RANDOM PROJECTIONS FOR SUPPORT VECTOR MACHINES

By

Saurabh Paul

A Thesis Submitted to the Graduate
Faculty of Rensselaer Polytechnic Institute
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
Major Subject: COMPUTER SCIENCE

Approved:

_____
Petros Drineas, Thesis Adviser

_____
Malik Magdon-Ismail, Thesis Adviser

_____
Mohammed J. Zaki, Thesis Adviser

Rensselaer Polytechnic Institute
Troy, New York

November 2012
(For Graduation December 2012)

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

Firstly, I would like to thank my adviser Professor Petros Drineas for all the support and guidance in this work. I would also like to thank Professor Malik Magdon-Ismail for his stimulating suggestions and discussions during the development of this thesis and Dr. Christos Boutsidis for his various suggestions on improving the thesis. I would also like to thank Professor Mohammed Zaki for his teachings on data mining, which helped me in my research and this thesis.

I would like to thank my parents for their continuous support and encouragement. Lastly, I would like to thank the Computer Science Department at RPI, for providing me the environment which made this work possible.

ABSTRACT

Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be a data matrix of rank ρ, representing n points in $\mathbb{R}^d$. The linear support vector machine constructs a hyperplane separator that maximizes the 1-norm soft margin. We develop a new *oblivious* dimension reduction technique which is precomputed and can be applied to any input matrix $\mathbf{X}$. We prove that, with high probability, the margin and minimum enclosing ball in the feature space are preserved to within ϵ-*relative error*, ensuring comparable generalization as in the original space. We present extensive experiments on synthetic and real-world datasets in support of the theory.

CHAPTER 1
INTRODUCTION

1.1 Motivation

Feature selection and classification are most common tasks in machine learning. In machine learning applications, one often encounters datasets which have huge number of features out of which only a few are relevant. Thus selecting relevant features is an important task in machine learning. While large number of feature selection methods exist in the literature which may depend on the data, we propose a fast feature selection method oblivious to the data. Various types of classifiers exist in the machine learning literature. Among them, the Support Vector Machine (SVM) [1] is one of the most celebrated classifiers. It has been used in a vast number of applications like bioinformatics, finance, text classification etc. Consider for example, a document-by-term matrix with several thousand features and two labels. A dimensionally-reduced data matrix with few hundred features decreases running time of SVM training and also exhibits nearly equal out-of-sample error.

For a learning task, we split the data into a training set and a test set. The training data set consists of n points $\mathbf{x}_i \in \mathbb{R}^d$, with respective labels $y_i \in \{-1, +1\}$ for $i = 1 \dots n$. For linearly separable data, the primal form of the SVM learning problem is to construct a hyperplane $\mathbf{w}^*$ which maximizes the geometric *margin* (the minimum distance of a data point to the hyperplane), while separating the data. For non-separable data one maximizes the "soft" 1-norm margin. The dual Lagrangian formulation of the problem leads to the following quadratic program:

$$\begin{aligned} \max_{\{\alpha_i\}} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to} \quad & \sum_{i=1}^{n} y_i \alpha_i = 0, \qquad \text{and} \qquad 0 \le \alpha_i \le C, \quad i = 1, \dots, n. \end{aligned} \tag{1.1}$$

In the above formulation, the unknown lagrange multipliers $\{\alpha_i\}_{i=1}^n$ are constrained

to lie inside the 'box constraint' $[0, C]^n$, where C is part of the input. One way to understand the out-of-sample performance of the SVM is through the VC-dimension of *fat*-separators. If the data lies in a ball of radius B and the hypothesis set is hyperplanes of width γ (corresponding to the margin), then the VC-dimension of this hypothesis set is $O\left(B^2/\gamma^2\right)$ [2]. Now, given the in-sample error, one obtains a bound for the out-of-sample error, which is monotonic in the VC-dimension [3].

We develop a new algorithm for linear dimension reduction that is *oblivious* to the data. We construct a dimension reduction matrix $\mathbf{R} \in \mathbb{R}^{d\times r}$ which produces r-dimensional feature vectors $\tilde{\mathbf{x}}_i = \mathbf{R}^T\mathbf{x}_i$; the matrix $\mathbf{R}$ does not depend on the data. We show that for the data in the dimension-reduced space, the margin of separability and radius are preserved. So, the SVM with an appropriate structure defined by the margin (width) of the hyperplanes [3] will have comparable VC-dimension and hence generalization.

1.2 Notation

$\mathbf{A}, \mathbf{B}, \ldots$ denote matrices and $\mathbf{a}, \mathbf{b}, \ldots$ denote column vectors; $\mathbf{e}_i$ (for all $i = 1 \ldots n$) is the standard basis, whose dimensionality will be clear from context; and $\mathbf{I}_n$ is the $n \times n$ identity matrix. The Singular Value Decomposition (SVD) of a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ of rank $\rho \leq \min\{n, d\}$ is equal to $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{U} \in \mathbb{R}^{n\times\rho}$ is an orthogonal matrix containing the left singular vectors, $\Sigma \in \mathbb{R}^{\rho\times\rho}$ is a diagonal matrix containing the singular values $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_\rho > 0$, and $V \in \mathbb{R}^{d\times\rho}$ is a matrix containing the right singular vectors. The spectral norm of a matrix $\mathbf{A}$ is equal to the largest singular value of $\mathbf{A}$, i.e., $\|\mathbf{A}\|_2 = \sigma_1$. We will use the Hadamard-Walsh matrix: for any d that is a power of two, define

$$\mathbf{H}_d = \begin{bmatrix} \mathbf{H}_{d/2} & \mathbf{H}_{d/2} \\ \mathbf{H}_{d/2} & -\mathbf{H}_{d/2} \end{bmatrix} \in \mathbb{R}^{d\times d}, \qquad \text{with} \qquad \mathbf{H}_1 = +1.$$

The normalized Hadamard-Walsh matrix is $\sqrt{\frac{1}{d}}\mathbf{H}_d$, which we simply denote by $\mathbf{H}$.

1.3 SVM Basics.

We introduce matrix notation that we will use for the remainder of the paper. Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be the data matrix whose rows are the vectors $\mathbf{x}_i^T$, where each data-point is of d-dimensions, $\mathbf{Y} \in \mathbb{R}^{n \times n}$ be the diagonal matrix with entries $\mathbf{Y}_{ii} = y_i$ representing the class or labels, and $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n] \in \mathbb{R}^n$ be the vector of lagrange multipliers to be determined by solving the quadratic optimization problem. The SVM optimization problem is

$$\begin{aligned} \max_{\alpha} \quad & \mathbf{1}^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^{\boldsymbol{T}}\mathbf{Y}\mathbf{X}\mathbf{X}^T\mathbf{Y}\boldsymbol{\alpha} \\ \text{subject to} \quad & \mathbf{1}^T\mathbf{Y}\boldsymbol{\alpha} = 0; \qquad \text{and} \qquad \mathbf{0} \leq \boldsymbol{\alpha} \leq \boldsymbol{C}. \end{aligned} \tag{1.2}$$

In the above formulation, $\mathbf{1}$, $\mathbf{0}$, $\boldsymbol{C}$ are vectors with the implied constant entry. Let $\boldsymbol{\alpha^*}$ be an optimal solution of the above problem. The optimal hyperplane is $\mathbf{w}^* = \mathbf{X}^T\mathbf{Y}\boldsymbol{\alpha^*} = \sum_{i=1}^{n} y_i\alpha_i^*\mathbf{x}_i$, and the points $\mathbf{x}_i$ for which $\alpha_i^* > 0$, i.e. the points which appear in the expansion $\mathbf{w}^*$, are the support vectors. In most cases, the number of support vectors is small, i.e., $\boldsymbol{\alpha^*}$ is a sparse vector. The geometric margin, γ^*, of this canonical optimal hyperplane is $\gamma^* = 1/\left\|\mathbf{w}^*\right\|_2$, where $\left\|\mathbf{w}^*\right\|_2^2 = \sum_{i=1}^{n} \alpha_i^*$. The data radius is $B = \min_{\mathbf{x}^*} \max_{\mathbf{x}_i} \left\|\mathbf{x}_i - \mathbf{x}^*\right\|_2$. The hyperplane is used to classify the test points. The generalization error of SVM depends on $O\left(B^2/\gamma^{*2}\right)$. Since B is fixed by the training set, we try to find a hyperplane which maximizes γ^* and reduces the generalization error.

The above formulation holds good for binary classification. For multi-class classification, several techniques are available for SVM. We use the one-against-one technique. If there are m classes, then $m(m-1)/2$ classifiers are constructed. For each pair of classes, we solve the SVM optimization problem. Voting strategy is used by the classifiers where votes are cast for all data points. The data point is assigned the class for which it has received most votes. If there is a tie, then the class appearing first, is given the preference.

1.4 Our Contribution

One important observation from eqn 1.2 is that the product $\mathbf{X}\mathbf{X}^T$ is independent of the dimension of the data. However, if the number of dimensions is huge, then computing this product will be expensive. Our goal is to study how the SVM transforms under a dimension reduction, that is linear w.r.t the size of the data matrix. Let $\mathbf{R} \in \mathbb{R}^{d \times r}$ be the dimension reduction matrix that reduces the dimensionality of the input from d to $r \ll d$. We will choose $\mathbf{R}$ to be a random projection matrix. The transformed data set into r dimensions is $\tilde{\mathbf{X}} = \mathbf{X}\mathbf{R}$, and the SVM optimization problem becomes

$$\begin{aligned} \max_{\tilde{\boldsymbol{\alpha}}} \quad & \mathbf{1}^T\tilde{\boldsymbol{\alpha}} - \frac{1}{2}\tilde{\boldsymbol{\alpha}}^{\boldsymbol{T}}\mathbf{Y}\mathbf{X}\mathbf{R}\mathbf{R}^T\mathbf{X}^T\mathbf{Y}\tilde{\boldsymbol{\alpha}} \\ \text{subject to} \quad & \mathbf{1}^T\mathbf{Y}\tilde{\boldsymbol{\alpha}} = 0; \qquad \text{and} \qquad \mathbf{0} \leq \tilde{\boldsymbol{\alpha}} \leq \boldsymbol{C}. \end{aligned} \tag{1.3}$$

Solving the dimensionally-reduced problem above is computationally more efficient than solving the original, d-dimensional problem. We will present a construction for $\mathbf{R}$ that leverages the fast Hadamard transform. The running time needed to apply this construction to the original data matrix is $O(nd \log r)$. Notice that while this running time is nearly linear on the size of the original data, it does not take advantage of any sparsity in the input. In order to address this deficiency, we leverage the recent work of [4], which proposes a construction for $\mathbf{R}$ that can be applied to $\mathbf{X}$ in $O(nnz(\mathbf{X}) \log(\rho))$ time; here $nnz(\mathbf{X})$ denotes the number of non-zero entries of $\mathbf{X}$ and ρ is the rank of $\mathbf{X}$. To the best of our knowledge, this is the first independent implementation and evaluation of this potentially ground-breaking random projection technique (a few experimental results were presented in [4]). All constructions for $\mathbf{R}$ are oblivious of the data and hence they can be precomputed. Also, all generalization bounds that depend on the final margin and radius of the data will continue to hold.

In the transformed space, let the resulting margin after solving the optimization problem be $\tilde{\gamma}^*$ and suppose that the data reside in a minimum enclosing ball of radius $\tilde{B}$. Our main theoretical result is to show that for a suitably chosen r (reduced dimension), both the margin and the data radius are preserved to relative

error:

$$\tilde{\gamma}^{*2} \geq (1-\epsilon)\gamma^{*2}; \qquad \tilde{B}^2 \leq (1+\epsilon)B^2.$$

Thus, it is possible to *obliviously* reduce the dimension of the data while preserving the good generalization properties of the SVM. We briefly discuss the appropriate values of r: if $\mathbf{R}$ is the randomized Hadamard transform, we need to set $r = O\left(\rho\epsilon^{-2}\log^2\left(d\rho\epsilon^{-2}\right)\right)$; if $\mathbf{R}$ is constructed as described in [4], then $r = O\left(\rho\epsilon^{-4}\log\left(\rho/\epsilon\right)\left(\rho+\log\left(1/\epsilon\right)\right)\right)$. The running time needed to apply the former transform on $\mathbf{X}$ is $O\left(nd\log\rho\right)$; the running time needed to apply the latter transform is $O\left(nnz(\mathbf{X})\log\rho\right)$.

CHAPTER 2
PRIOR WORK

The quadratic programming problem for SVM training is solvable in polynomial time. Several speed-up techniques for solving SVMs exist. One of them is subset selection, where the problem is divided into smaller sub-problems, thereby reducing size of QP problem. To the best of our knowledge, the work most closely related to our results, is the paper of Krishnan *et al* [5], who improved upon the work of Balcazar *et al* [6].

Balcazar *et al* [6] proposed a random-sampling based subset-selection type algorithm for SVM training. The algorithm requires the existence of some hypothesis for convergence and works for primal form linear SVM. It initializes the weights to all training points to the same value, then randomly selects some training points and solves the optimization problem. The training points which were misclassified had their weights increased. Then the training points were chosen randomly according to their weights. This process is repeated several times to get the global solution. The algorithm requires $O(d \log n)$ iterations to terminate for the linearly separable case. For the non-linearly separable case, the authors use the reduced convex hull method to estimate the combinatorial dimension of the problem. The algorithm takes a large number of steps to terminate. The drawback of these algorithms are that they are not realizable in practice.

