RICE UNIVERSITY

oASIS: Adaptive Column Sampling for Kernel Matrix Approximation

by

Raajen Patel

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

APPROVED, THESIS COMMITTEE:

Dr. Richard Baraniuk, Chair
Victor E. Cameron Professor of Electrical
and Computer Engineering

Dr. Ashok Veeraraghavan
Assistant Professor of Electrical and
Computer Engineering

Dr. Danny C. Sorensen
Noah Harding Professor of Computational
and Applied Mathematics

Houston, Texas

April, 2015
ABSTRACT

oASIS: Adaptive Column Sampling for Kernel Matrix Approximation

by

Raajen Patel

Kernel or similarity matrices are essential for many state-of-the-art approaches to classification, clustering, and dimensionality reduction. For large datasets, the cost of forming and factoring such kernel matrices becomes intractable. To address this challenge, we introduce a new adaptive sampling algorithm called Accelerated Sequential Incoherence Selection (oASIS) that samples columns without explicitly computing the entire kernel matrix. We provide conditions under which oASIS is guaranteed to exactly recover the kernel matrix with an optimal number of columns selected. Numerical experiments on both synthetic and real-world datasets demonstrate that oASIS achieves performance comparable to state-of-the-art adaptive sampling methods at a fraction of the computational cost. The low runtime complexity of oASIS and its low memory footprint enable the solution of large problems that are simply intractable using other adaptive methods.
Acknowledgments

Thanks to my advisor, Dr. Richard Baraniuk, for his support, aphorisms, and good humor. Thanks also to Drs. Eva Dyer, Thomas Goldstein, Azalia Mirhoseini, and Mohammed Golbabae for their tireless effort, insight, and comments. Thanks to my committee, Dr. Ashok Veeraraghavan and Dr. Danny Sorensen, for their helpful comments and input. Thanks to Heather Seeba and Ebrahim Songhori for computing backend help, and to Denise Fly for administrative backup.

Most of all I would like to thank my family, my friends, and my wife, for reasons too numerous to sufficiently record.
Contents

Abstract ii
Acknowledgments iii
List of Illustrations vi
List of Tables vii

1 Introduction 1
1.1 Contributions ..................................................... 2

2 Background 6
2.1 Notation ............................................................. 6
2.2 Kernel Matrices and their Applications ...................... 6
2.3 The Nyström Method ............................................. 7
2.4 Column Sampling Methods ..................................... 9
2.4.1 Uniform Random Sampling ................................ 9
2.4.2 Non-deterministic Adaptive Sampling .................... 9
2.4.3 Deterministic Adaptive Sampling ......................... 10
2.4.4 Representative Point Sampling with K-means Clustering .... 10
2.5 Finding Low-Dimensional Structure ......................... 11

3 Accelerated Sequential Incoherence Selection (oASIS) 13
3.1 Sequential Incoherence Selection (SIS) ...................... 13
3.2 oASIS ............................................................... 15
3.3 Parallelized oASIS ............................................... 17

4 Properties and Applications of oASIS 20
4.1 Theoretical Guarantees of oASIS ................................. 20
4.1.1 Independent Selection Property of oASIS ................ 20
4.1.2 Exact Matrix Recovery ...................................... 21
4.1.3 Guarantees of oASIS when $G$ is a Gram Matrix ........... 22
4.1.4 Comparison with other theory ................................ 24
4.2 Complexity of oASIS .............................................. 25
4.3 Complexity of oASIS-P ........................................... 26

5 Numerical Experiments 27
5.1 Full Kernel Matrices ............................................. 29
5.2 Implicit Kernel Matrices ........................................ 31
5.3 Implicit Kernel Matrices with oASIS-P .................. 34
5.4 Discussion of Results ........................................... 36

6 Conclusion ......................................................... 40
Illustrations

3.1 The oASIS algorithm. Note that $G$ need not be precomputed.  
3.2 oASIS-P, for data matrices too large for a single core.

4.1 For a PSD matrix $G = Z^T Z$, with the dataset $Z$ as in (a), we 
compare approximation errors for $\tilde{G}$ formed using oASIS or uniform 
random sampling in (b), and the rank of $\tilde{G}$ with either method in (c).
We terminate trials at exact recovery. See Section 4.1.4 for details.

5.1 Nyström Approximation Error Curves and Run Time Comparison.
See Section 5 for details. Note that oASIS is accurate and scales well 
to large problem sizes due to its low runtime complexity (see Section 
4.2).

