{g}_B, \boldsymbol{\eta}_B \rangle + \frac{\beta \nu_{\boldsymbol{\delta}^{(r)}}}{2}\|\boldsymbol{\eta}_B\|^2 \tag{15}$$

where the second inequality uses the fact $\phi_{b_i} = 1$ and

$$\boldsymbol{g}_B := \nabla_B Loss(\bar{\boldsymbol{w}}^{(r)}, \boldsymbol{\phi}).$$

Now consider the regularization term, note since block B is drawn from an inifinite-dimensional space, the probability that B is in active set is 0. Therefore, we have $B \cap \mathcal{A}^{(r)} = \emptyset$, $\bar{\boldsymbol{w}}_B^{(r)} = \boldsymbol{0}$ and $\mathcal{R}_B(\bar{\boldsymbol{w}}_B^{(r)}) = 0$. As a result,

$$\begin{aligned} &F(\bar{\boldsymbol{w}}^{(r)} + \boldsymbol{\eta}_B) - F(\bar{\boldsymbol{w}}^{(r)}) \\ &\leq \mathcal{R}_B(\boldsymbol{\eta}_B) + \langle \boldsymbol{g}_B, \boldsymbol{\eta}_B\rangle + \frac{\beta \nu_{\boldsymbol{\delta}^{(r)}}}{2}\|\boldsymbol{\eta}_B\|^2 \end{aligned} \tag{16}$$

Let $\boldsymbol{\eta}_B$ be the minimizer of RHS of (16). It satisfies $\boldsymbol{\rho}_B + \boldsymbol{g}_B + \beta v_{\boldsymbol{\delta}^{(r)}}\boldsymbol{\eta}_B = \boldsymbol{0}$ for some $\boldsymbol{\rho}_B \in \partial\mathcal{R}(\boldsymbol{\eta}_B)$, and thus,

$$\begin{aligned} &F(\bar{\boldsymbol{w}}^{(r)} + \boldsymbol{\eta}_B) - F(\bar{\boldsymbol{w}}^{(r)}) \\ &\leq \langle \boldsymbol{\rho}_B, \boldsymbol{\eta}_B\rangle + \langle \boldsymbol{g}_B, \boldsymbol{\eta}_B\rangle + \frac{\beta \nu_{\boldsymbol{\delta}^{(r)}}}{2}\|\boldsymbol{\eta}_B\|^2 \\ &= -\frac{1}{2\beta\nu_{\boldsymbol{\delta}^{(r)}}}\|\boldsymbol{\rho}_B + \boldsymbol{g}_B\|^2 \end{aligned} \tag{17}$$

Now taking expectation w.r.t. $p(\boldsymbol{\delta})$ on both sides of (17), we have

$$\begin{aligned} E[F(\bar{\boldsymbol{w}}^{(r)} + \boldsymbol{\eta}_B)] - F(\bar{\boldsymbol{w}}^{(r)}) &\leq -\frac{1}{2\beta}E\left[\frac{1}{\nu_{\boldsymbol{\delta}^{(r)}}}\|\boldsymbol{\rho}_B + \boldsymbol{g}_B\|^2\right] \\ &\leq -\frac{1}{2\beta}E\left[\frac{1}{\nu_{\boldsymbol{\delta}^{(r)}}}\right]E\left[\|\boldsymbol{\rho}_B + \boldsymbol{g}_B\|^2\right] \\ &\leq -\frac{\kappa}{2\beta}\|\bar{\boldsymbol{\rho}} + \bar{\boldsymbol{g}}\|^2 \end{aligned} \tag{18}$$

where $\bar{\boldsymbol{\rho}} := \sqrt{\boldsymbol{p}} \circ \boldsymbol{\rho}$, $\bar{\boldsymbol{g}} := \sqrt{\boldsymbol{p}} \circ \boldsymbol{g}$, and the second inequality uses the fact that the number of used bins $\kappa_{\boldsymbol{\delta}^{(r)}} = 1/\nu_{\boldsymbol{\delta}^{(r)}}$ has non-negative correlation with the discriminative power of block B measured by the magnitude of gradient with soft-thresholding $\|\bar{\boldsymbol{\rho}}_B + \bar{\boldsymbol{g}}_B\|$ (i.e. less collisions on grid B implies B to be a better block of features).

The result of (18) expresses descent amount in terms of the proximal gradient of the reparameterized objective (12). Note for $B : B \cap \mathcal{A}^{(r)} = \emptyset$, we have $\boldsymbol{w}_B^{(r)} = \boldsymbol{0}$, and $\mathcal{R}_B(\bar{\boldsymbol{\eta}}) - \mathcal{R}_B(\boldsymbol{0}) = \langle \bar{\boldsymbol{\rho}}, \bar{\boldsymbol{\eta}}\rangle$; on the other hand, for $B \subseteq \mathcal{A}^{(r)}$, we have

$$\boldsymbol{0} \in arg\min_{\bar{\boldsymbol{\eta}}_B} \; \mathcal{R}_B(\sqrt{p_B}\boldsymbol{w}_B + \bar{\boldsymbol{\eta}}_B) + \langle \bar{\boldsymbol{g}}_B, \sqrt{p_B}\boldsymbol{w}_B + \bar{\boldsymbol{\eta}}_B\rangle$$