Krishnan *et al* [5] overcame the problem by using sub-problems based on Gaussian random projections. They showed that the minimal number of support vectors needed to solve the SVM problem such that the solution is near optimal with high probability is $O(\log n)$. By using random projections, they obtained a solution to the SVM problem having a margin that is relative-error close to optimal. Their sampling complexity (the parameter r in our parlance) depended on B^4, and, most importantly, on $1/\gamma^{*2}$. Experiments on real and synthetic datasets were used to demonstrate the working of large scale SVM solver. The bound on their sampling complexity is not directly comparable to our result, which only depends on the rank

of the data manifold, and holds regardless of the margin of the original problem (which could be arbitrarily small). Our results dramatically improve the running time needed to apply the random projections; our running times are (theoretically) linear in the number of non-zero entries in $\mathbf{X}$, whereas [5] necessitates $O(ndr)$ time to apply $\mathbf{R}$ on $\mathbf{X}$.

Shi *et al* [7] establish the conditions under which margins are preserved after random projection and show that error free margins are preserved for both binary and multi-class problems if these conditions are met. They discuss the theory of margin and angle preservation after random projections using Gaussian matrices. They show that margin preservation is closely related to acute angle preservation and inner product preservation. Smaller acute angle leads to better preservation of the angle and the inner product. When the angle is well preserved, the margin is well-preserved too. They provide bounds on error-free margin preservation for arbitrary tolerances. However, these results are very different from ours, since they are not directly tied to SVMs.

Finally, it is worth noting that random projection techniques have been applied extensively in the compressed sensing literature, and our theorems have the same flavor to a number of results in that area. However, to the best of our knowledge, the compressed sensing literature has not investigated the 1-norm soft-margin SVM optimization problem.

CHAPTER 3
RANDOM PROJECTION

3.1 Background

Random projections are a useful technique to reduce dimensionality. Let the data matrix be $\mathbf{X} \in \mathbb{R}^{n \times d}$ (n data points in $\mathbb{R}^d$); For $\mathbf{R} \in \mathbb{R}^{d \times r}$ (with $r \ll d$), the projected data matrix is $\tilde{\mathbf{X}} = \mathbf{XR} \in \mathbb{R}^{n \times r}$ (n points in $\mathbb{R}^r$). If $\mathbf{R}$ is carefully chosen, then all pairwise Euclidean distances are preserved with high probability, hence the geometry of the problem is preserved.

A classical result of Johnsonson and Lindenstrauss [8] states that for any $0 \leq \epsilon \leq \frac{1}{2}$, any set of n points in d dimensions can be projected into a r-dimensional subspace where $r = O\left(\frac{\log n}{\epsilon^2}\right)$ while preserving pairwise distances of points within distortion factor of $1 \pm \epsilon$. If $\mathbf{x}_i$ and $\mathbf{x}_j$ are two points in d dimensions which are projected onto a r-dimensional sub-space, then with high probability,

$$(1-\epsilon)\ \|\mathbf{x}_i - \mathbf{x}_j\|_2 \leq \left\|\mathbf{R}^T\mathbf{x}_i - \mathbf{R}^T\mathbf{x}_j\right\|_2 \leq (1+\epsilon)\ \|\mathbf{x}_i - \mathbf{x}_j\|_2 .$$

We will call the result of Johnson and Lindenstrauss JL for short. They showed that $\mathbf{R}$ can be constructed using a random orthonormal matrix. There exists several constructions for random projection matrices. We will describe four such constructions.

3.1.1 Gaussian Matrix

Gupta and Dasgupta [9, 10] simplified the proof of JL and showed that such an embedding can be constructed using a random Gaussian matrix R, i.e. a matrix whose entries are i.i.d Gaussian random variables with zero mean and variance $\frac{1}{\sqrt{r}}$. Computing the random projection of a $n \times d$ matrix takes $O(ndr)$ time.

3.1.2 Achlioptas Method

Achlioptas [11] showed much simpler constructions of JL-embeddings are possible without sacrificing the quality of the embeddings. He showed that by using

Input: d (dimensionality of feature space), r (dimensionality of projected feature space)

Output: Random projection matrix $\mathbf{R} \in \mathbb{R}^{d \times r}$

1. Let $\mathbf{R} \in \mathbb{R}^{d \times d}$ be a matrix with $\mathbf{R}_{ii}$ equal to $\frac{+1}{\sqrt{r}}$ or $\frac{-1}{\sqrt{r}}$ with probability 1/2.
2. Return $\mathbf{R}$.

Algorithm 1: A random projection matrix based on the Random Sign Matrix

a random-sign matrix, i.e. a matrix whose entries are $\pm 1/\sqrt{r}$ with probability 1/2 it is possible to perform random projections. This led to less storage space but computation time remained the same. We call this method **RS** for short.

3.1.3 Subsampled Fast Hadamard Transform

More recently, faster methods of constructing random projections have been developed, using, for example, the Fast Hadamard Transform - **FHT** for short [12]. We will use the Hadamard-Walsh matrix: for any d that is a power of two, define

$$\mathbf{H}_d = \begin{bmatrix} \mathbf{H}_{d/2} & \mathbf{H}_{d/2} \\ \mathbf{H}_{d/2} & -\mathbf{H}_{d/2} \end{bmatrix} \in \mathbb{R}^{d \times d}, \qquad \text{with} \qquad \mathbf{H}_1 = +1.$$

The normalized Hadamard-Walsh matrix is $\sqrt{\frac{1}{d}}\mathbf{H}_d$, which we simply denote by $\mathbf{H}$. We set:

$$\mathbf{R}_{\text{SRHT}} = \sqrt{\frac{d}{r}}\mathbf{DHS}, \tag{3.1}$$

a rescaled product of three matrices. $\mathbf{D} \in \mathbb{R}^{d \times d}$ is a random diagonal matrix with $\mathbf{D}_{ii}$ equal to ± 1 with probability $\frac{1}{2}$. $\mathbf{H} \in \mathbb{R}^{d \times d}$ is the normalized Hadamard transform matrix. $\mathbf{S} \in \mathbb{R}^{d \times r}$ is a random *sampling matrix* which randomly samples columns of $\mathbf{DH}$; specifically, each of the r columns of $\mathbf{S}$ is independent and selected uniformly at random (with replacement) from the columns of $\mathbf{I}_d$, the identity matrix. This construction assumes that d is a power of 2. If not, we just pad X with columns of zeros (affecting run times by at most a factor of 2). An important property of $\mathbf{R}$

(that follows from prior work) is that it preserves orthogonality. Another important property of this transform is that the projected features $\tilde{\mathbf{X}} = \mathbf{XR}$ can be computed efficiently in $O(nd \ln r)$ time as can be seen from Theorem 2.1 of [13].

Input: d (dimensionality of feature space), r (dimensionality of projected feature space)

Output: Random projection matrix $\mathbf{R} \in \mathbb{R}^{d \times r}$

1. Let $\mathbf{S}$ be a $d \times r$ all zeros matrix.
2. **For** $j = 1, \ldots, r$ (i.i.d. trials with replacement) **select uniformly at random** an integer from $\{1, 2, \ldots, d\}$.
   - **If** i is selected, **then** set $\mathbf{S}_{ij} = \sqrt{d/r}$.
3. Let $\mathbf{H} \in \mathbb{R}^{d \times d}$ be the normalized Hadamard transform matrix.
4. Let $\mathbf{D} \in \mathbb{R}^{d \times d}$ be a diagonal matrix with $\mathbf{D}_{ii}$ equal to $+1$ or -1 with probability $1/2$.
5. Return $\mathbf{R} = \mathbf{DHS}$.

Algorithm 2: A random projection matrix based on the Randomized Hadamard Transform. (If d is not a power of two, we simply construct a matrix for $d' \geq d$, such that d' is the nearest integer that is a power of two.)

3.1.4 Clarkson-Woodruff Method

While the randomized Hadamard transform is a major improvement over prior work, it does not take advantage of any sparsity in the input matrix. To fix this, very recent work [4] shows that carefully constructed random projection matrices can be applied in input sparsity time by making use of generalized sparse embedding matrices. We will call the method of Clarkson and Woodruff [4] to construct a sparse embedding matrix **CW**.

To understand their construction of $\mathbf{R}$, assume that the rank of $\mathbf{X}$ is ρ and let $r = O(\rho\epsilon^{-4} \log(\rho/\epsilon)(\rho + \log(1/\epsilon)))$. Then, let $k = \Theta(\epsilon^{-2} \log(r/\epsilon))$, let $v = \Theta(\epsilon^{-1})$, and let $q = r/k$ be an integer (by appropriately choosing the constants). The construction starts by letting $h : 1 \ldots d \to 1 \ldots q$ be a random hash function;

Input: Two integers a and b, so that integers from $1, 2, \cdots a$ are mapped to $1, 2, \cdots b$

Output: An array h containing mapped integers

1. For $i = 1, 2, \cdots a$
   - $h(i) = g$, where g is chosen from $1, 2, \cdots b$ with probability $1/b$.

Algorithm 3: Random hash function

Input: Integers w, v and a_i .

Output: B

1. Initialize $\mathbf{B}$ to be an empty matrix.
2. For $i = 1 \cdots w$
   - Use Algorithm 3 to generate random map from a_i to v.
   - Φ is a $v \times a_i$ binary matrix such that $\Phi_{h(i),i} = 1$ and rest of the entries are zero.
   - $\mathbf{D}$ is a $a_i \times a_i$ diagonal matrix with each entry chosen to be +1 or -1 with probability 1/2.
   - $\mathbf{B} = \left[\mathbf{B}; \sqrt{v/k}\Phi_i D_i\right]$ i.e stack $\left(\sqrt{v/k}\right) \Phi_i D_i$ vertically to form $\mathbf{B}$.

Algorithm 4: Algorithm to construct B

then, for $i = 1 \ldots q$, let $a_i = |h^{-1}(i)|$ and let $d = \sum_{i=1}^{q} a_i$. The construction proceeds by creating q independent matrices $\mathbf{B}_1 \ldots \mathbf{B}_q$, such that $\mathbf{B}_i \in \mathbb{R}^{k \times a_i}$. Each $\mathbf{B}_i$ is the concatenation (stacking the rows of matrices on top of each other) of the following matrices: $\sqrt{\frac{v}{k}}\mathbf{\Phi_1 D}_1 \ldots \sqrt{\frac{v}{k}}\mathbf{\Phi}_{k/v}\mathbf{D}_{k/v}$. The matrix $\mathbf{\Phi_i D}_i \in \mathbb{R}^{v \times a_i}$ is defined as follows: for each $m \in \{1 \ldots a_i\}$, $h(m) = g'$, where g' is selected from $\{1 \ldots v\}$ uniformly at random. $\mathbf{\Phi_i}$ is a $v \times a_i$ binary matrix with $\mathbf{\Phi_{h(m),m}} = 1$ and all remaining entries set to zero. $\mathbf{D}$ is an $a_i \times a_i$ random diagonal matrix, with each diagonal entry

independently set to be $+1$ or -1 with probability $1/2$. Finally, let $\mathbf{S}$ be the block diagonal matrix constructed by stacking the $\mathbf{B}_i$'s across its diagonal and let $\mathbf{P}$ be a $d \times d$ permutation matrix; then, $\mathbf{R} = (\mathbf{SP})^T$.

Input: d (dimensionality of feature space), t (dimensionality of projected feature space)

Output: Random projection matrix $\mathbf{R} \in \mathbb{R}^{d \times t}$

1. Choose v and k such that $q = t/k$ and $w = k/v$.
2. Use Algorithm 3 to generate a random map from n to t.
3. Initialize $\mathbf{S}$ to be an empty matrix.
4. For $i = 1 \cdots q$
   - $a_i = |h^{-1}(i)|$ such that $\sum_{i=1}^{q} a_i = d$
   - Use algorithm 4 to construct $\mathbf{B}$ using w and a_i
5. Place $\mathbf{B}_1, \mathbf{B}_2, ...\mathbf{B}_q$ along the block diagonal of $\mathbf{S}$.
6. $\mathbf{P}$ is a $d \times d$ permutation matrix.
7. $\mathbf{R} = (\mathbf{SP})^T$

Algorithm 5: A random projection matrix based on Generalized Sparse Embedding Matrices.

CHAPTER 4
Geometry of SVM is preserved under Random Projection

We now state and prove our main result, namely that solving the SVM optimization problem in the projected space results in comparable margin and data radius as in the original space, indicating comparable generalization performance as in the original feature space. These results are dependent on the technical result that $\|\mathbf{V}^T\mathbf{V} - \mathbf{V}^T\mathbf{R}\mathbf{R}^T\mathbf{V}\|_2 \leq \epsilon$ holds with high probability for different constructions of the random projection matrix. We state this technical result in the form of three lemmas below.

Lemma 1. *Fix $\epsilon \in (0, \frac{1}{2}]$, $\delta \in (0, 1]$. Let $\mathbf{V} \in \mathbb{R}^{d\times\rho}$ be any matrix with orthonormal columns and set $\mathbf{R} = \mathbf{R}_{\text{SRHT}}$ as in eqn (3.1), with $r = O(\rho\epsilon^{-2} \cdot \log(\rho d\delta^{-1}) \cdot \log(\rho\epsilon^{-2}\delta^{-1}\log(\rho d\delta^{-1})))$. Then, with probability at least $1 - \delta$,*

$$\|\mathbf{V}^T\mathbf{V} - \mathbf{V}^T\mathbf{R}\mathbf{R}^T\mathbf{V}\|_2 \leq \epsilon.$$

Proof. Consider the matrix $\mathbf{V}^T\mathbf{R} = \mathbf{V}^T\mathbf{D}\mathbf{H}\mathbf{S}$. Using Lemma 3 of [14],

$$\left\|(\mathbf{HDV})_{(i)}\right\|_2^2 \leq \frac{2\rho\ln(40d\rho)}{d} \Rightarrow (2\ln(40d\rho))^{-1} \frac{\left\|(\mathbf{HDV})_{(i)}\right\|_2^2}{\rho} \leq \frac{1}{d}$$

holds for all $i = 1, \ldots, d$ with probability at least $1 - \delta$. In the above, the notation $\mathbf{A}_{(i)}$ denotes the i-th row of $\mathbf{A}$ as a row vector. Applying Theorem 4 with $\beta = (2\ln(40d\rho))^{-1}$ ([14], Appendix) concludes the lemma. □

Lemma 2. *Fix $\epsilon \in (0, \frac{1}{2}]$ and let $\mathbf{V} \in \mathbb{R}^{d\times\rho}$ be any matrix with stable rank at most ρ. Let $\mathbf{R}$ be a $d \times \rho$ random sign matrix rescaled by $1/\sqrt{t}$. If $r = O(\rho\epsilon^{-2}\log\rho\log d)$, then with probability at least $1 - 1/n$,*

$$\|\mathbf{V}^T\mathbf{V} - \mathbf{V}^T\mathbf{R}\mathbf{R}^T\mathbf{V}\|_2 \leq \epsilon.$$

Proof. The proof of this result is essentially the same, using Theorem 3.1(i) of [15].