5.2 Nyström Error curves and number of samples based on time. See 
Section 5 for details. Note that oASIS is accurate and scales well to 
large problem sizes due to its low runtime complexity (see Section 4.2).
Tables

5.1 Error and Runtime Results for Gaussian (top) and Diffusion (bottom) Kernel Matrices. ........................ 32
5.2 Error and Runtime Results for Implicit Kernel Matrices. .................. 33
5.3 Error and Runtime Results for Parallelized Implicit Kernel Matrices. 33
Chapter 1

Introduction

Many machine learning and data analysis frameworks for classification, clustering, and dimensionality reduction require the formation of kernel matrices that contain the pairwise “similarities” or distances between signals in a collection of data. For instance, kernel methods for classification [1], nonlinear dimensionality reduction [2, 3], and spectral clustering methods [4] all require computing and storing an $n \times n$ kernel matrix, where $n$ is the number of examples (points) in the dataset.

As the size of the dataset grows, the computation and storage of kernel matrices becomes increasingly difficult. For instance, explicitly storing a matrix with dimension $n = 10^5$ using double precision requires 80 gigabytes of memory. To extend kernel-based methods to extremely large $n$, a host of methods have focused on sampling a small subset of columns from the kernel matrix and then computing a low rank approximation of the matrix using the Nyström method [5]. Nyström methods are one example of a general approach to computing low-rank matrix approximation from a subset of rows and/or columns of the matrix [6]. Choosing a relevant subset is broadly referred to as column subset selection (CSS) methods and have been applied successfully in applications ranging from image segmentation [7] to genomic analysis [8] and matrix factorization [9].
The success of CSS-based approaches for matrix approximation depends on the columns chosen to approximate the range of the matrix. Intuitively, uniform random sampling of the columns will provide a good approximation when columns of the kernel matrix are uniformly distributed. However, when the underlying data are non-uniformly distributed or even clustered, uniform sampling is extremely inefficient. In these settings, it has been shown in both theory [10] and practice [11] that adaptive sampling methods provide accurate approximations of low-rank matrices with far fewer samples than uniform random sampling.

The downside of current adaptive methods is their computational burden. Adaptive methods do not scale well to large problem sizes for two reasons. First, existing adaptive methods inspect the entire kernel matrix to determine which columns to select. For extremely large datasets, both forming and storing an explicit representation of the kernel matrix is intractable. Second, existing adaptive approaches require dense $n \times n$ matrix computations, even for sparse matrices that are otherwise easy to store because they have relatively few non-zeros elements. For these reasons, adaptive methods cannot be applied to extremely large collections of data [12].

1.1 Contributions

In general, it is not possible to determine which columns to select from a kernel matrix without explicitly forming all of the candidate columns (i.e. computing the kernel measure between one data point and the rest of the dataset, for all data points).
However, in the case of symmetric kernel matrices, a small sample of columns provides partial information about the remaining un-sampled columns. Based upon this observation, we introduce a principled adaptive sampling scheme called Accelerated Sequential Incoherence Selection (oASIS) that predicts the most informative columns to sample from a kernel matrix without forming the entire matrix. oASIS has several advantages over existing adaptive sampling schemes.

- oASIS forms a column of the kernel matrix only when it has been selected to be included in the approximation. For this reason, only the submatrix of sampled columns must be computed/stored.

- oASIS’s total runtime scales linearly with the matrix dimension $n$, making it practical for large datasets. For this reason, oASIS is orders of magnitude faster than other greedy methods that have $O(n^2)$ or higher runtime.

- oASIS accelerates a specific component of Nyström approximation by ensuring that it is fully invertible at every column selection. If no proper selection can be made, oASIS will exit, having exactly recovered the kernel matrix.

- oASIS preserves zero entries in sampled columns of sparsified kernel matrices, enabling greater efficiency in these cases. This is in contrast to conventional greedy methods that require dense $n \times n$ matrix computations that “fill in” the zero entries in a matrix [11].
oASIS provides a tractable solution for approximating extremely large kernel matrices where other adaptive methods simply cannot be applied [11, 13, 14]. In a range of numerical experiments below, we demonstrate that oASIS provides accuracy comparable to the best existing adaptive methods [11, 13, 14, 15] with dramatically reduced complexity and memory requirements.