since they are solved to optimality in the previous iteration. Then

$$\begin{aligned} &E[F(\bar{\boldsymbol{w}}^{(r)} + \boldsymbol{\eta})] - F(\bar{\boldsymbol{w}}^{(r)}) \\ &\leq -\frac{\kappa}{2\beta}\|\bar{\boldsymbol{\rho}} + \bar{\boldsymbol{g}}\|^2 = \langle \bar{\boldsymbol{\rho}}, \bar{\boldsymbol{\eta}}\rangle + \langle \bar{\boldsymbol{g}}, \bar{\boldsymbol{\eta}}\rangle + \frac{\beta}{2\kappa}\|\bar{\boldsymbol{\eta}}_{\bar{\mathcal{A}}^{(r)}}\|^2 \\ &= \mathcal{R}(\sqrt{\boldsymbol{p}} \circ (\boldsymbol{w}^{(r)} + \bar{\boldsymbol{\eta}})) - \mathcal{R}(\sqrt{\boldsymbol{p}} \circ \boldsymbol{w}^{(r)}) + \langle \bar{\boldsymbol{g}}, \bar{\boldsymbol{\eta}}\rangle + \frac{\beta}{2\kappa}\|\bar{\boldsymbol{\eta}}_{\bar{\mathcal{A}}^{(r)}}\|^2 \end{aligned} \tag{19}$$

where $\bar{\boldsymbol{\eta}}_{\bar{\mathcal{A}}^{(r)}} := (\bar{\boldsymbol{\eta}}_B)_{B : B \cap \mathcal{A}^{(r)} = \emptyset}$ and $\bar{\boldsymbol{\eta}} := \sqrt{\boldsymbol{p}} \circ \boldsymbol{\eta}$. Thus the final step is to show the descent amount given by RHS of (19) decreases the suboptimality $\bar{F}(\boldsymbol{w}^{(r)}) - \bar{F}(\boldsymbol{w}^*)$ significantly. This can be achieved by considering $\bar{\boldsymbol{\eta}}$ of the form $\alpha(\boldsymbol{w}^* - \boldsymbol{w}^{(r)})$ for some $\alpha \in [0, 1]$ as follows:

$$\begin{aligned} &E[\bar{F}(\boldsymbol{w}^{(r)} + \bar{\boldsymbol{\eta}})] - \bar{F}(\boldsymbol{w}^{(r)}) \\ &\leq \min_{\bar{\boldsymbol{\eta}}} \mathcal{R}(\sqrt{\boldsymbol{p}} \circ (\boldsymbol{w}^{(r)} + \bar{\boldsymbol{\eta}})) - \mathcal{R}(\sqrt{\boldsymbol{p}} \circ \boldsymbol{w}^{(r)}) + \langle \bar{\boldsymbol{g}}, \bar{\boldsymbol{\eta}}\rangle + \frac{\beta}{2\kappa}\|\bar{\boldsymbol{\eta}}_{\bar{\mathcal{A}}^{(r)}}\|^2 \\ &\leq \min_{\bar{\boldsymbol{\eta}}} \bar{F}(\boldsymbol{w}^{(r)} + \bar{\boldsymbol{\eta}}) - \bar{F}(\boldsymbol{w}^{(r)}) + \frac{\beta}{2\kappa}\|\bar{\boldsymbol{\eta}}_{\bar{\mathcal{A}}^{(r)}}\|^2 \\ &\leq \min_{\alpha \in [0,1]} \bar{F}((1-\alpha)\boldsymbol{w}^{(r)} + \alpha\boldsymbol{w}^*) - \bar{F}(\boldsymbol{w}^{(r)}) + \frac{\beta\alpha^2}{2\kappa}\|\boldsymbol{w}^*\|^2 \\ &\leq \min_{\alpha \in [0,1]} -\alpha(\bar{F}(\boldsymbol{w}^{(r)}) - \bar{F}(\boldsymbol{w}^*)) + \frac{\beta\alpha^2}{2\kappa}\|\boldsymbol{w}^*\|^2, \end{aligned} \tag{20}$$

where the second and fourth inequalities are from convexity of $\bar{F}(.)$. The α minimizing (20) is $\alpha^* := \min(\frac{\kappa(\bar{F}(\boldsymbol{w}^{(r)}) - \bar{F}(\boldsymbol{w}^*))}{\beta\|\boldsymbol{w}^*\|^2}, 1)$, which leads to

$$E[\bar{F}(\boldsymbol{w}^{(r)} + \bar{\boldsymbol{\eta}})] - \bar{F}(\boldsymbol{w}^{(r)}) \leq -\frac{\kappa(\bar{F}(\boldsymbol{w}^{(r)}) - \bar{F}(\boldsymbol{w}^*))^2}{2\beta\|\boldsymbol{w}^*\|^2} \tag{21}$$

if $\bar{F}(\boldsymbol{w}^{(r)}) - \bar{F}(\boldsymbol{w}^*) \leq \frac{\beta}{\kappa}\|\boldsymbol{w}^*\|^2$; otherwise, we have $E[\bar{F}(\boldsymbol{w}^{(r)} + \bar{\boldsymbol{\eta}})] - \bar{F}(\boldsymbol{w}^{(r)}) \leq -\frac{\beta}{2\kappa}\|\boldsymbol{w}^*\|^2$. Note the latter case cannot happen more than $c = \lceil \frac{2\kappa(\bar{F}(\boldsymbol{0}) - \bar{F}(\boldsymbol{w}^*))}{\beta\|\boldsymbol{w}^*\|^2} \rceil$ times since FC-RBCD is a descent method. Therefore, for $r' := r - c > 0$, solving the recursion (21) leads to the conclusion. $\square$