□

Lemma 3. *Let $\epsilon \in (0,1)$ and $\mathbf{R}$ be the random projection matrix constructed as described in 3.1.4 of dimensions $d \times r$, with $r = O\left(\rho\epsilon^{-4}\log\left(\rho/\epsilon\right)\left(\rho + \log\left(1/\epsilon\right)\right)\right)$, and assume that the following is true : For any vector x,*

$$(1-\epsilon)\ \|Vx\|_2^2 \leq \left\|R^T Vx\right\|_2^2 \leq (1+\epsilon)\ \|Vx\|_2^2 .$$

Then, $\left\|\mathbf{V}^T\mathbf{V} - \mathbf{V}^T\mathbf{R}\mathbf{R}^T\mathbf{V}\right\|_2 \leq \epsilon$.

Proof. The assumption,

$$(1-\epsilon)\ \|\mathbf{V}\mathbf{x}\|_2^2 \leq \left\|\mathbf{R}^T\mathbf{V}\mathbf{x}\right\|_2^2 \leq (1+\epsilon)\ \|\mathbf{V}\mathbf{x}\|_2^2$$

is equivalent to

$$(1-\epsilon)\,\mathbf{x}^T\mathbf{V}^T\mathbf{V}\mathbf{x} \leq \mathbf{x}^T\mathbf{V}^T\mathbf{R}\mathbf{R}^T\mathbf{V}\mathbf{x} \leq (1+\epsilon)\,\mathbf{x}^T\mathbf{V}^T\mathbf{V}\mathbf{x},$$

which in turn is equivalent to (focus on the second inequality),

$$\mathbf{x}^T\left(\mathbf{V}^T\mathbf{R}\mathbf{R}^T\mathbf{V} - \mathbf{V}^T\mathbf{V}\right)\mathbf{x} \leq \epsilon\mathbf{x}^T\mathbf{V}^T\mathbf{V}\mathbf{x}.$$

Apply this for $\mathbf{x} = \mathbf{y}$ where $\mathbf{y}$ is the eigenvector of $\left(\mathbf{V}^T\mathbf{R}\mathbf{R}^T\mathbf{V} - \mathbf{V}^T\mathbf{V}\right)$ that corresponds to the largest eigenvalue of $\left(\mathbf{V}^T\mathbf{R}\mathbf{R}^T\mathbf{V} - \mathbf{V}^T\mathbf{V}\right)$ So,

$$\lambda_{max}\left(\mathbf{V}^T\mathbf{R}\mathbf{R}^T\mathbf{V} - \mathbf{V}^T\mathbf{V}\right) \leq \epsilon\mathbf{y}^T\mathbf{y}.$$

Notice that $\mathbf{y}^T$ has unit norm, so $\lambda_{max}\left(\mathbf{V}^T\mathbf{R}\mathbf{R}^T\mathbf{V} - \mathbf{V}^T\mathbf{V}\right) \leq \epsilon$.
Now since $\left(\mathbf{V}^T\mathbf{R}\mathbf{R}^T\mathbf{V} - \mathbf{V}^T\mathbf{V}\right)$ is symmetric, so

$$\lambda_{max}\left(\mathbf{V}^T\mathbf{R}\mathbf{R}^T\mathbf{V} - \mathbf{V}^T\mathbf{V}\right) = \left\|\mathbf{V}^T\mathbf{V} - \mathbf{V}^T\mathbf{R}\mathbf{R}^T\mathbf{V}\right\|_2 \leq \epsilon.$$

□

4.1 Margin Preservation

Theorem 1. *Let $\epsilon \in (0, \frac{1}{2}]$ be an accuracy parameter, $\mathbf{R} \in \mathbb{R}^{d\times r}$ be any matrix for which $\|\mathbf{V}^T\mathbf{V} - \mathbf{V}^T\mathbf{R}\mathbf{R}^T\mathbf{V}\|_2 \leq \epsilon$, and let $\tilde{X} = \mathbf{X}\mathbf{R}$. Let γ^* and $\tilde{\gamma}^*$ be the margins obtained by solving the SVM problems using data $\mathbf{X}$ and $\tilde{\mathbf{X}}$ respectively (eqns. (1.2) and (1.3)). Then, $\tilde{\gamma}^{*2} \geq (1-\epsilon)\cdot\gamma^{*2}$.*

Theorem 1 will follow from Lemma 1, 2 or 3 depending on the choice of random projection matrix.

Proof: (of Theorem 1) Let $\mathbf{E} = \mathbf{V}^T\mathbf{V} - \mathbf{V}^T\mathbf{R}\mathbf{R}^T\mathbf{V}$, and $\alpha^* = [\alpha_1^*, \alpha_2^*, \ldots, \alpha_n^*]^T \in \mathbb{R}^n$ be the vector achieving the optimal solution for the problem of eqn. (1.2) in Section 1.3. Then,

$$\begin{aligned} Z_{opt} &= \sum_{i=1}^{n}\alpha_i^* - \frac{1}{2}\alpha^{*T}\mathbf{Y}\mathbf{X}\mathbf{X}^T\mathbf{Y}\alpha^* \\ &= \sum_{i=1}^{n}\alpha_i^* - \frac{1}{2}\alpha^{*T}\mathbf{Y}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}\mathbf{U}^T\mathbf{Y}\alpha^* \\ &= \sum_{i=1}^{n}\alpha_i^* - \frac{1}{2}\alpha^{*T}\mathbf{Y}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{R}\mathbf{R}^T\mathbf{V}\mathbf{\Sigma}\mathbf{U}^T\mathbf{Y}\alpha^* \\ &\quad -\frac{1}{2}\alpha^{*T}\mathbf{Y}\mathbf{U}\mathbf{\Sigma}\mathbf{E}\mathbf{\Sigma}\mathbf{U}^T\mathbf{Y}\alpha^*. \end{aligned} \tag{4.1}$$

Let $\tilde{\boldsymbol{\alpha}}^* = [\tilde{\alpha}_1^*, \tilde{\alpha}_2^*, \ldots, \tilde{\alpha}_n^*]^T \in \mathbb{R}^n$ be the vector achieving the optimal solution for the dimensionally-reduced SVM problem of eqn. (1.3) using $\tilde{\mathbf{X}} = \mathbf{X}\mathbf{R}$. Using the SVD of $\mathbf{X}$, we get

$$\tilde{Z}_{opt} = \sum_{i=1}^{n}\tilde{\alpha}_i^* - \frac{1}{2}\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{Y}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{R}\mathbf{R}^T\mathbf{V}\mathbf{\Sigma}\mathbf{U}^T\mathbf{Y}\tilde{\boldsymbol{\alpha}}^*. \tag{4.2}$$

Since the constraints on $\alpha^*, \tilde{\boldsymbol{\alpha}}^*$ do not depend on the data (see eqns. (1.2) and (1.3)), it is clear that $\tilde{\boldsymbol{\alpha}}^*$ is a feasible solution for the problem of eqn. (1.2). Thus, from

the optimality of α^*, and using eqn. (4.2), it follows that

$$\begin{aligned} Z_{opt} &= \sum_{i=1}^{n} \alpha_i^* - \frac{1}{2}\alpha^{*T}\mathbf{Y}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{R}\mathbf{R}^T\mathbf{V}\mathbf{\Sigma}\mathbf{U}^T\mathbf{Y}\alpha^* \\ &\quad -\frac{1}{2}\alpha^{*T}\mathbf{Y}\mathbf{U}\mathbf{\Sigma}\mathbf{E}\mathbf{\Sigma}\mathbf{U}^T\mathbf{Y}\alpha^* \\ &\geq \sum_{i=1}^{n} \tilde{\alpha}_i^* - \frac{1}{2}\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{Y}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{R}\mathbf{R}^T\mathbf{V}\mathbf{\Sigma}\mathbf{U}^T\mathbf{Y}\tilde{\boldsymbol{\alpha}}^* \\ &\quad -\frac{1}{2}\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{Y}\mathbf{U}\mathbf{\Sigma}\mathbf{E}\mathbf{\Sigma}\mathbf{U}^T\mathbf{Y}\tilde{\boldsymbol{\alpha}}^* \\ &= \tilde{Z}_{opt} - \frac{1}{2}\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{Y}\mathbf{U}\mathbf{\Sigma}\mathbf{E}\mathbf{\Sigma}\mathbf{U}^T\mathbf{Y}\tilde{\boldsymbol{\alpha}}^*. \end{aligned} \tag{4.3}$$

We now analyze the second term using standard sub-multiplicativity properties and $\mathbf{V}^T\mathbf{V} = \mathbf{I}$. Taking $\mathbf{Q} = \left\|\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{Y}\mathbf{U}\mathbf{\Sigma}\right\|_2$

$$\begin{aligned} \frac{1}{2}\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{Y}\mathbf{U}\mathbf{\Sigma}\mathbf{E}\mathbf{\Sigma}\mathbf{U}^T\mathbf{Y}\tilde{\boldsymbol{\alpha}}^* &\leq \frac{1}{2}\left\|\mathbf{Q}\right\|_2 \left\|\mathbf{E}\right\|_2 \left\|\mathbf{Q}^T\right\|_2 \\ &= \frac{1}{2}\left\|\mathbf{E}\right\|_2 \left\|\mathbf{Q}\right\|_2^2 \\ &= \frac{1}{2}\left\|\mathbf{E}\right\|_2 \left\|\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{Y}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\right\|_2^2 \\ &= \frac{1}{2}\left\|\mathbf{E}\right\|_2 \left\|\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{Y}\mathbf{X}\right\|_2^2. \end{aligned} \tag{4.4}$$

Combining eqns. (4.3) and (4.4), we get

$$Z_{opt} \geq \tilde{Z}_{opt} - \frac{1}{2}\left\|\mathbf{E}\right\|_2 \left\|\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{Y}\mathbf{X}\right\|_2^2. \tag{4.5}$$

We now proceed to bound the second term in the right-hand side of the above

equation. Towards that end, we bound the difference:

$$\begin{aligned}
& \left|\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{YXRR}^T\mathbf{X}^T\mathbf{Y}\tilde{\boldsymbol{\alpha}}^* - \tilde{\boldsymbol{\alpha}}^{*T}\mathbf{YXX}^T\mathbf{Y}\tilde{\boldsymbol{\alpha}}^*\right| \\
= & \left|\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{YU\Sigma}\left(\mathbf{V}^T\mathbf{RR}^T\mathbf{V} - \mathbf{V}^T\mathbf{V}\right)\mathbf{\Sigma U}^T\mathbf{Y}\tilde{\boldsymbol{\alpha}}^*\right| \\
= & \left|\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{YU\Sigma}\left(-\mathbf{E}\right)\mathbf{\Sigma U}^T\mathbf{Y}\tilde{\boldsymbol{\alpha}}^*\right| \\
\leq & \left\|\mathbf{E}\right\|_2 \left\|\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{YU\Sigma}\right\|_2^2 \\
= & \left\|\mathbf{E}\right\|_2 \left\|\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{YU\Sigma V}^T\right\|_2^2 \\
= & \left\|\mathbf{E}\right\|_2 \left\|\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{YX}\right\|_2^2.
\end{aligned}$$

We can rewrite the above inequality as $\left| \left\|\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{YXR}\right\|_2^2 - \left\|\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{YX}\right\|_2^2 \right| \leq \left\|\mathbf{E}\right\|_2 \left\|\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{YX}\right\|_2^2$; thus,

$$\left\|\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{YX}\right\|_2^2 \leq \frac{1}{1 - \left\|\mathbf{E}\right\|_2} \left\|\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{YXR}\right\|_2^2.$$

Combining with eqn. (4.5), we get

$$Z_{opt} \geq \tilde{Z}_{opt} - \frac{1}{2}\left(\frac{\left\|\mathbf{E}\right\|_2}{1 - \left\|\mathbf{E}\right\|_2}\right) \left\|\tilde{\boldsymbol{\alpha}}^{*T}\mathbf{YXR}\right\|_2^2. \tag{4.6}$$

Now recall from our discussion in Section 1.1 that $\mathbf{w}^{*T} = \alpha^{*T}\mathbf{YX}$, $\tilde{\mathbf{w}}^{*T} = \tilde{\boldsymbol{\alpha}}^{*T}\mathbf{YXR}$, $\left\|\mathbf{w}^*\right\|_2^2 = \sum_{i=1}^n \alpha_i^*$, and $\left\|\tilde{\mathbf{w}}^*\right\|_2^2 = \sum_{i=1}^n \tilde{\alpha}_i^*$. Then, the optimal solutions Z_{opt} and $\tilde{Z}_{opt}$ can be expressed as follows:

$$\begin{aligned}
Z_{opt} &= \left\|\mathbf{w}^*\right\|_2^2 - \frac{1}{2}\left\|\mathbf{w}^*\right\|_2^2 = \frac{1}{2}\left\|\mathbf{w}^*\right\|_2^2, & (4.7)\\
\tilde{Z}_{opt} &= \left\|\tilde{\mathbf{w}}^*\right\|_2^2 - \frac{1}{2}\left\|\tilde{\mathbf{w}}^*\right\|_2^2 = \frac{1}{2}\left\|\tilde{\mathbf{w}}^*\right\|_2^2. & (4.8)
\end{aligned}$$