While oASIS is useful for kernel matrices, its usefulness becomes more pronounced when the data matrix is so large that it can no longer be held entirely in memory. This is because we can parallelize oASIS by splitting up the data matrix and the working matrices among various core processors. We introduce an algorithm called oASIS-P that utilizes OpenMPI [16] in C++ to distribute the data and the submatrices used in approximation, as well as the computation and selection of a new column to add to the approximation. In doing so, we can perform the Nyström approximation in a regime where even the simplest algorithms become difficult to practically implement.

In addition to introducing oASIS, we provide a sufficient condition to guarantee oASIS that will exactly recover a rank-$r$ kernel matrix in $r$ steps. This is due to the fact that oASIS chooses linearly independent columns at each step, which allows for efficient sampling and for efficient computations. This style of column selection is not available with random selection methods, and other greedy methods do not have this guarantee.

This paper is organized as follows. In Section 2, we introduce the Nyström method, survey existing sampling methods, and describe important applications of kernel ma-
trices in classification, clustering, and dimensionality reduction. In Section 3, we describe the motivation behind our column sampling method, called Sequential Incoherence Selection or SIS. We then describe the accelerated version of SIS, or oASIS. We then describe a parallel version of oASIS, which we call oASIS-P. In Section 4, we provide theory determining the conditions under which oASIS will exactly recover the kernel matrix. And in Section 5, we use multiple synthetic and real data sets to demonstrate the efficacy of the proposed method for approximating kernel matrices and diffusion distances for nonlinear dimensionality reduction [2].
Chapter 2

Background

To set the stage, we will quickly describe a few common kernel and distance matrices used in machine learning. Following this, we introduce the Nyström method and describe its variants and applications.

2.1 Notation

We write matrices $G$ and vectors $x$ in upper and lowercase script, respectively. We use $A^\dagger$ to denote the Moore-Penrose pseudo-inverse of $A$. We represent the element-wise product of matrices $A$ and $B$ as $A \circ B$. $\text{colsum}(A)$ denotes a row vector, where the $i^{th}$ entry contains the sum of the $i^{th}$ column of $A$. When describing algorithms, we use “Matlab” style indexing in which $G(i, j)$ denotes the $(i, j)$ entry of the matrix $G$ and $G(:, j)$ denotes its $j^{th}$ column. We use $\Lambda$ to denote a collection of chosen indices of the columns of a matrix $A$, and $\Lambda^c$ is the set of indices not chosen. For example, $A_\Lambda$ are all of the columns of $A$ indexed by the set $\Lambda$.

2.2 Kernel Matrices and their Applications

Kernel methods are widely used in classification and clustering to “lift” data sets into a high-dimensional space where the classes of data are linearly separable. This is done
with the help of a kernel function $k(\cdot, \cdot)$ which measures pairwise similarities between points in the lifted space. An $n \times n$ kernel matrix is then formed from $n$ data points \( \{z_i\}_{i=1}^{n} \) with $G(i, j) = k(z_i, z_j)$, where high magnitude entries of $G$ correspond to pairs of similar data points. The singular vectors of $G$ are then computed and used to map the data back into a low-dimensional space where the data is still linearly separable. The kernel trick is widely used in classification and clustering [1, 4, 7, 17, 18, 19, 20, 21].

Manifold learning methods, including diffusion maps [2] and Laplacian eigenmaps [22], map high-dimensional data that lie on nonlinear but low-dimensional manifolds onto linear low-dimensional spaces. These methods use a kernel matrix that encodes the geodesic distance between pairs of points — the shortest path between two points along the surface of the underlying manifold. High-dimensional data are mapped into a low-dimensional space using the left singular vectors of $G$ and thus a singular value decomposition (SVD) of $G$ is required. For a review of dimensionality reduction using geodesic and diffusion distance kernel matrices, see [12, 23].

### 2.3 The Nyström Method

Williams and Seeger [1] first presented the Nyström method to improve the speed of kernel-based methods for classification. The method approximates a low-rank symmetric positive semidefinite (PSD) matrix using a subset of its columns.