Note we have $\|\sqrt{\boldsymbol{p}} \circ \boldsymbol{w}^*\|_1 \leq \|\sqrt{\boldsymbol{p}}\|\|\boldsymbol{w}^*\| = \|\boldsymbol{w}^*\|$ in the L1-regularized case, and thus the FC-RBCD guarantees convergence of the L1-norm objective to the (non-square) L2-norm objective. The convergence result of Theorem 1 is of the same form to the rate proved in [31] for other random features, however, with an additional multiplicative factor $\kappa \geq 1$ that speeds up the rate by κ times. Recall that κ is the lower bound on the expected number of bins being used by data samples for each block of features $B_{\boldsymbol{\delta}}$, which in practice is a factor much larger than 1, as shown in the Figure 2 and also in our experiments. In particular, in case each grid $B_{\boldsymbol{\delta}}$ has similar number of bins being used, we have $D \approx \kappa R$, and thus obtain a rate of the form

$$E[\bar{F}(\boldsymbol{w}^{(R)})] - \bar{F}(\boldsymbol{w}^*) \lesssim \frac{\beta\|\boldsymbol{w}^*\|^2}{D}. \tag{22}$$

Note for a fixed R, the total number of features D is increasing with kernel parameter $1/\sigma$ in the case of Laplacian Kernel, which means the less smooth the kernel, the faster convergence of RB. A simple extreme case is when $\sigma \to 0$, where one achieves 0 training loss, and the RB, by putting each sample in a separate bin, converges to 0 loss with $R = 1$, $D = N$. On the other hand, other random features, such as Fourier, still require large R for convergence to 0 loss. In practice, there are many data that require a small kernel bandwidth σ to avoid underfitting, for which RB has dramatically faster convergence than other RF.

4. STRONG PARALLELIZABILITY OF RANDOM BINNING

In this section, we study another strength of RB Features in the context of *Sparse Random Feature* [31], where one aims to train a sparse nonlinear predictor that has faster prediction and more compact representation through an L1-regularized objective. In this case, the CD method is known as state-of-the-art solver [32, 21], and we aim to show that the structure of RB allows CD to be parallelized with much more speedup than that of other random features.

Algorithm 2 Sparse Random Binning Features Algorithm based on RCD Method

```
0. Generate RB feature matrix $Z$ by Algorithm 1
1. $\boldsymbol{z}^1 = \mathbf{0}, \boldsymbol{w}^1 = \mathbf{0}$.
for t=1...T do
    2. Draw $j$ from $[D]$ uniformly at random.
    3. Compute $d_j^*$ by (26).
    4. $\boldsymbol{w}^{t+1} := \boldsymbol{w}^t + d_j^* \boldsymbol{e}_j$.
    5. Maintain $\hat{y}_i, \forall i \in [N]$ to satisfy (27).
end for
```

4.1 Coordinate Descent for Sparse Random Binning

Given the $N \times D$ data matrix produced by the RB Algorithm 1, a RCD Method solves

$$\min_{\boldsymbol{w} \in \mathbb{R}^D} \lambda \|\boldsymbol{w}\|_1 + \frac{1}{N} \sum_{n=1}^{N} L(\boldsymbol{w}^T \boldsymbol{z}_i, y_i) \tag{23}$$

by minimizing (23) w.r.t. a single coordinate j

$$\min_{d_j} \lambda |w_j + d_j| + g_j d_j + \frac{M_j}{2} d_j^2 \tag{24}$$

at a time, where

$$g_j := \frac{1}{N} \sum_{n=1}^{N} (\nabla_j L(\boldsymbol{w}^T \boldsymbol{z}_i, y_i)) z_{ij} \tag{25}$$

is the gradient of loss term in (23) w.r.t. the j-th coordinate, and $M_j := \beta \frac{1}{N} \sum_{i=1}^{N} z_{ij}^2$ is an upper bound on $\nabla_{jj} L(.)$. By focusing on single coordinate, (24) is a tighter upper bound than other algorithms such as Proximal Gradient Method and allows simple closed-form solution

$$d_j^* := \mathbf{prox}_{R/M_j}(w_j - \frac{g_j}{M_j}) - w_j \tag{26}$$

$$\mathbf{prox}_{R/M_j}(v_j) := \begin{cases} 0, & |v_j| \leq \lambda / M_j \\ v_j - \lambda/M_j, & v_j > \lambda/M_j \\ v_j + \lambda/M_j, & v_j < -\lambda/M_j \end{cases}.$$

To have efficient evaluation of the gradient (25), a practical implementation maintain the responses

$$\hat{y}_i := \boldsymbol{w}^T z_i \tag{27}$$

after each update $\boldsymbol{w}^{t+1} := \boldsymbol{w}^t + d_j^* \boldsymbol{e}_j$, so the cost for each coordinate-wise minimization takes $O(nnz(\boldsymbol{z}_j))$ time for both gradient evaluation and maintenance of (27), where $\boldsymbol{z}_j := (z_{ij})_{i \in [N]}$. The algorithm is summarized in Alg. 2, which just like the iterative solver introduced in section 3.1, has cost $O(nnz(Z))$ for one pass of all variables $j \in [D]$.