Combining eqns. (4.6), (4.7), and (4.8), we get

$$\begin{aligned}
\left\|\mathbf{w}^*\right\|_2^2 &\geq \left\|\tilde{\mathbf{w}}^*\right\|_2^2 - \left(\frac{\left\|E\right\|_2}{1 - \left\|E\right\|_2}\right) \left\|\tilde{\mathbf{w}}^*\right\|_2^2 \\
&= \left(1 - \frac{\left\|E\right\|_2}{1 - \left\|E\right\|_2}\right) \left\|\tilde{\mathbf{w}}^*\right\|_2^2. & (4.9)
\end{aligned}$$

Let $\gamma^* = \left\|\mathbf{w}^*\right\|_2^{-1}$ be the geometric margin of the problem of eqn. (1.2) and let $\tilde{\gamma}^* = \left\|\tilde{\mathbf{w}}^*\right\|_2^{-1}$ be the geometric margin of the problem of eqn. (1.3). Then, the above

equation implies:

$$\begin{aligned} \gamma^{*2} &\leq \left(1 - \frac{\|E\|_2}{1 - \|E\|_2}\right)^{-1} \tilde{\gamma}^{*2} \\ \Rightarrow \tilde{\gamma}^{*2} &\geq \left(1 - \frac{\|E\|_2}{1 - \|E\|_2}\right) \gamma^{*2}. \qquad (4.10) \\ \Rightarrow \tilde{\gamma}^{*2} &\geq \left(\frac{1-2\epsilon}{1-\epsilon}\right) \cdot \gamma^{*2}. \qquad (4.11) \end{aligned}$$

Since ϵ is very small, $(1-2\epsilon)(1-\epsilon)^{-1} = (1-\epsilon)$, from where we get $\tilde{\gamma}^{*2} \geq (1-\epsilon)\gamma^{*2}$.

$\diamond$

4.2 Data Radius Preservation

Our second theorem argues that the radius of the minimum ball enclosing all projected points (the rows of the matrix $\mathbf{XR}$) is very close to the radius of the minimum ball enclosing all original points (the rows of the matrix $\mathbf{X}$).

Theorem 2. *Let $\epsilon \in (0, \frac{1}{2}]$ be an accuracy parameter and consider the SVM formulations of eqns. (1.2) and (1.3), let B be the radius of the minimum ball enclosing all points in the full-dimensional space, and let $\tilde{B}$ be the radius of the ball enclosing all points in the projected subspace. For $\mathbf{R}$ as in Theorem 1, with probability at least $1-\delta$, $\tilde{B}^2 \leq (1+\epsilon)B^2$.*

Proof:(of Theorem 2) We consider the matrix $\mathbf{X}_B \in \mathbb{R}^{(n+1)\times d}$ whose first n rows are the rows of $\mathbf{X}$ and whose last row is the vector $\mathbf{x}_B^T$; here $\mathbf{x}_B$ denotes the center of the minimum radius ball enclosing all n points. Then, the SVD of $\mathbf{X}_B$ is equal to $\mathbf{X}_B = \mathbf{U}_B \mathbf{\Sigma}_B \mathbf{V}_B^T$, where $\mathbf{U}_B \in \mathbb{R}^{(n+1)\times \rho_B}$, $\mathbf{\Sigma}_B \in \mathbb{R}^{\rho_B \times \rho_B}$, and $\mathbf{V} \in \mathbb{R}^{d \times \rho_B}$. Here ρ_B is the rank of the matrix $\mathbf{X}_B$ and clearly $\rho_B \leq \rho + 1$. (Recall that ρ is the rank of the matrix $\mathbf{X}$.) Let B be the radius of the minimal radius ball enclosing all n points in the original space. Then, for any $i = 1, \ldots, n$,

$$B^2 \geq \|\mathbf{x}_i - \mathbf{x}_B\|_2^2 = \left\| (\mathbf{e}_i - \mathbf{e}_{n+1})^T \mathbf{X}_B \right\|_2^2. \qquad (4.12)$$

Now consider the matrix $\mathbf{X}_B\mathbf{R}$ and notice that

$$\begin{aligned}
& \left| \left\| (\mathbf{e}_i - \mathbf{e}_{n+1})^T \mathbf{X}_B \right\|_2^2 - \left\| (\mathbf{e}_i - \mathbf{e}_{n+1})^T \mathbf{X}_B \mathbf{R} \right\|_2^2 \right| \\
= & \left| (\mathbf{e}_i - \mathbf{e}_{n+1})^T \left(\mathbf{X}_B \mathbf{X}_B^T - \mathbf{X}_B \mathbf{R} \mathbf{R}^T \mathbf{X}_B^T \right) (\mathbf{e}_i - \mathbf{e}_{n+1}) \right| \\
= & \left| (\mathbf{e}_i - \mathbf{e}_{n+1})^T \left(\mathbf{U}_B \mathbf{\Sigma}_B \mathbf{V}_B^T \mathbf{V}_B \mathbf{\Sigma}_B \mathbf{U}_B^T - \mathbf{U}_B \mathbf{\Sigma}_B \mathbf{V}_B^T \mathbf{R} \mathbf{R}^T \mathbf{V}_B \mathbf{\Sigma}_B \mathbf{U}_B^T \right) (\mathbf{e}_i - \mathbf{e}_{n+1}) \right| \\
= & \left| (\mathbf{e}_i - \mathbf{e}_{n+1})^T \mathbf{U}_B \mathbf{\Sigma}_B \mathbf{E}_B \mathbf{\Sigma}_B \mathbf{U}_B^T (\mathbf{e}_i - \mathbf{e}_{n+1}) \right| \\
\leq & \left\| \mathbf{E}_B \right\|_2 \left\| (\mathbf{e}_i - \mathbf{e}_{n+1})^T \mathbf{U}_B \mathbf{\Sigma}_B \right\|_2^2 \\
= & \left\| \mathbf{E}_B \right\|_2 \left\| (\mathbf{e}_i - \mathbf{e}_{n+1})^T \mathbf{U}_B \mathbf{\Sigma}_B \mathbf{V}_B^T \right\|_2^2 \\
= & \left\| \mathbf{E}_B \right\|_2 \left\| (\mathbf{e}_i - \mathbf{e}_{n+1})^T \mathbf{X}_B \right\|_2^2 .
\end{aligned}$$

In the above, we let $\mathbf{E}_B \in \mathbb{R}^{\rho_B \times \rho_B}$ be the matrix that satisfies $\mathbf{V}_B^T \mathbf{V}_B = \mathbf{V}_B^T \mathbf{R} \mathbf{R}^T \mathbf{V}_B + \mathbf{E}_B$, and we also used $\mathbf{V}_B^T \mathbf{V}_B = \mathbf{I}$. Now consider the ball whose center is the $(n+1)$-st row of the matrix $\mathbf{X}_B\mathbf{R}$ (essentially, the projection of the center of the minimal radius enclosing ball for the original points). Let $\tilde{i} = \arg\max_{i=1\ldots n} \left\| (\mathbf{e}_i - \mathbf{e}_{n+1})^T \mathbf{X}_B \mathbf{R} \right\|_2^2$; then, using the above bound and eqn. (4.12), we get

$$\begin{aligned}
\left\| (\mathbf{e}_{\tilde{i}} - \mathbf{e}_{n+1})^T \mathbf{X}_B \mathbf{R} \right\|_2^2 & \leq (1 + \left\| \mathbf{E}_B \right\|_2) \left\| (\mathbf{e}_{\tilde{i}} - \mathbf{e}_{n+1})^T \mathbf{X}_B \right\|_2^2 \\
& \leq (1 + \left\| \mathbf{E}_B \right\|_2) B^2 .
\end{aligned}$$

Thus, there exists a ball centered at $\mathbf{e}_{n+1}^T \mathbf{X}_B \mathbf{R}$ (the projected center of the minimal radius ball in the original space) with radius at most $\sqrt{1 + \left\| \mathbf{E}_B \right\|_2} B$ that encloses all the projected points. Recall that $\tilde{B}$ is defined as the radius of the minimal radius ball that encloses all points in projected subspace; clearly,

$$\tilde{B}^2 \leq (1 + \left\| \mathbf{E}_B \right\|_2) B^2 .$$

Note that $\left\| \mathbf{E}_B \right\|_2 \leq \epsilon$, which concludes the proof of Theorem 2.

$\diamond$

Combining the results of Theorem 1 and 2 we get,

$$\frac{\tilde{\mathbf{B}}^2}{\tilde{\gamma^{*2}}} \leq \left(\frac{1+\epsilon}{1-\epsilon}\right) \frac{\mathbf{B}^2}{\gamma^{*2}} \approx (1+2\epsilon)\, \frac{\mathbf{B}^2}{\gamma^{*2}}.$$

This shows that the generalization error of SVM in the projected space is comparable to the generalization error of SVM in the original space.

CHAPTER 5
Experiments

5.1 Experimental Setup

We describe experimental evaluations on two real-world datasets, namely a collection of document-term matrices (the TechTC-300 dataset [16]) and a population genetics dataset (the joint Human Genome Diversity Panel or HGDP [17] and the HapMap Phase 3 data [18]) and also on three synthetic datasets. The synthetic datasets and the TechTC-300 dataset contain binary labels, and thus correspond to binary classification tasks. However, the joint HapMap-HGDP dataset contains samples from many world-wide regions and populations, and thus corresponds to a multi-class classification task, and our algorithms perform well here as well.

In our experimental evaluations, we implemented random projections using three different methods: RS, FHT, and CW in MATLAB version 7.13.0.564 (R2011b). We ran the algorithms using the same values of r (the dimension of the projected feature space) for all algorithms, but we varied r across different datasets. We used LIBSVM [19] as our linear SVM solver with default settings. In all cases, we ran our experiments on the original full data (referred to as "full" in the results), as well as on the projected data. We partitioned the data randomly for ten-fold cross-validation in order to estimate out-of-sample error. We repeated this partitioning ten times to get ten ten-fold cross-validation experiments. In order to estimate the effect of the randomness in the construction of the random projection matrices, we repeated our cross-validation experiments ten times using ten different random projection matrices for all datasets. We report in-sample error (ϵ_{in}), out-of-sample error (ϵ_{out}), the time to compute random projections (t_{rp}), the total time needed to both compute random projections *and* run SVMs on the lower-dimensional problem (t_{run}), and the margin (γ). All results are averaged over the ten cross-validation experiments and the ten choices of random projection matrices. For each of the aforementioned quantities, we report both its mean value μ and its standard deviation σ. For the multi-class experiment of Section 5.1.3, we do not report a margin.

Table 5.1: Synthetic data: ϵ_{out} decreases as a function of r in all three families of matrices, using any of the three random projection methods. μ and σ indicate the mean and the standard deviation of ϵ_{out} over ten matrices in each family $D1$, $D2$, and $D3$, ten ten-fold cross-validation experiments, and ten choices of random projection matrices for the three methods that we investigated (a total of 1,000 experiments for each family of matrices).

ϵ_{out}		Projected Dimension r			
		256	512	1024	**full**
D1	CW (μ)	24.08	19.45	16.66	**15.10**
	(σ)	4.52	4.15	3.52	**2.60**
	RS (μ)	24.1.0	19.46	16.36	**15.10**
	(σ)	4.45	3.79	3.22	**2.60**
	FHT (μ)	23.52	19.59	16.67	**15.10**
	(σ)	4.21	4.05	3.37	**2.60**
D2	CW (μ)	25.94	21.07	17.33	**15.44**
	(σ)	4.13	4.16	3.45	**2.54**
	RS (μ)	25.80	20.80	17.47	**15.44**
	(σ)	4.40	3.93	3.42	**2.54**
	FHT (μ)	25.33	21.23	17.58	**15.44**
	(σ)	3.69	4.24	3.53	**2.54**
D3	CW (μ)	27.62	22.97	18.93	**15.83**
	(σ)	3.46	3.22	3.32	**2.00**
	RS (μ)	28.15	23.00	18.72	**15.83**
	(σ)	3.02	3.48	2.78	**2.00**
	FHT (μ)	27.92	23.41	18.73	**15.83**
	(σ)	3.46	3.60	3.02	**2.00**

5.1.1 Synthetic datasets

The synthetic datasets are separable by construction. More specifically, we first constructed a weight vector $\mathbf{w} \in \mathbb{R}^d$, whose entries were selected in i.i.d. trials from a Gaussian distribution $\mathcal{N}(\mu, \sigma)$ of mean μ and standard-deviation σ. We experimented with the following three distributions: $\mathcal{N}(0,1)$, $\mathcal{N}(1,1.5)$, and $\mathcal{N}(2,2)$. Then, we normalized $\mathbf{w}$ to create $\hat{\mathbf{w}} = \mathbf{w}/\left\|\mathbf{w}\right\|_2$. Let $\mathbf{X}_{ij} = \mathcal{N}(0,1)$; then, we set $\mathbf{x}_i$ to be equal to the i-th row of $\mathbf{X}$, while $\mathbf{y}_i = sign\left(\hat{\mathbf{w}}^T\mathbf{x}_i\right)$. We generated families of matrices of different dimensions. More specifically, family $D1$ contained matrices in $\mathbb{R}^{200\times5,000}$; family $D2$ contained matrices in $\mathbb{R}^{250\times10,000}$; and family $D3$ contained

Table 5.2: Synthetic data: γ increases as a function of r in all three families of matrices. See the caption of Table 1 for an explanation of μ and σ.