Consider an $n \times n$ PSD matrix $G$ of rank $r$. For all PSD matrices $G$, there exists a matrix $X \in \mathbb{R}^{r \times n}$ such that $G = X^T X$. Suppose we choose a set $\Lambda$ of $k \leq r$ indices,
then sample those Λ columns from $G$ as $C_k, C_k \in \mathbb{R}^{n \times k}$. We collect the $k$ indices into a set Λ. The sampling forms a partition of $X = \begin{bmatrix} X_\Lambda & X_{\Lambda^c} \end{bmatrix}$. We can express $G$ as

$$G = \begin{bmatrix} X_\Lambda^T & X_{\Lambda^c}^T \\ X_{\Lambda^c}^T & X_\Lambda^T \end{bmatrix} = \begin{bmatrix} W_k & X_\Lambda^T X_{\Lambda^c} \\ X_{\Lambda^c}^T X_\Lambda & X_{\Lambda^c}^T X_{\Lambda^c} \end{bmatrix},$$

(2.1)

where $C_k = \begin{bmatrix} W_k & X_{\Lambda^c}^T X_\Lambda \end{bmatrix}^T$ consists of the $n \times k$ sampled columns of $G$, and $W_k = X_{\Lambda^c}^T X_\Lambda$ is the $k \times k$ symmetric matrix containing the row entries of the sampled columns at the indices of the sampled columns. Note that without loss of generality we can permute the rows and columns of $G$ so that the columns in $C_k$ are the first $k$ columns of $G$. The Nyström approximation of $G$ is defined as

$$G \approx \tilde{G}_k = C_k W_k^T C_k^T.$$  

(2.2)

Note that neither $X$ nor any partition of $X$ is found explicitly, but that the $\tilde{G}_k$ is found through the set of sampled columns $C_k$ and the respective rows $W_k$.

An approximate SVD of $G$ can then be obtained from the SVD of $W_k$, which is written as $W_k = U_W \Sigma_W U_W^T$. The singular values $\Sigma$ of the approximation $\tilde{G}_k = \tilde{U} \tilde{\Sigma} \tilde{U}^T$ are given by $(n/k) \Sigma_W$ [24], and the singular vectors are given by

$$\tilde{U} = \sqrt{\frac{k}{n}} C_k U_W \Sigma_W^{-1}.$$

Since $W_k$ is $k \times k$, this computation is much faster than computing the full $n \times n$ SVD of $G$. The complexity of the SVD step reduces from $O(n^3)$ to $O(k^3)$ with $k \leq r \ll n$.

Note that the Nyström method enables the singular vectors of $G$, and thus a low dimensional embedding of the data, to be computed from only a subset of its columns.
When $n$ is large, it is desirable to form only a subset of the columns of $G$ rather than calculate and store all $n(n-1)/2$ pairwise kernel distances.

### 2.4 Column Sampling Methods

We now describe the four main categories of column selection methods. We compare oASIS with the methods listed below, as together they cover all of the types of sampling used currently in Nyström approximation.

#### 2.4.1 Uniform Random Sampling

Early work on the Nyström method focused on random column sampling methods [1]. Theoretical bounds on the accuracy of a rank-$k$ approximation to $G$ after sampling a sufficient number of columns have been developed for various norms [14].

Uniform random sampling methods can be improved using weighted probability distributions that consider the norms of the columns of $G$ [5]. Determining the distributions can be done in $O(n^2)$ time, assuming the matrix is precomputed.

#### 2.4.2 Non-deterministic Adaptive Sampling

Leverage scores are a recent method for computing the distribution over the column draw [14]. Given the rank-$k$ SVD of $G_k = U_k \Sigma_k U_k^T$, the scores are computed as $s_j = \|U_k(j,:)\|^2$, and each column is selected with probability proportional to its score. Leverage scores require the low-rank approximate SVD of $G$ to be precomputed at
\(O(n^3)\) cost. There are faster approximations to finding the first few singular vectors and values of \(G\) [25]. Regardless, all methods require \(G\) to be precomputed.

### 2.4.3 Deterministic Adaptive Sampling

Early adaptive methods [13] use an exhaustive search to find columns that minimize the Frobenius norm error \(\|G - \tilde{G}_k\|_F\). A more efficient adaptive scheme from Farahat [11] builds a Nyström approximation by selecting columns sequentially using a matrix of “residuals.” At each stage of the method, the column with the largest residual is selected, and the residual matrix is updated to reflect its contribution. The cost of updating the residual matrix is \(O(n^2)\) per iteration.

This criterion is related to the adaptive probability distribution calculated by Deshpande [10]. After a sufficient number of columns are chosen, an orthogonalization step ensures a best rank-\(k\) approximation of \(\tilde{G}_k\).