4.2 Parallel Randomized Coordinate Descend on Random Binning Features

The RCD, however, is hard to parallelize [21]. It is known that simultaneous updates of two coordinates j_1, j_2 could lead to divergence, and although one can enforce convergence by shortening the step size $\frac{1}{M_p} \ll \frac{1}{M_j}$, the convergence rate will not be improved with parallelization without additional assumption [1, 22].

On the other hand, in [22], it is shown that a function with *partially separable* smooth term plus a separable non-smooth term

$$\min_{\boldsymbol{w} \in \mathbb{R}^D} F(\boldsymbol{w}) := \Omega(\boldsymbol{w}) + \sum_{i=1}^{N} f_i(\boldsymbol{w}) \tag{28}$$

can be parallelized with guaranteed speedup in iteration complexity, where $\Omega(\boldsymbol{w})$ is a non-smooth separable function and each function $f_i(\boldsymbol{w})$ is a smooth depends only on at most ω number of variables. The form (28), fortunately, fits our objective (23) with features $\boldsymbol{z}_i$ generated by RB. In particular, the generating process of RB guarantees that, for each block of feature $B_{\boldsymbol{\delta}}$, the i-th sample can fall in exactly one bin $b = (\lfloor \frac{x_{n1} - u_1}{\delta_1} \rfloor, ..., \lfloor \frac{x_{nd} - u_d}{\delta_d} \rfloor)$, therefore each sample inolves at most R features out of D. Specifically, let $\Omega(\boldsymbol{w}) := \lambda \|\boldsymbol{w}\|_1$ and

$$f_i(\boldsymbol{w}) := \frac{1}{N} L(\boldsymbol{w}^T \boldsymbol{z}_i, y_i),$$

we have $\omega = R$. Then by Theorem 19 of [22], a parallel RCD of τ threads that selects coordinate j uniformly at random achieves a speed-up (i.e. time-of-sequential/time-of-parallel) of

$$\textit{speedup-ratio} = \frac{\tau}{1 + \frac{(R-1)(\tau-1)}{D-1}}. \tag{29}$$

When $D, R \gg 1$, and $\tau = a\bar{\kappa} + 1$ where $\bar{\kappa} := D/R$, (29) becomes

$$\textit{speedup-ratio} = \frac{a\bar{\kappa} + 1}{1 + a}, \tag{30}$$

which approaches $\bar{\kappa}$ when $a \to \infty$. Therefore, it is guaranteed in theory that parallelization can speedup RCD significantly as long as $\bar{\kappa} = D/R \gg 1$. We give our sparse RB Features algorithm based on parallel RCD in Alg. 2. In our experiments, although a perfect speedup of $\bar{\kappa}$ is not achievable, the speedup of parallel RCD on RB features is significantly larger than that for other random features. Note that the speedup achieved in this section is orthogonal to the faster convergence rate achieved in section 3, so by increasing κ, the advantage of RB over other RF is super-linearly increasing in the case of parallel RCD.

5. EXPERIMENTS

In this section, we present extensive sets of experiments to demonstrate the efficiency and effectiveness of RB. The datasets are chosen to overlap with those in other papers in the literature, where the details are shown in the table 1. All sets except census are available at LIBSVM data set [2]. All computations are carried out on a DELL dual socket with Intel Xeon processors at 2.93GHz for a total of 16 cores and 250 GB of memory running the SUSE Linux operating system. We implemented all methods in C++ and all dense matrix operations are performed by using the optimized BLAS and LAPACK routines provided in the OpenBLAS library. Due to the limited space, we only choose subsets of our results to present in each subsection. However, these results are objective and unbiased.

Table 1: Properties of the datasets.

Name	C: Classes	d: Features	N: Train	M: Test
cadata	1	8	16,512	4,128
census	1	119	18,186	2,273
ijcnn1	2	22	35,000	91,701
cod_rna	2	8	49,437	271,617
covtype	2	54	464,809	116,203
SUSY	2	18	4,000,000	1,000,000
mnist	10	780	60,000	10,000
acoustic	3	50	78,823	19,705
letter	26	16	10,500	5,000

5.1 Effects of σ and R on Random Binning

We perform experiments to investigate the characteristics of RB by varying the kernel parameter λ and the rank R, respectively. We use a regularization $\lambda = 0.01$ to make sure the reasonable performance of RB and other low-rank kernels, although we found that RB is not sensitive to this parameter. We increase the σ in the large interval from $1e-2$ to $1e2$ so that the optimal σ locates within the interval. We apply CG iterative solver to operate on Z directly. In order to make fair runtime comparison in each run, we set the $tol = 1e - 15$ to force similar CG iterations with different σ.

We evaluate the training and testing performance of regression and classification, when varying σ with fixed R. In [19], it does not consider the effect of σ in their analysis, which however has a large impact on the performance since D depends on the number of bins which is controlled by σ. Fig. 3 shows that the training and testing performance coincidentally decrease (increase) before they diverge when D grows by increasing σ. This confirms with our analysis in Theorem 1 that the larger κ, the faster convergence of RB Feature (recall that the convergence rate is $O(1/\kappa R)$).

Second, one should not be surprised that the empirical training time increases with D. The operations involving the weighting vector w_{RB} could become as expensive as a sparse matrix-vector operation in an iterative solver. However, the total computational costs are still bounded by $O(NR)$ but the constant factor may vary with different datasets. Fortunately, in most of cases, the training time corresponding to the peak performance is just slightly higher than the smallest one. In practice, there are several ways to improve the computation costs by exploiting more advanced sparse matrix techniques such as preconditioning and efficient storage scheme, which is out scope of this paper and left for future study.