γ		Projected Dimension r			
		256	512	1024	**full**
D1	CW (μ)	5.72	6.67	7.16	**7.74**
	(σ)	0.58	0.58	0.59	**0.59**
	RS (μ)	5.73	6.66	7.18	**7.74**
	(σ)	0.57	0.55	0.55	**0.59**
	FHT (μ)	5.76	6.64	7.15	**7.74**
	(σ)	0.56	0.58	0.56	**0.59**
D2	CW (μ)	6.62	8.09	8.88	**9.78**
	(σ)	0.64	0.62	0.59	**0.66**
	RS (μ)	6.65	8.10	8.88	**9.78**
	(σ)	0.64	0.60	0.63	**0.66**
	FHT (μ)	6.66	8.06	8.84	**9.78**
	(σ)	0.63	0.65	0.63	**0.66**
D3	CW (μ)	7.69	9.84	11.07	**12.46**
	(σ)	0.67	0.60	0.71	**0.69**
	RS (μ)	7.61	9.85	11.05	**12.46**
	(σ)	0.59	0.6212	0.62	**0.69**
	FHT (μ)	7.63	9.83	11.11	**12.46**
	(σ)	0.67	0.64	0.64	**0.69**

matrices in $\mathbb{R}^{300\times 20,000}$. We generated ten datasets for each of the families $D1$, $D2$, and $D3$, and we report average results over the ten datasets. We set r to 256, 512, and 1024 and set C to 1,000 in LIBSVM for all the experiments. Tables 5.1 and 5.2 show ϵ_{out} and γ for the three datasets $D1$, $D2$, and $D3$. ϵ_{in} is zero for all three data families. As expected, ϵ_{out} and γ improve as r grows for all three random projection methods. Also, the time needed to compute random projections is very small compared to the time needed to run SVMs on the projected data. Figure 5.1 shows the combined running time of random projections and SVMs, which is nearly the same for all three random projection methods. It is obvious that this combined running time is much smaller that the time needed to run SVMs on the full dataset (with out any dimensionality reduction). For instance, for $r = 1024$, t_{run} for $D1$, $D2$, and $D3$ is (respectively) 6, 9, and 25 times smaller than t_{run} on the full-data.

5.1.2 The TechTC-300 dataset

For our first real dataset, we use the TechTC-300 data, consisting of a family of 295 document-term data matrices. The TechTC-300 dataset comes from the Open Directory Project (ODP), which is a large, comprehensive directory of the web, maintained by volunteer editors. Each matrix in the TechTC-300 dataset contains a pair of categories from the ODP. Each category corresponds to a label, and thus the resulting classification task is binary. The documents that are collected from the union of all the subcategories within each category are represented in the bag-of-words model, with the words constituting the features of the data [16]. Each data matrix consists of 150-280 documents (the rows of the data matrix $\mathbf{X}$), and each document is described with respect to 10,000-40,000 words (features, columns of the matrix $\mathbf{X}$). Thus, TechTC-300 provides a diverse collection of data sets for a systematic study of the performance of the SVM on the projected versus full data.

We set the parameter C to 500 in LIBSVM for all 295 document-term matrices and set r to 128, 256, and 512. We use a lower value of C than for the other data sets for computational reasons: larger C is less efficient. We note that our classification accuracy is slightly worse (on the full data) than the accuracy presented in Section 4.4 of [16], because we did not fine-tune the SVM parameters as they did, since that is not the focus of this study. For every dataset and every value of r we tried, the in-sample error on the projected data matched the in-sample error on the full data. We thus focus on ϵ_{out}, the margin γ, the time needed to compute random projections t_{rp}, and the total running time t_{run}. We report our results averaged over 295 data matrices. Table 5.3 shows the behavior of these parameters for different choices of r.

As expected, ϵ_{out} and the margin γ improve as r increases, and they are nearly identical for all three random projection methods. The time needed to compute random projections is smallest for CW, followed by RS and FHT. As a matter of fact, t_{rp} for CW is ten to 20 times faster than RS and FHT for different values of r. This is predicted by the theory in [4], since CW is optimized to take advantage of input sparsity. However, this advantage is lost when SVMs are applied on the dimensionally-reduced data. Indeed, the combined running time t_{run} is fastest for

Table 5.3: Results on the Techtc300 dataset, averaged over 295 data matrices using three different random projection methods. The table shows how ϵ_{out}, γ, t_{rp} (in seconds), and t_{run} (in seconds) depend on r. μ and σ indicate the mean and the standard deviation of each quantity over 295 matrices, ten ten-fold cross-validation experiments, and ten choices of random projection matrices for the three methods that we investigated.

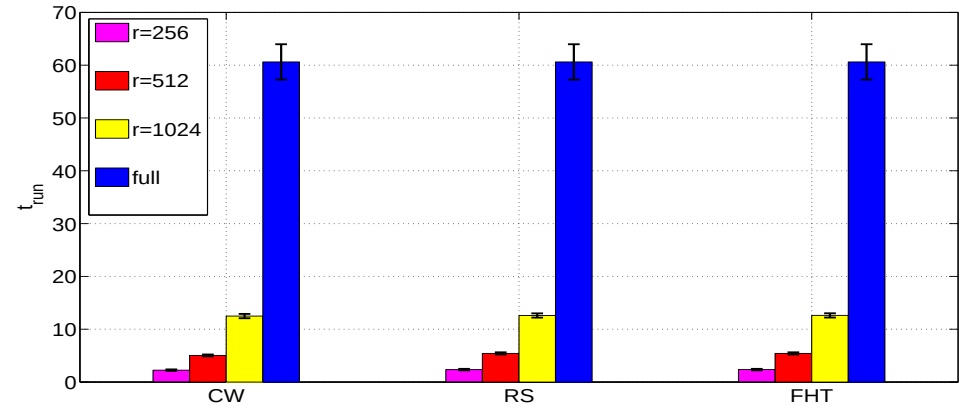
		Projected Dimension r			
		128	256	512	**full**
$\epsilon_{\mathbf{out}}$	CW(μ)	24.63	22.84	21.26	**17.35**
	(σ)	10.57	10.37	10.17	**9.45**
	RS(μ)	24.58	22.90	21.38	**17.35**
	(σ)	10.57	10.39	10.23	**9.45**
	FHT (μ)	24.63	22.93	21.35	**17.35**
	(σ)	10.66	10.39	10.2	**9.45**
γ	CW (μ)	1.66	1.88	1.99	**2.09**
	(σ)	3.68	3.79	3.92	**4.00**
	RS (μ)	1.66	1.88	1.99	**2.09**
	(σ)	3.65	3.80	3.91	**4.00**
	FHT (μ)	1.66	1.88	1.98	**2.09**
	(σ)	3.65	3.81	3.88	**4.00**
$\mathbf{t_{rp}}$	CW (μ)	0.0046	0.0059	0.0075	$--$
	(σ)	0.0019	0.0026	0.0033	$--$
	RS (μ)	0.0429	0.0855	0.1719	$--$
	(σ)	0.0178	0.0356	0.072	$--$
	FHT (μ)	0.0443	0.0882	0.1764	$--$
	(σ)	0.0206	0.0413	0.0825	$--$
$\mathbf{t_{run}}$	CW (μ)	1.23	2.22	4.63	**4.85**
	(σ)	0.87	0.93	1.93	**2.12**
	RS (μ)	0.99	1.53	3.02	**4.85**
	(σ)	0.97	0.59	1.12	**2.12**
	FHT (μ)	0.95	1.46	2.83	**4.85**
	(σ)	0.96	0.55	1.02	**2.12**

FHT, followed by RS and CW. In all cases, the total running time is smaller than the SVM running time on full dataset. For example, in the case of FHT, setting $r = 512$ achieves a running time t_{run} which is about 70% faster than running SVMs on the full dataset; ϵ_{out} increases by less than 4%.

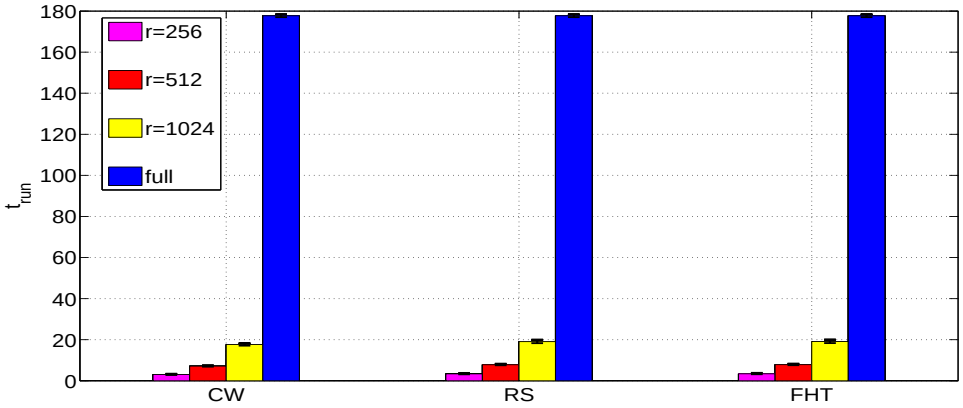
5.1.3 The HapMap-HGDP dataset

Predicting ancestry of individuals using a set of genetic markers is a well-studied classification problem. We use a population genetics dataset from the Human Genome Diversity Panel (HGDP) and the HapMap Phase 3 dataset (see [18]) for details), in order to classify individuals into broad geographic regions, as well as into (finer-scale) populations. We study a total of 2,250 individuals from approximately 50 populations and five broad geographic regions. The features in this dataset correspond to $492,516$ Single Nucleotide Polymorphisms (SNPs), which are well-known biallelic loci of genetic variation across the human genome. Each entry in the resulting $2,250 \times 492,516$ matrix is set to $+1$ (homozygotic in one allele), -1 (homozygotic in the other allele), or 0 (heterozygotic), depending on the genotype of the respective SNP for a particular sample. Missing entries were filled in with -1, $+1$, or 0, with probability $1/3$. Each sample has a known population and region of origin, which constitute its label.

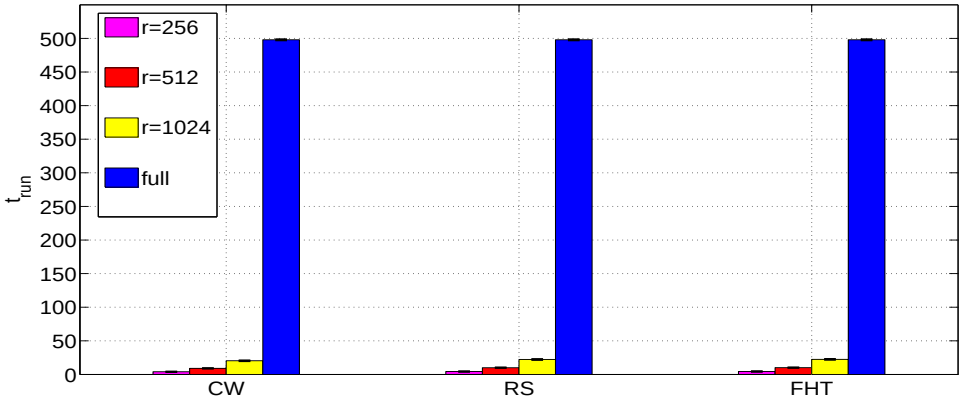
We set r to 256, 512, 1024, and 2048 in our experiments. Since this task is a multi-class classification problem, we used LIBSVM's one-against-one technique for classification. We ran two sets of experiments: in the first set, the classification problem is to assign samples to broad regions of origin, while in the second experiment, our goal is to classify samples into (fine-scale) populations. We set C to 1,000 in LIBSVM for all the experiments. The in-sample error is zero in all cases. Figure 5.2 shows the out-of-sample error for regions and populations classification, which are nearly identical for all three random projection methods. For regional classification, we estimated ϵ_{out} to be close to 2%, and for population-level classification, ϵ_{out} is close to 20%. This experiment strongly supports the computational benefits of our methods in terms of main memory. $\mathbf{X}$ is $2,250 \times 492,516$, which is too large to fit into memory in order to run SVMs. Figure 5.3 shows that the combined running time for three different random projection methods are nearly identical for both regions and population classification tasks. However, the time needed to compute the random projections is different from one method to the next. FHT is fastest, followed by RS and CW. In this particular case, the input matrix in quite dense, and CW seems to be outperformed by the other two methods.



D1

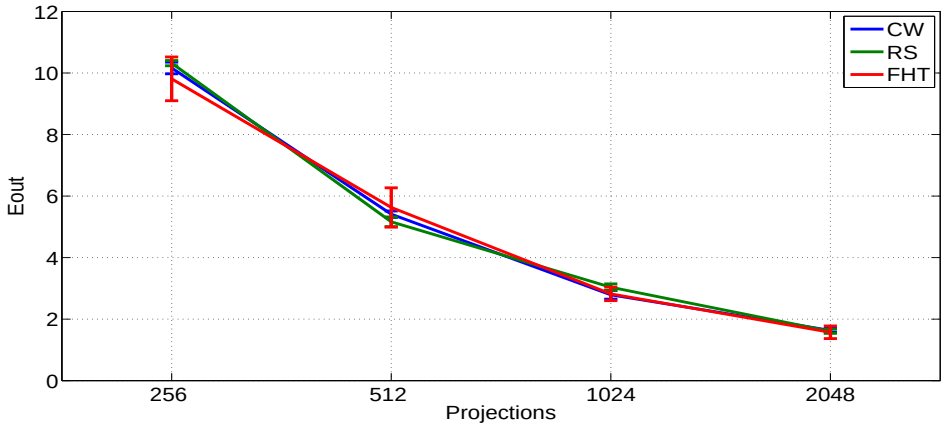


D2

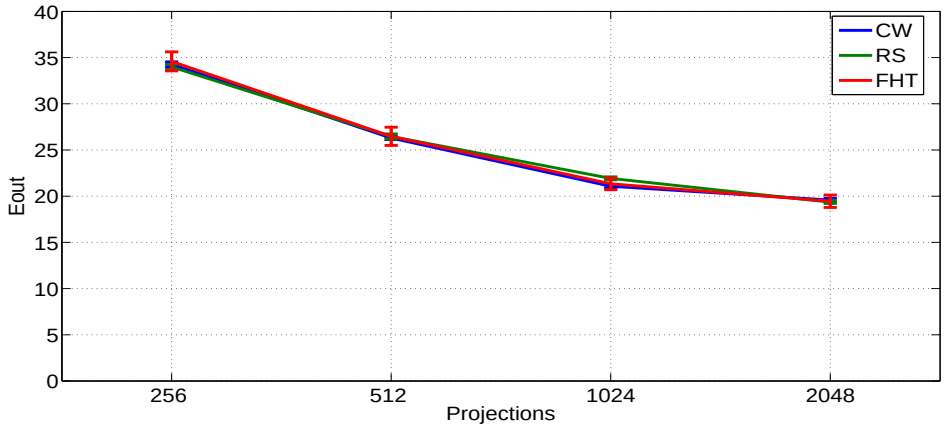


D3

Figure 5.1: Total (average) running times, in seconds, of random projections *and* SVMs on the lower-dimensional data for each of the three families of synthetic data. Vertical bars indicate the, relatively small, standard deviation (see the caption of Table 1).