### 2.4.4 Representative Point Sampling with \(K\)-means Clustering

A representative sampling from a set of data points can be found with a \(K\)-means algorithm. A dataset consisting of clouds of points clustering around \(k\) centroids can be described by finding the locations of the centroids. Each datapoint is then remapped into the eigenspace of its cluster. This method was first described by Zhang [15]. Since the computed centroids do not exist in the dataset, the method does not directly sample columns, but remaps them onto a rank-\(k\) space. Once the solution to the \(K\)-means is found, the remapping is \(O(\ell n)\). While finding an exact solution to
$K$-means is NP-hard, generally $K$-means will converge fairly quickly.

In a survey of methods by Kumar [24], $K$-means was found to be the state-of-the-art approximation method compared to previous sampling methods such as Incomplete Cholesky Decomposition [26], Sparse Matrix Greedy Approximation [13], and Kumar’s Adaptive Partial method derived from Deshpande’s Adaptive Sampling method [10]. For this reason, in lieu of comparisons with many different adaptive sampling techniques we can compare our results directly with $K$-means. For our experiments, we used the same code as provided in [15], and parameters used in both [15] and [24].

2.5 Finding Low-Dimensional Structure

In addition to using CSS for low-rank kernel approximation, column selection approaches have also been used to find important columns of $X$ and in turn, reveal low-dimensional structure in data [27, 28]. Recently, we have shown that oASIS can be used to select representative examples from large data sets in order to enable clustering and dimensionality reduction [29]. This method, which we dub Sparse Self-Expressive Decomposition (SEED), consists of two main steps. First, we employ oASIS for column selection on the Gram matrix of the data, where $G = Z^T Z$. Following column selection, we form a factorization of $Z = DV$, where $D$ is a dictionary formed by the normalized $Z_A$ and the other points of $Z$ are then represented by a sparse combination of the points in $D$ using Orthogonal Matching Pursuit [30, 31].
Once we have computed the final representation $Z \approx \hat{Z} = DV$, we can use the sparsity patterns in $V$ for clustering, denoising, or classification. A full treatment of SEED is beyond the scope of this paper, but can be found in [29].

One of the most important aspects of SEED is its ability to analyze a data set of points that lie on different low-dimensional subspaces, and recover those subspaces from the data. For this purpose, the appropriate selection $Z_\Lambda$ is paramount. The selected data points need to represent all of the subspaces, and the representations can only use selected data points that lie in the same subspace. This condition is called exact feature selection. Provable guarantees on recovering the correct subspaces in which the data lie require at least $k$ linearly independent columns in $Z$ that span each $k$-dimensional subspace in the dataset [32, 33, 34]. In [29], we show that when $Z$ is drawn from a union of independent subspaces, SEED, through oASIS, returns an $Z_\Lambda$ that provides such a set. We state the guarantee here, although the proofs can be found in [29].
Chapter 3

Accelerated Sequential Incoherence Selection (oASIS)

In this section, we introduce a new adaptive sampling method for column subset selection called oASIS, a parallel version called oASIS-P, and analyze their complexity.

3.1 Sequential Incoherence Selection (SIS)

We now address the question of how to build a Nyström approximation by sequentially choosing columns from $G$. Suppose we have already selected $k$ columns from $G$ and computed a Nyström approximation $\tilde{G}_k$. Our goal is to select a new column that improves the approximation. If a candidate column lies in the span of the columns that we have already sampled, then adding this column has no effect on $\|G - \tilde{G}_k\|$. Ideally, we would like to quantify how much each candidate column will change the existing approximation and select the column that will produce the most significant impact. Since an ideal column does not lie in the span of the columns that have been selected, we say that this column is incoherent with those already selected.

Consider a PSD matrix $G$. Recall that any such $G$ can be written as $X^TX$, where $X$ contains $n$ points in $r$ dimensions. Given an index set $\Lambda$ of $k$ columns we can form a partition $X = \begin{bmatrix} X_\Lambda & X_{\Lambda^c} \end{bmatrix}$, and can collect the $k$ selected columns from $G$ into a
matrix \( C_k = G_\Lambda = \begin{bmatrix} W_k & X_\Lambda^T X_\Lambda \end{bmatrix}^T \). To improve the approximation \( \tilde{G}_k \), we select the best new column from \( G \), append it to \( C_k \), and compute a new \( \tilde{G}_{k+1} \). The best new column index \( i \in \Lambda^c \) in \( G \) directly corresponds to the index of the column \( x_i \) that lies farthest from the span of \( X_\Lambda \). This column \( x_i \) satisfies

\[
\arg \max_{i \in \Lambda^c} \| (I - P_\Lambda) x_i \|_2^2,
\]

where \( I \) is the identity matrix and \( P_\Lambda = X_\Lambda X_\Lambda^T \) is an orthogonal projection onto the span of the \( k \) columns in \( X \) that have already been selected. Provided that the columns in \( X_\Lambda \) are linearly independent, we can expand (3.1) as

\[
\arg \max_{i \in \Lambda^c} x_i^T x_i - x_i^T X_\Lambda (X_\Lambda^T X_\Lambda)^{-1} X_\Lambda^T x_i,
\]