Finally, we evaluate the training and testing performance when varying R with fixed σ. Fig.4 shows that the training and testing performance converge almost linearly with D, which again confirms our analysis in Theorem 1. In addition, we observe that RB has strong overfit ability which turns out to be a strong attribute, especially when the hypothesis space has not yet saturated.

5.2 Performance Comparisons of All Methods

We present a large sets of experiments to compare RB with other most popular low-rank kernel approximations, including RF [19], Nyström [29], and recently proposed independent block approximation [27]. We also compare all methods with the exact kernel as a benchmark [24]. We do not report the results of the vanilla kernel on covtype and SUSY since the programs run out of memory. To make a fair comparison, we also apply CG on RB and Nyström directly on Z to admit similar computational costs. Since the independent block kernel approximation approximates the kernel matrix directly, we employ direct solver of dense matrix for this method. In practice, the CG iterative solver has no need to solve in high precision [3], which has also been observed in our experiments. Thus, we set the tolerance to $1e - 3$.

Fig.5 clearly demonstrates the superiority of RB compared to other low-rank kernels. For example, in the first column, RB significantly outperforms other methods in testing performance on all of these datasets, especially when R is relatively small. This is because RB enjoys much faster convergence rate to the optimal function than other methods. The advantage generally diminishes when R increases to reasonably large. However, for some large datasets such as covtype and SUSY, increasing number of random features or R boosts the performance extremely slow. This is consistent with our analysis that RB enjoys its fast convergence rate of $O(1/\kappa R)$ while other methods has slow convergence rates $O(1/\sqrt{R})$. The third and fourth columns further promote the insights about how many number of random features or how large rank R that is needed for achieving similar performance of RB. In particular, RB is often between one and three orders of magnitude faster and less memory consumptions than other methods.

In the second column, we also observe that the training time of all low-rank kernels are linear with R, which is expected since all these methods has computational complexity of $O(kNR)$. The difference in training time between these low-rank kernels is only within some constant factors. However, we point out that the computations of RF, Nyström and independent block approximation are mainly carried out by the high-optimized BLAS library since they are dense matrices. In contrast, the computations of RB are most involved in sparse matrix operations, which are self-implemented and not yet optimized. In addition, more advanced sparse matrix techniques such as preconditioning can be explored to significantly accelerate the computation, which we leave it as future work.

5.3 Parallel Performance of Random Binning and Random Fourier

We perform experiments to compare RB with RF when using RCD to solve L1-regularized Lasso and kernel SVM for both regression and binary classification problems. Since the goal is to demonstrate the strong parallel performance of RB, we implement the basic parallel implementation of RCD based on simple shared memory parallel programming model with OpenMP. We leave the high-performance distributed RCD implementation as one of the future works. We define the speedup of RCD on multicore implementation as follows:

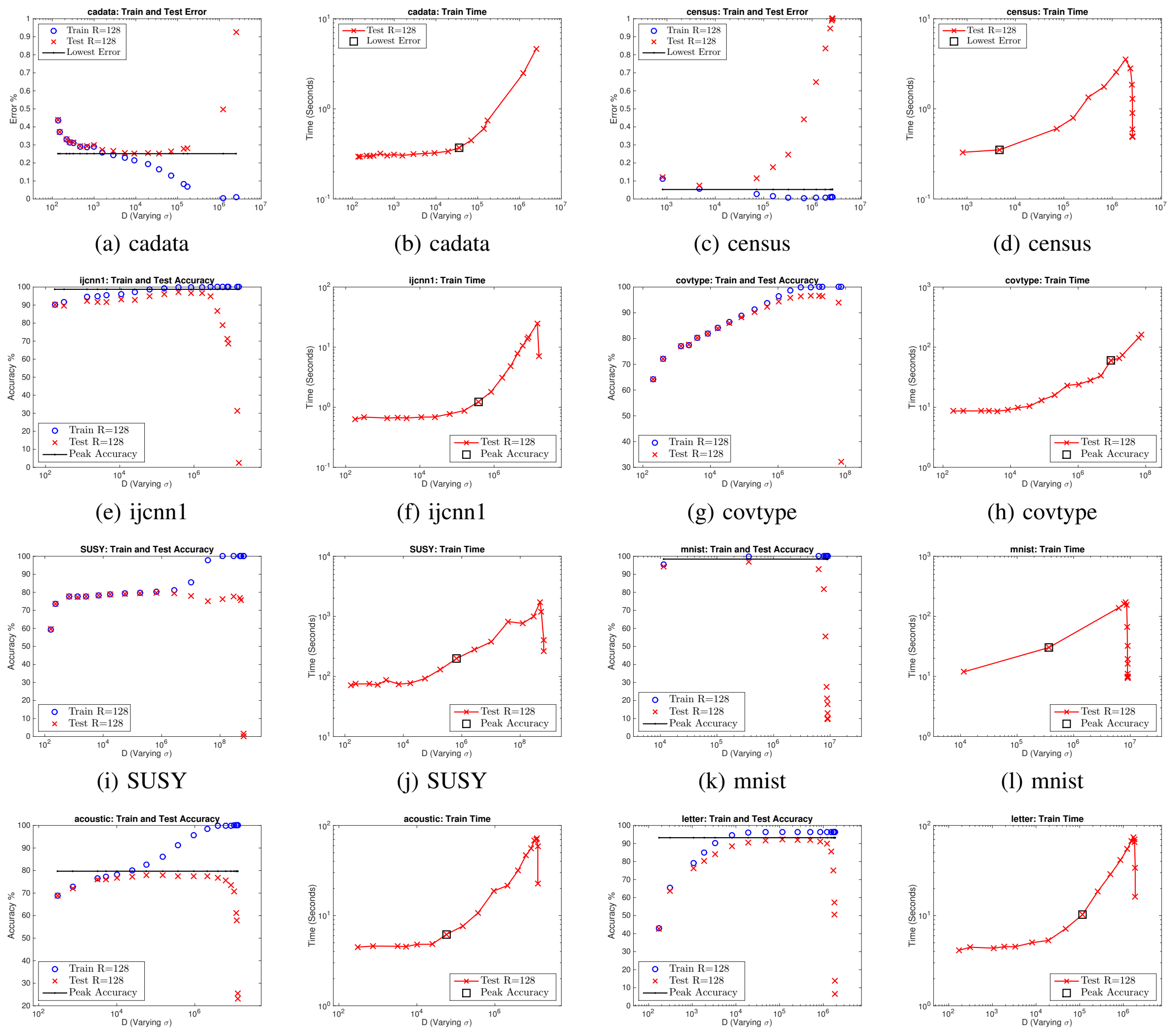
$$speedup = \frac{\textit{runtime of RCD using single core}}{\textit{runtime using } P \textit{ cores}}$$