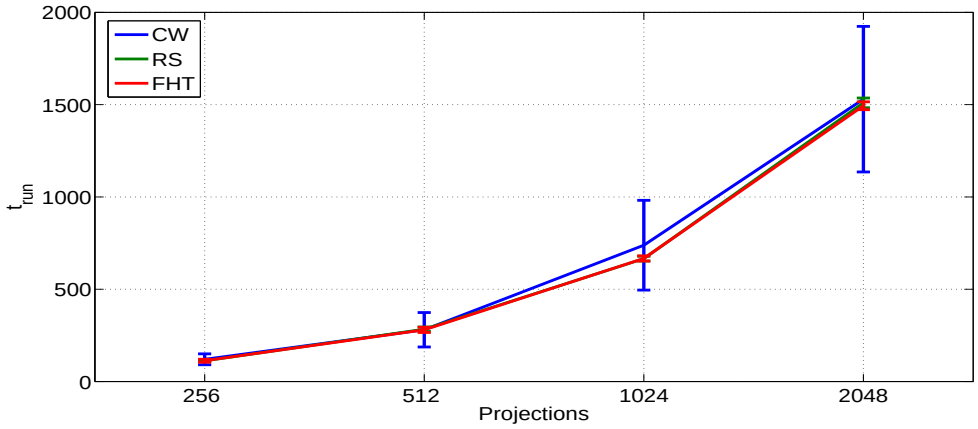


Regional classification

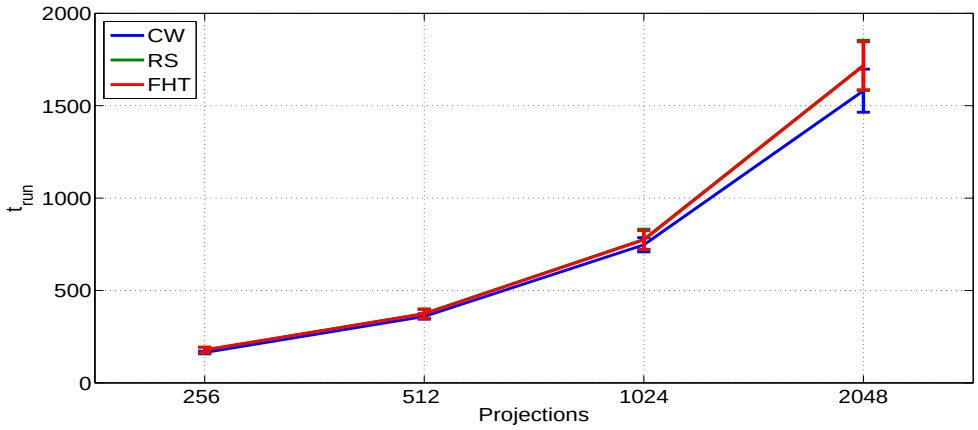


Population-level classification

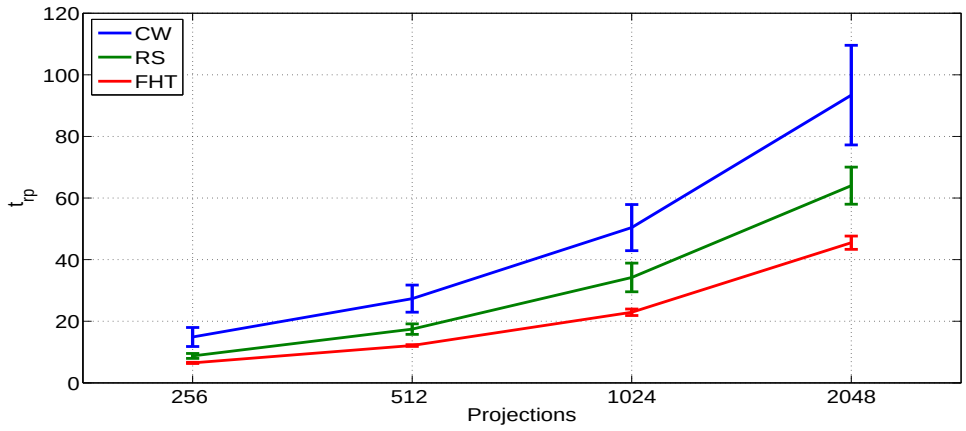
Figure 5.2: **ϵ_{out} as a function of r in the Hapmap-HGDP dataset for three different random projection methods and two different classification tasks. Vertical bars indicate the standard-deviation over the ten ten-fold cross-validation experiments and the ten choices of the random projection matrices for each of the three methods.**



Total running time: regional classification



Total running time: population-level classification



Time needed to compute random projections

Figure 5.3: Total running time in seconds (random projections *and* SVM classification on the dimensionally-reduced data) for Hapmap-HGDP dataset for three different projection methods using both regional and population-level labels. Notice that the time needed to compute random projection is independent of the classification labels. Vertical bars indicate standard-deviation, as in Figure 5.2.

CHAPTER 6
Conclusions and open problems

We present theoretical and empirical results indicating that random projections are a useful dimensionality reduction technique for SVM classification problems that handle sparse or dense data in high-dimensional feature spaces. Our theory predicts that the dimensionality of the projected space (denoted by r) has to grow essentially *linearly* (up to logarithmic factors) in ρ (the rank of the data matrix) in order to achieve relative error approximations to the margin and the radius of the minimum ball enclosing the data. Such relative-error approximations imply excellent generalization performance. However, our experiments show that considerably smaller values for r (e.g., in the case of the TechTC data, setting r to 1/70-th of all available features) results in classification that is essentially as accurate as running SVMs on all available features, despite the fact that the matrices have full numerical rank. This seems to imply that our theoretical results can be improved. FHT and RS work well on dense data while CW is an excellent choice for sparse data, as indicated by our experiments. However, this solid performance of CW (which is predicted by the theoretical bounds of [4]) comes at a cost, at least according to our experimental evaluation: solving the SVM optimization problem on the resulting low-dimensional dataset is quite expensive, and, as a result, the total running time of the CW method is eventually higher than that of FHT and RS. This seems to indicate that more research is necessary in terms of random projection methods that are both fast (e.g., can be applied on the input matrix in time that is proportional to the number of non-zero entries in the matrix), but also result in low-dimensional data matrices that are "friendly" (e.g., correspond to well-structured problem instances) for SVM solvers. Understanding this aspect of random projection matrices is important and it has not been investigated at all in existing literature.

REFERENCES

[1] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and other kernel-based learning methods*, 1st ed. New York: Cambridge University Press, 2000.

[2] V. N. Vapnik, *Statistical Learning Theory*, 1st ed. New York: John Wiley & Sons, Inc., 1998.

[3] V. Vapnik and A. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," *Theory of Probability and its Applicat.*, vol. 16, pp. 264–280, 1971.

[4] K. Clarkson and D. Woodruff, "Low rank approximation and regression in input sparsity time," *CoRR*, vol. abs/1207.6365, 2012, Retrieved : 09/10/2012. [Online]. Available: http://arxiv.org/abs/1207.6365

[5] S. Krishnan, C. Bhattacharya, and R. Hariharan, "A randomized algorithm for large scale support vector learning," in *Adv. in 20th Neural Inform. Process. Syst.*, 2008, pp. 793–800.

[6] J. Balcazar, Y. Dai, and O. Watanabe, "A random sampling technique for training support vector machines," in *Proc. of the 12th Int. Conf. on Algorithmic Learning Theory*, 2001, pp. 119–134.

[7] Q. Shi, C. Shen, R. Hill, and A. Hengel, "Is margin preserved after random projection?" in *Proc. of the 29th Int. Conf. on Machine Learning*, 2012, pp. 591–598.

[8] W. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space," *Contemporary Math.*, vol. 1-1, no. 26, pp. 189–206, 1984.

[9] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proc. of the 30th Annu. ACM Symp. on Theory of Computing*, 1998, pp. 604–613.

[10] S. Dasgupta and A. Gupta, "An elementary proof of a theorem of Johnson and Lindenstrauss," *Random Structures and Algorithms*, vol. 22, no. 1, pp. 60–65, 2003.

[11] D. Achlioptas, "Database-friendly random projections: Johnson-Lindenstrauss with binary coins," *J. of Comput. and Syst. Sci.*, vol. 66, no. 4, pp. 671–687, 2003.

[12] N. Ailon and B. Chazelle, "Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform," in *Proc. of the 38th Annu. ACM Symp. on Theory of Computing*, 2006, pp. 557–563.

[13] N. Ailon and E. Liberty, "Fast dimension reduction using Rademacher series on dual BCH codes," in *Proc. of the 19th Annu. ACM-SIAM Symp. on Discrete Algorithms*, 2008, pp. 1–9.

[14] P. Drineas, M. Mahoney, S. Muthukrishnan, and T. Sarlos, "Faster least squares approximation," *Numerische. Math.*, vol. 117, no. 2, pp. 219–249, 2011.

[15] A. Magen and A. Zouzias, "Low rank matrix-valued Chernoff bounds and approximate matrix multiplication," in *Proc. of the 22nd Annu. ACM-SIAM Symp. on Discrete Algorithms*, 2011, pp. 1422–1436.

[16] D. Davidov, E. Gabrilovich, and S. Markovitch, "Parameterized generation of labeled datasets for text categorization based on a hierarchical directory," in *Proc. of the 27th Annu. Int. ACM SIGIR Conf. on Research and Develop. in Inform. Retrieval*, 2004, pp. 250–257, Retrieved : 06/01/2012. [Online]. Available: http://techtc.cs.technion.ac.il/techtc300/techtc300.html

[17] J. Li, D. Absher, H. Tang, A. Southwick, A. Casto, S. Ramachandran, H. Cann, G. Barsh, M. Feldman, L. Cavalli-Sforza, and R. Myers, "Worldwide human relationships inferred from genome-wide patterns of variation," *Sci.*, vol. 319, no. 5866, pp. 1100–1104, 2008.

[18] P. Paschou, J. Lewis, A. Javed, and P. Drineas, "Ancestry informative markers for fine-scale individual assignment to worldwide populations." *J. of Medical Genetics*, vol. 47, no. 12, pp. 835–47, 2010.

[19] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. on Intell. Syst. and Tech.*, vol. 2, pp. 27:1–27:27, 2011, Retrieved : 06/01/2012. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm

APPENDIX A
Theroems

The following is Theorem 2.1 of [13] in connection with the running time of Fast Hadamard Transform.

Theorem 3. *[13] For any matrix A of size $d \times r$ with $r < d$, the mapping $x \to \mathbf{A}x$ can be computed in time $O(d \log r)$.*

Below are Theorem 4 and Lemma 3 of [14] in connection with Lemma 1.

Lemma 4. *Let U be a $n \times d$ orthogonal matrix and let the product $\mathbf{HD}$ be the $n \times n$ Randomized Hadamard Transform. Then with probability at least 0.95,*

$$\left\|(\mathbf{HDU})_{(i)}\right\|_2^2 \leq \frac{2d \ln(40nd)}{n}$$

for all $i = 1, \ldots, d$ with probability at least .95.

Theorem 4. *Let $A \in \mathbb{R}^{m \times n}$ with $\|A\|_2 \leq 1$. Construct C using the* Exactly(c) *algorithm and let the sampling probabilities p_i satisfy*

$$p_i \geq \beta \frac{\left\|A^{(i)}\right\|_2^2}{\|A\|_F^2} \tag{A.1}$$

for all $i \in [n]$ for some constant $\beta \in (0, 1]$. Let $\epsilon \in (0, 1)$ be an accuracy parameter and assume $\|A\|_F^2 \geq 1/24$. If

$$c \geq \frac{96 \|A\|_F^2}{\beta \epsilon^2} \ln\left(\frac{96 \|A\|_F^2}{\beta \epsilon^2 \sqrt{\delta}}\right) \tag{A.2}$$

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

$$\left\|AA^T - CC^T\right\|_2 \leq \epsilon.$$

The following is Theorem 3.1 of [15] in connection with the proof of Lemma 2.

Theorem 5. *Fix* $\epsilon \in (0, \frac{1}{2}]$ *and let* $\mathbf{A} \in \mathbb{R}^{n\times m}$, $\mathbf{B} \in \mathbb{R}^{n\times p}$ *both having stable rank at most* ρ. *Let* $\mathbf{R}$ *be a* $t \times n$ *random sign matrix rescaled by* $1/\sqrt{t}$. *If* $t = O\left(\rho\epsilon^{-2}\log\left(m+p\right)\right)$, *then* $\left\|A^TR^TRB - A^TB\right\|_2 \leq \epsilon \left\|A\right\|_2 \left\|B\right\|_2$ *with probability* $\left(1 - 1/poly(\rho)\right)$.