As shown in Fig.6, when the sparsity level of the feature matrix Z is high, the near-linear speedup can be achieved [17, 15]. This is because the minimization problem can almost be separated along the coordinate axes, then higher degrees of parallelism are possible. In contrast, if Z is lack of sparsity, then the penalty for data correlations slows the speedup to none. This is confirmed by no gain of parallel speedup of RF since Z is always fully dense. Obviously, in order to empower strong parallel performance of RB, a very large D is expected, which interestingly coincides with power of its faster convergence. Therefore, one can enjoy the double benefits of fast convergence and strong parallelizability of RB, which is especially useful for very large-scale problems.

6. CONCLUSIONS

In this paper, we revisit RB features, an overlooked yet very powerful random features, which we observe often to be orders of magnitude faster than other random features and kernel approximation methods to achieve the same accuracy. Motivated by these impressive empirical results, we propose the first analysis of RB from the perspective of optimization, to make a solid attempt to quantify its faster convergence, which is not captured by traditional Monte-Carlo analysis. By interpreting RB as a RBCD in the infinite-dimensional space, we show that by drawing R grids with at least κ expected number of non-empty bins per grid, RB achieves a convergence rate of $O(1/\kappa R)$. In addition, in the L1-regularized setting, we demonstrate the sparse structure of RB features allows RCD solver to be parallelized with guaranteed speedup proportional to κ. Our extensive experiments demonstrate the superior performance of the RB features over other random feature and kernel approximation methods.

7. ACKNOWLEDGEMENT



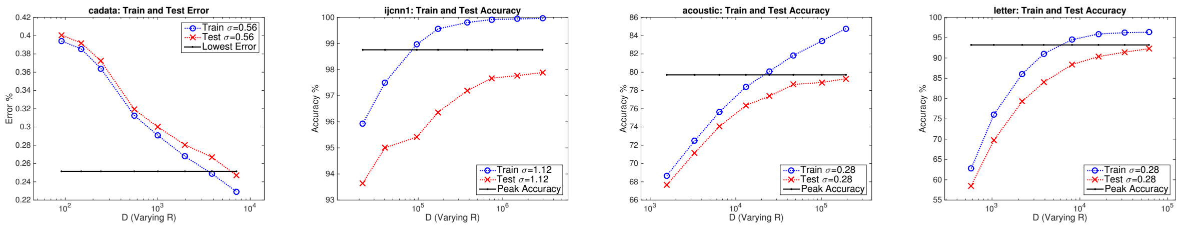
(a) cadata (b) cadata (c) census (d) census

(e) ijcnn1 (f) ijcnn1 (g) covtype (h) covtype

(i) SUSY (j) SUSY (k) mnist (l) mnist

(m) acoustic (n) acoustic (o) letter (p) letter

Figure 3: Train and test performance, and train time when varying σ with fixed R. The black line and square box represent the best test performance of the exact kernel and RB respectively.



(a) cadata (b) ijcnn1 (c) acoustic (d) letter

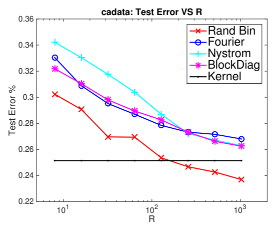
Figure 4: Train and test performance when varying R with fixed σ.

This work was done while L. Wu was a research intern at IBM Research. J. Chen is supported in part by the XDATA program of the Advanced Research Projects Agency (DARPA), administered through Air Force Research Laboratory contract FA8750-12-C-0323.

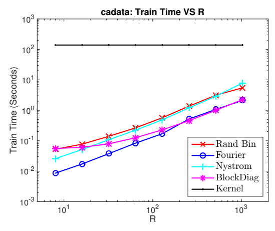
References

[1] J. K. Bradley, A. Kyrola, D. Bickson, and C. Guestrin. Parallel coordinate descent for l1-regularized loss minimization. *CoRR, abs/1105.5379*, 2011.

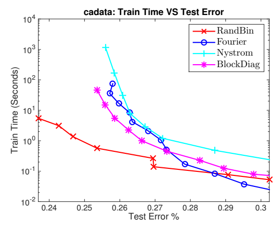
[2] C. Chang and C. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Tech-*



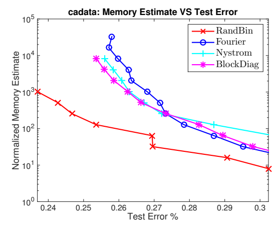
(a) cadata



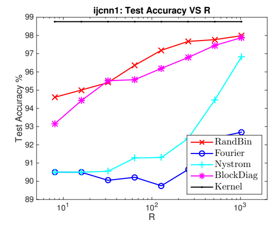
(b) cadata



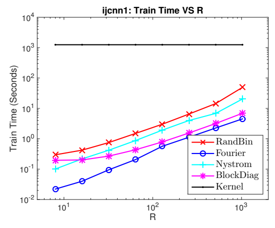
(c) cadata



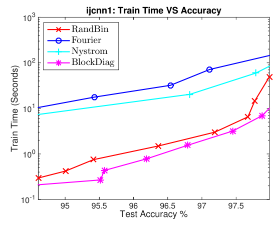
(d) cadata



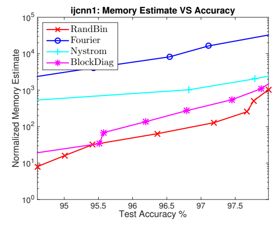
(e) ijcnn1



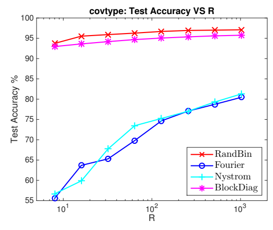
(f) ijcnn1



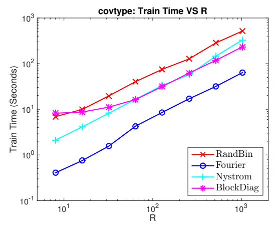
(g) ijcnn1



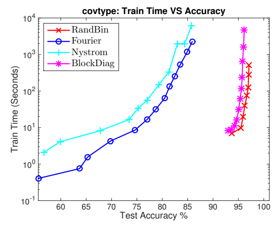
(h) ijcnn1



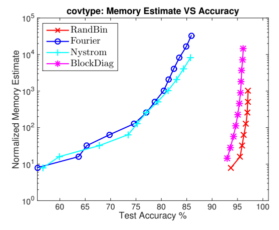
(i) covtype



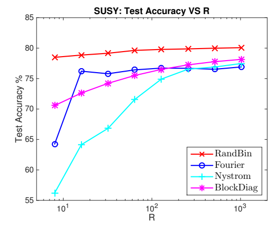
(j) covtype



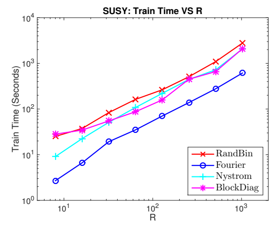
(k) covtype



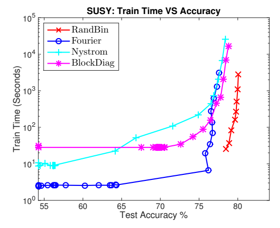
(l) covtype



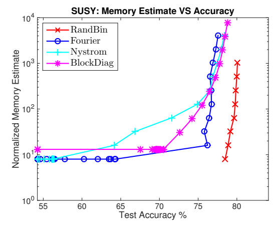
(m) SUSY



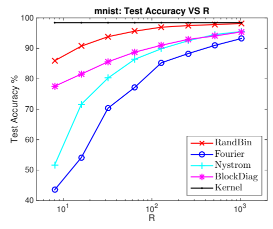
(n) SUSY



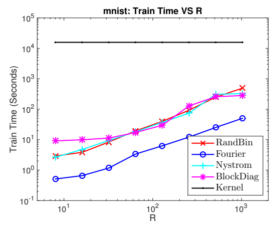
(o) SUSY



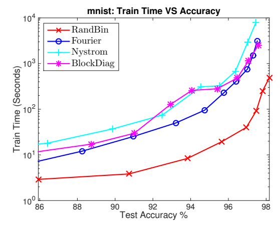
(p) SUSY



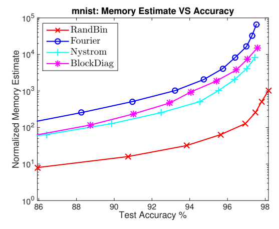
(q) mnist



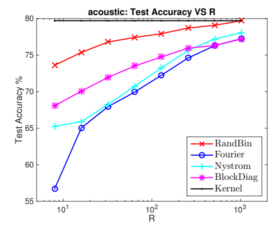
(r) mnist



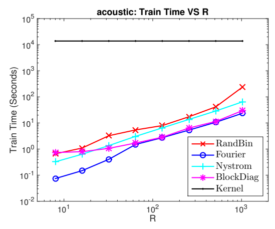
(s) mnist



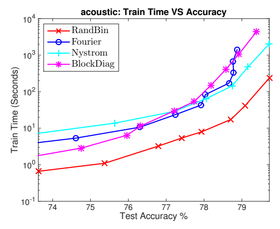
(t) mnist



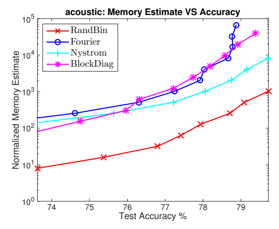
(u) acoustic



(v) acoustic

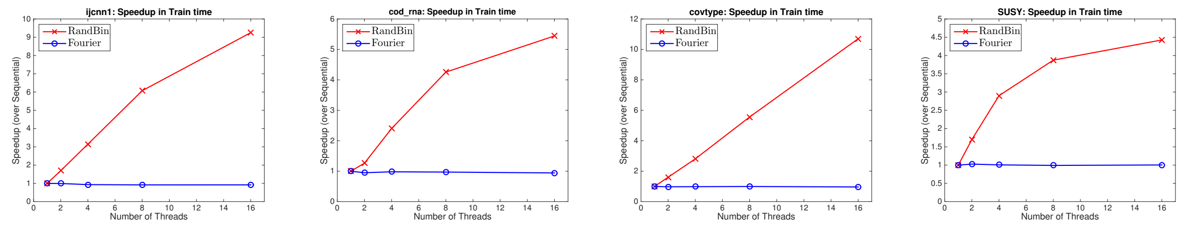


(w) acoustic



(x) acoustic

Figure 5: Comparisons among RB, RF, Nyström and Independent Block approximation. The first and second columns plot test performance and train time when increasing R. The third and fourth columns plot the train time and memory consumptions when achieving the desired test performance.



(a) ijcnn1 (b) cod_rna (c) covtype (d) SUSY

Figure 6: Comparisons of parallel performance between RB and RF using RCD when increasing the number of threads.

nology, 2:27:1–27:27, 2011.

[3] J. Chen, L. Wu, K. Audhkhasi, B. Kingsbury, and B. Ramabhadran. Efficient one-vs-one kernel ridge regression for speech recognition. In *ICASSP*, 2016.

[4] B. Dai, B. Xie, N. He, Y. Liang, A. Raj, M.-F. F. Balcan, and L. Song. Scalable kernel methods via doubly stochastic gradients. In *NIPS*. Curran Associates, Inc., 2014.

[5] P. Drineas and M. W. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *JMLR*, 6:2153–2175, Dec. 2005.

[6] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

[7] S. Fine and K. Scheinberg. Efficient svm training using low-rank kernel representations. *JMLR*, 2:243–264, Mar. 2002.

[8] A. Gittens and M. W. Mahoney. Revisiting the nyström method for improved large-scale machine learning. *CoRR*, abs/1303.1849, 2013.

[9] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[10] P.-S. Huang, H. Avron, T. Sainath, V. Sindhwani, and B. Ramabhadran. Kernel methods match deep neural networks on timit. In *ICASSP*, 2014.