APPENDIX B
Plots of Synthetic Dataset

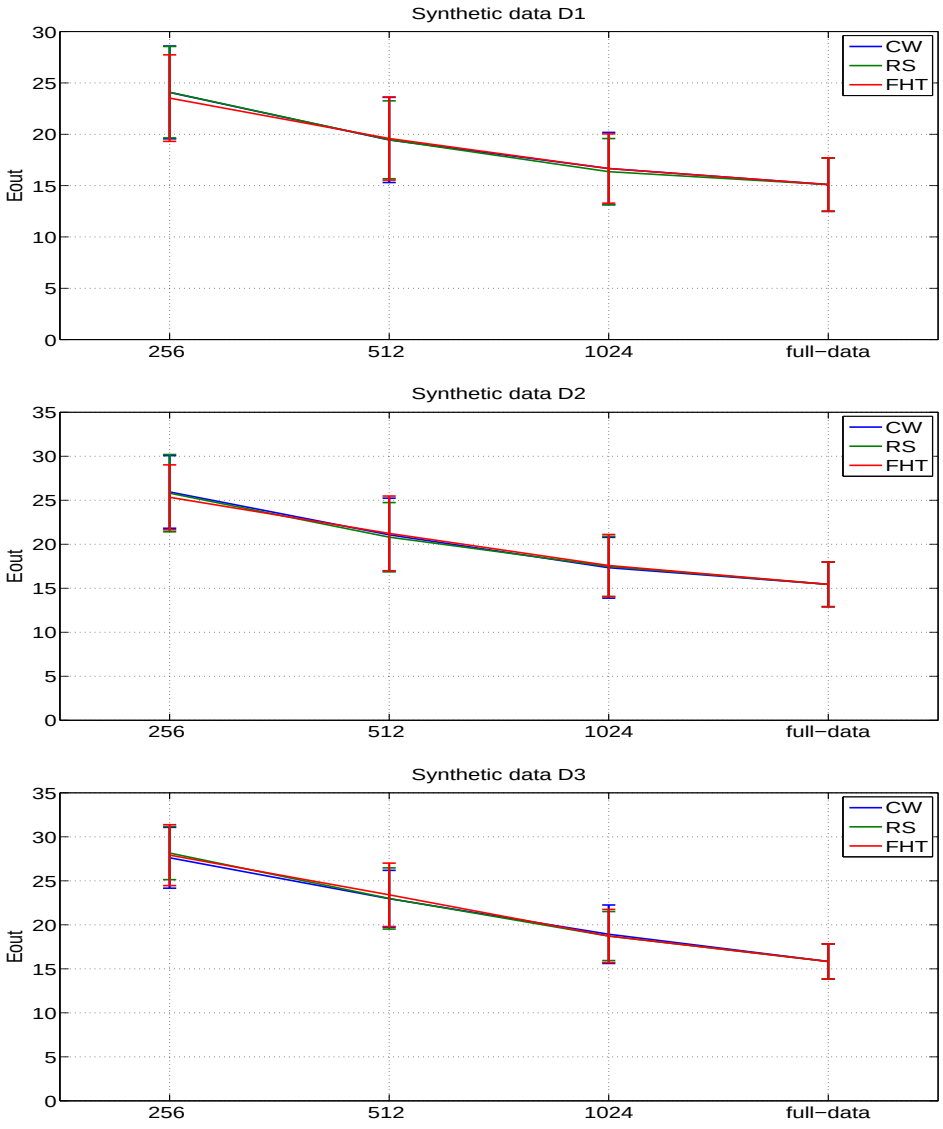


Figure B.1: Performance of out-of-sample error as a function of r on different synthetic datasets using three different random projection methods. Vertical bars represent standard-deviation over 10 ten-fold cross-validation experiments and ten choices of random projection matrices for each of the three methods.

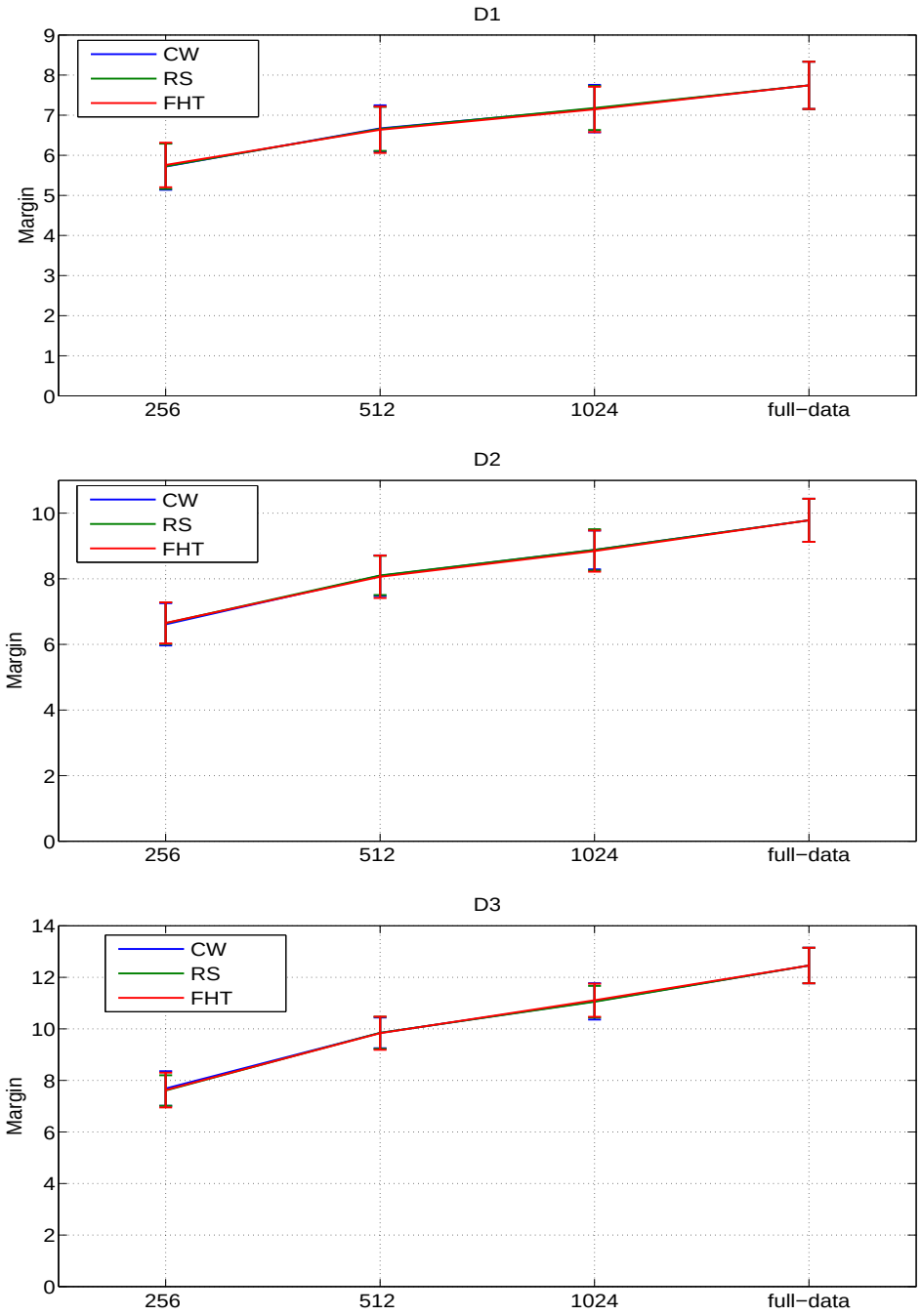


Figure B.2: Performance of margin on different datasets using different methods of projection. Vertical bars represent standard-deviation over 10 ten-fold cross-validation experiments and ten choices of random projection matrices for each of the three methods.

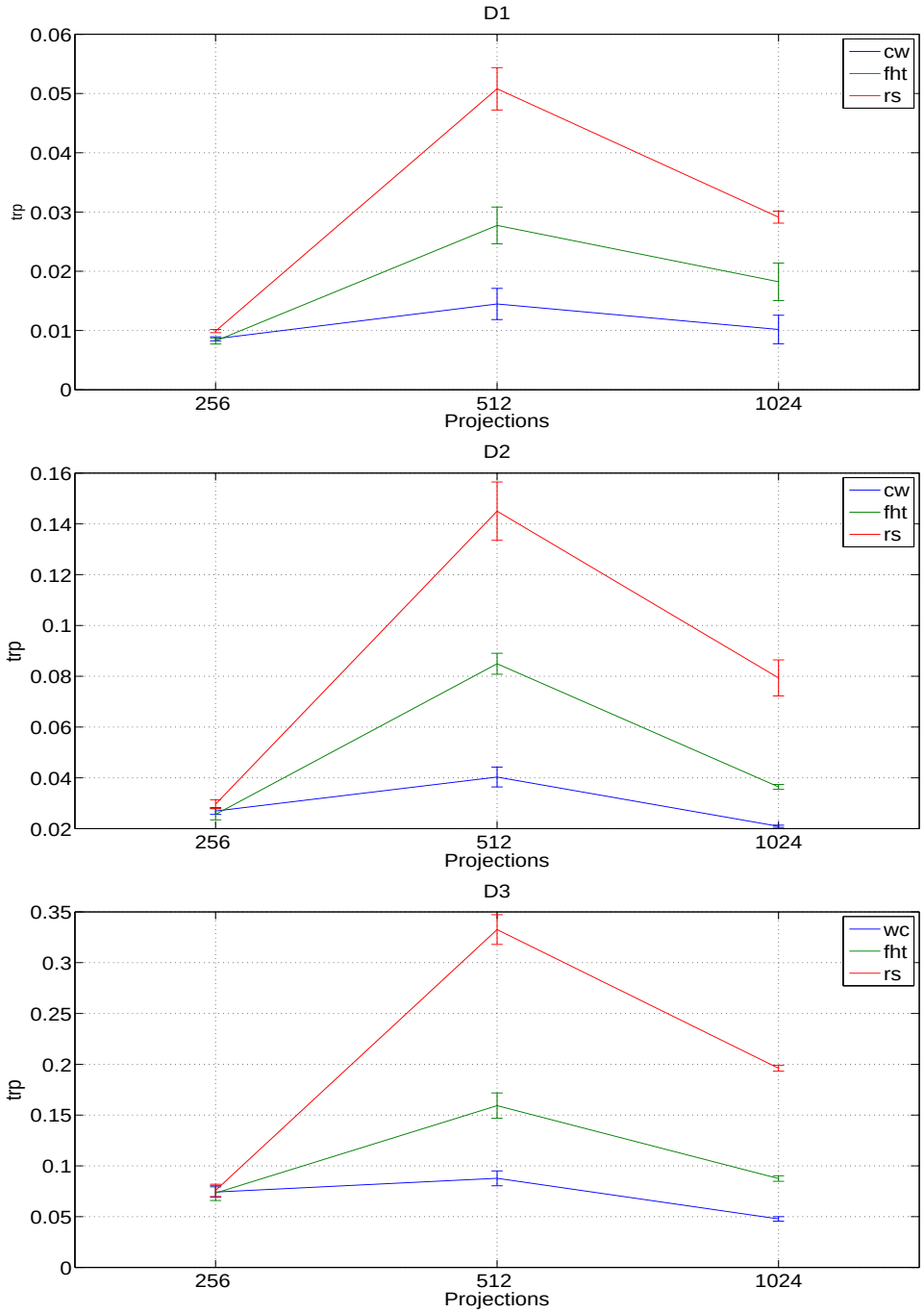


Figure B.3: The above plots show time to compute random projections (in seconds) for three datasets. Vertical bars represent standard-deviation over 10 ten-fold cross-validation experiments and ten choices of random projection matrices for each of the three methods. t_{rp} is almost the same for all the synthetic datasets.

APPENDIX C
Plots of TechTC-300 Dataset

In the following plots related to TechTC-300, the results are sorted according to the increasing order of eout-full.