[11] S. Kumar, M. Mohri, and A. Talwalkar. Sampling methods for the nyström method. *JMLR*, 13:981–1006, Apr. 2012.

[12] Q. V. Le, T. Sarlós, and A. J. Smola. Fastfood: Approximate kernel expansions in loglinear time. *ICML*, 2013.

[13] C.-J. Lin, R. C. Weng, and S. S. Keerthi. Trust region newton method for logistic regression. *JMLR*, 9:627–650, 2008.

[14] H.-T. Lin and L. Li. Support vector machinery for infinite ensemble learning. *JMLR*, 9:285–312, 2008.

[15] J. Liu, S. J. Wright, C. Ré, and V. Bittorf. An asynchronous parallel stochastic coordinate descent algorithm. *JMLR*, 32(1):469–477, 2014.

[16] Z. Lu, A. May, K. Liu, A. B. Garakani, D. Guo, A. Bellet, L. Fan, M. Collins, B. Kingsbury, M. Picheny, and F. Sha. How to scale up kernel methods to be as good as deep neural nets. *CoRR*, abs/1411.4000, 2014.

[17] J. Mareček, P. Richtárik, and M. Takáč. Distributed block coordinate descent for minimizing partially separable functions. *Numerical Analysis and Optimization*, 134:261–288, 2014.

[18] J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Royal Society London*, A 209:415–446, 1909.

[19] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*. Curran Associates, Inc., 2007.

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

[21] P. Richtárik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014.

[22] P. Richtárik and M. Takáč. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, pages 1–52, 2015.

[23] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003.

[24] B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[25] S. Si, C. Hsieh, and I. S. Dhillon. Memory efficient kernel approximation. In *ICML*, 2014.

[26] A. J. Smola and B. Schökopf. Sparse greedy matrix approximation for machine learning. In *ICML*, ICML '00, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[27] M. L. Stein. Limitations on low rank approximations for covariance matrices of spatial data. *Spatial Statistics*, 8:1 – 19, 2014. Spatial Statistics Miami.

[28] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *NIPS*. MIT Press, 2004.

[29] C. K. I. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In *NIPS*. MIT Press, 2001.

[30] Z. Yang, A. G. Wilson, A. J. Smola, and L. Song. A la carte - learning fast kernels. In *AISTATS*, 2015.

[31] I. E.-H. Yen, T.-W. Lin, S.-D. Lin, P. K. Ravikumar, and I. S. Dhillon. Sparse random feature algorithm as coordinate descent in hilbert space. In *JMLR*, 2014.

[32] G.-X. Yuan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin. A comparison of optimization methods and software for large-scale l1-regularized linear classification. *JMLR*, 11:3183–3234, 2010.