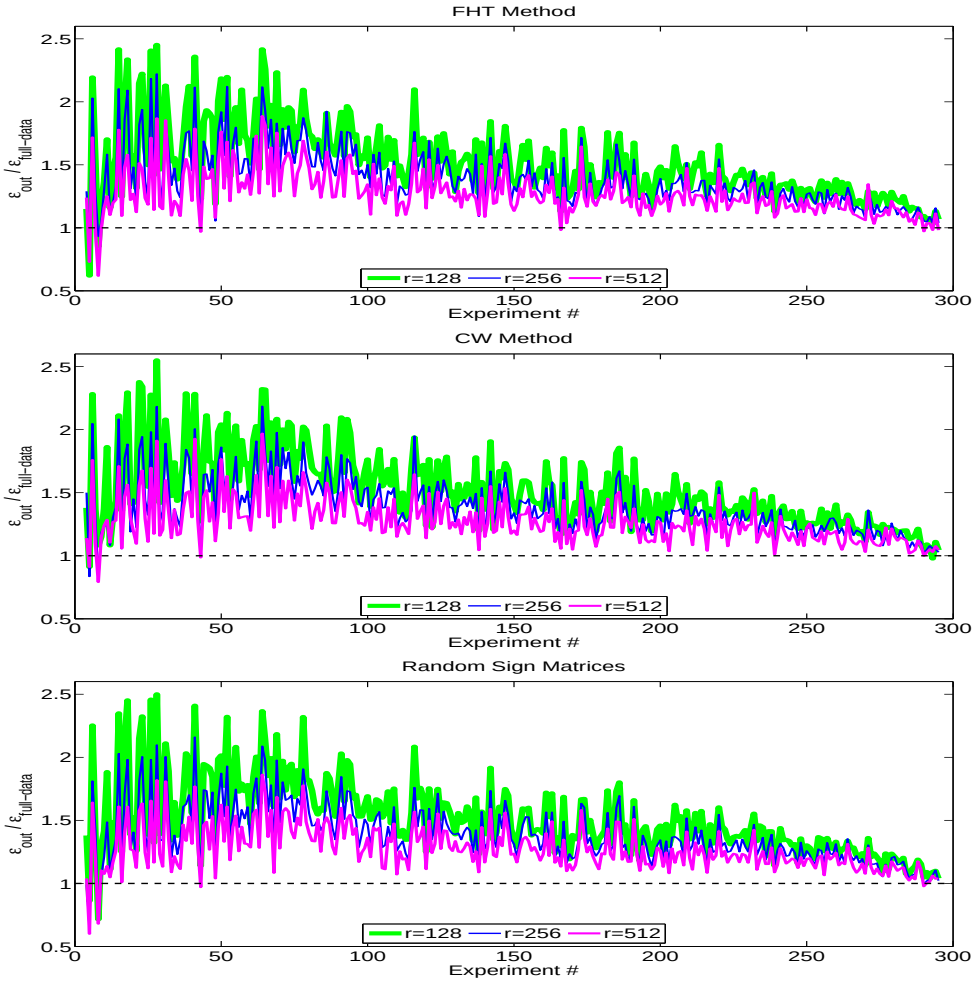


Figure C.1: The above plots show the mean gain in out-of-sample error (eout-gain) as a function of r for three different random projection methods in the TechTC-300 dataset. The results show eout-gain averaged over 10 ten-fold cross-validation experiments and ten choices of random projection matrices for the three methods we investigated. Note that in some of the experiments, random projections beats the results of full-data.

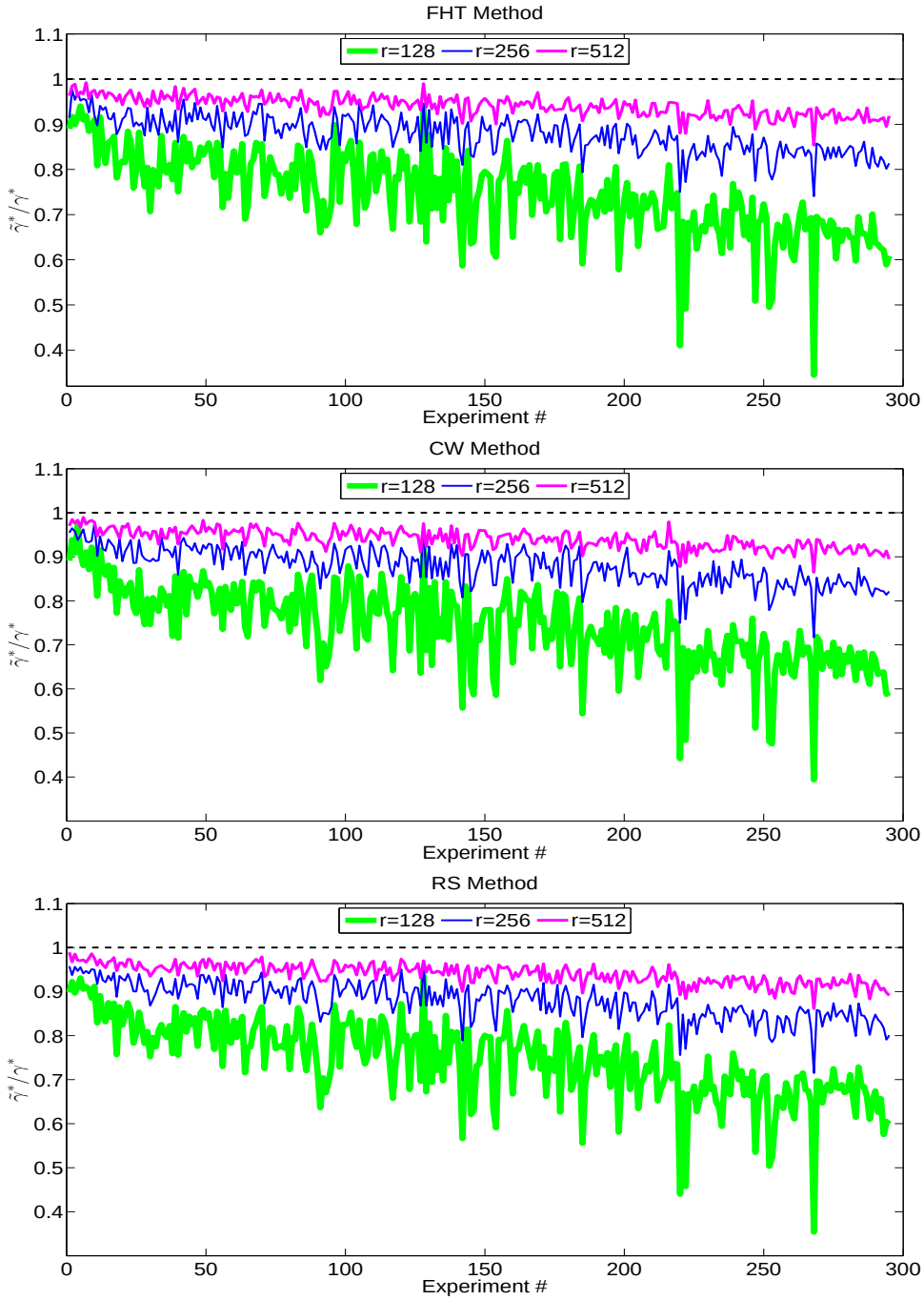


Figure C.2: The above plots show the mean gain in margin as a function of r for three different random projection methods in the TechTC-300 dataset. The results show margin-gain averaged over 10 ten-fold cross-validation experiments and ten choices of random projection matrices for the three methods we investigated.

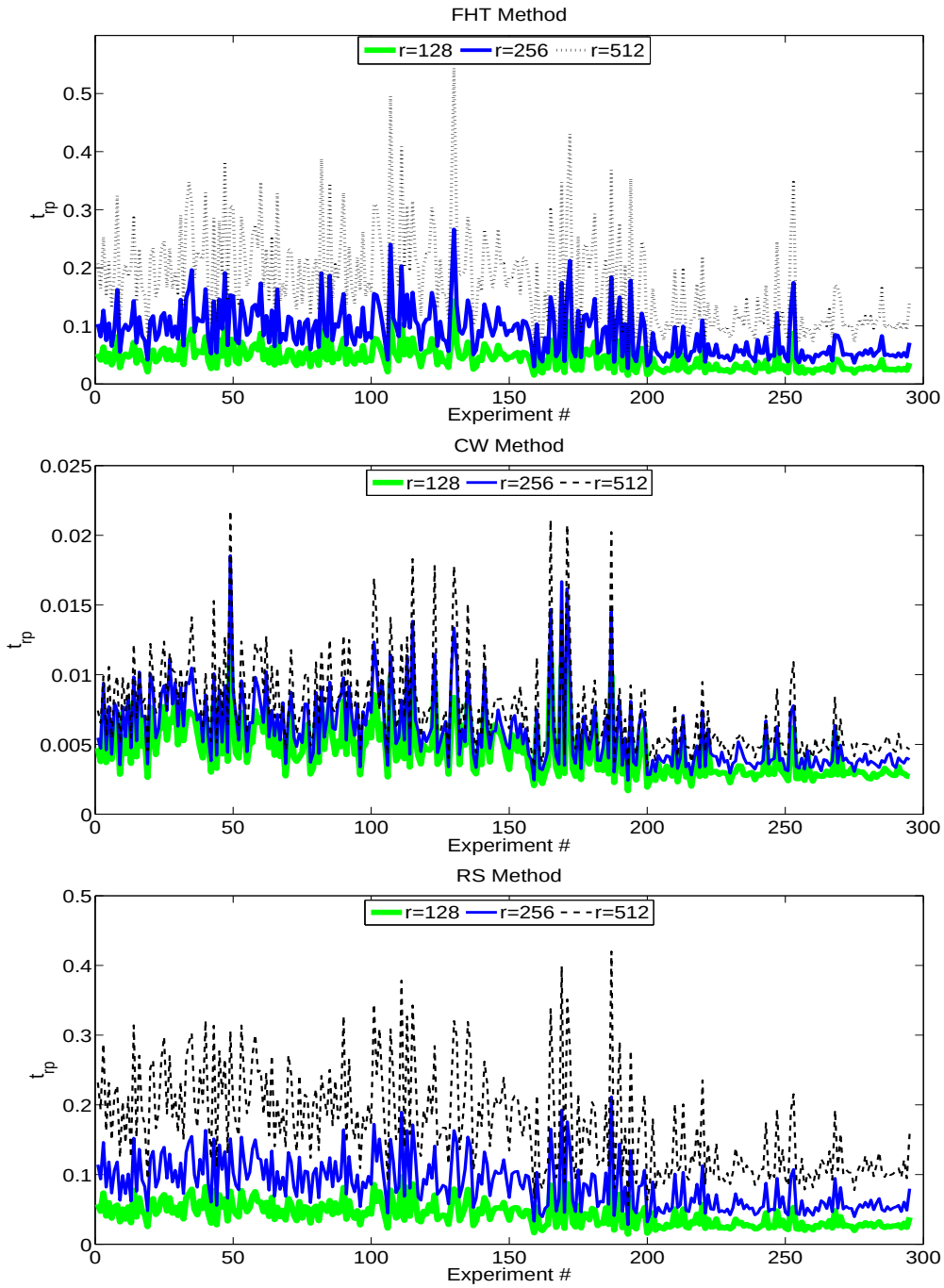


Figure C.3: The above plots show the mean time to compute random projection (in seconds) as a function of r for three different random projection methods in the TechTC-300 dataset. The results show trp averaged over 10 ten-fold cross-validation experiments and ten choices of random projection matrices for the three methods we investigated.

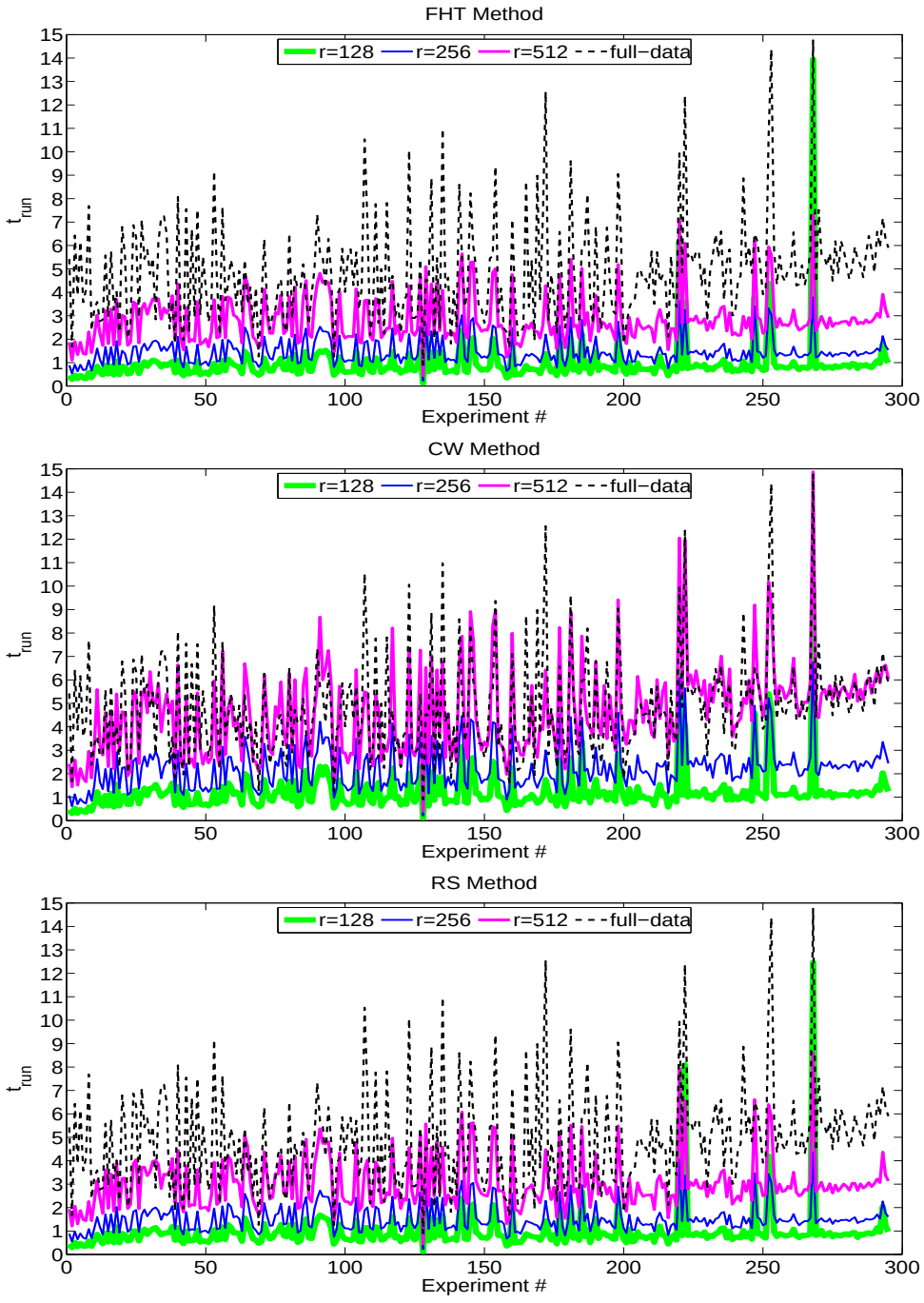


Figure C.4: The above plots show the mean total running time (in seconds) as a function of r for three different random projection methods in the TechTC-300 dataset.