(3.2)

Even though \( X \) is not known explicitly, Eq. (3.2) can be evaluated based upon knowledge of \( C_k \) and \( \text{diag}(G) \). The first term of the expression in (3.2) is the diagonal entry \( i \) of \( G \), which we denote as \( d_i \). The second term can be written as \( b_i^T W_k^{-1} b_i \), where \( b_i^T = x_i^T X_\Lambda \) is one of the \( n - k \) rows of \( C_k \) indexed by \( \Lambda^c \), and \( W_k \) is comprised of the \( k \) rows of \( C_k \) indexed by \( \Lambda \). Note that when \( X_\Lambda \) contains linearly dependent columns, we replace \( W_k^{-1} \) with \( W_k^\dagger \).

Based upon this idea, we develop the following sampling method for sequential incoherent selection (SIS). We assume the process has been seeded with a small set \( \Lambda \) of \( k_0 \) column indices chosen at random. Columns are then selected to be included in the approximation as follows:

1. Let \( k = |\Lambda| \). Collect the columns of \( G \) indexed by \( \Lambda \) as \( C_k \). Form \( W_k^{-1} \) from the
rows of $C_k$ indexed by $\Lambda$.

2. Let $b_i^T$ denote row $i$ of $C_k$, and let $d_i$ denote element $i$ of $\text{diag}(G)$. For each unselected column $i \in \Lambda^c$, calculate:

$$\Delta_i = d_i - b_i^T W_k^{-1} b_i.$$  

3. Select the column that maximizes $|\Delta_i|$ and set $\Lambda \leftarrow \Lambda \cup \{i\}$.

4. If the selected value of $|\Delta_i|$ is smaller than a user set threshold, then terminate. Otherwise, return to Step 1.

### 3.2 oASIS

A naive implementation of SIS in Section 3.1 is inefficient because each step requires a matrix inversion to form $W_k^{-1}$ in addition to calculating the errors $\Delta_i$. Fortunately, both of these calculations can be performed efficiently by updating the results from the previous step using block matrix inversion formulas. We dub this new method oASIS.

We first consider the calculation of $W_{k+1}^{-1}$ after a new column is added to the approximation made from $k$ columns. We assume throughout the rest of this section that $W_{k+1}$ is invertible and thus $W_{k+1}^T = W_{k+1}^{-1}$. We show that our column selection rule guarantees the invertibility of $W_k$ in Section 4.1. Let $b$ denote the first $k$ entries of the new column, $d$ denote the relevant element of $\text{diag}(G)$, and $\Delta_{k+1} = d - b^T W_k^{-1} b$. 
Using a block inversion formula, we obtain

\[
W_{k+1}^{-1} = \begin{bmatrix} W_k & b \\ b^T & d \end{bmatrix}^{-1} = \begin{bmatrix} W_k^{-1} + sqq^T & -sq \\ -sq^T & s \end{bmatrix},
\]

(3.3)

where \( s = (d - b^TW_k^{-1}b)^{-1} = \Delta_{k+1}^{-1} \) is the (scalar valued) Schur complement and \( q = W_k^{-1}b \) is a column vector. This update formula enables \( W_{k+1}^{-1} \) to be formed by updating \( W_k^{-1} \) and only requires inexpensive vector-vector multiplication. Note that \( W_{k+1} \) is invertible as long as \( \Delta_{k+1} \) is non-zero, which is guaranteed since the algorithm terminates if \( \Delta_{k+1} = 0 \), in which case our approximation is exact.

Now consider the calculation of \( \Delta_i = d_i - b_i^TW_kb_i \) for all candidate columns \( i \). Note that \( C_k^T = [b_1, b_2, \ldots, b_n] \). We can evaluate all \( b_i^TW_kb_i \) simultaneously by computing the entry-wise product of \( C_k \) with the matrix \( R_k := W_k^{-1}C_k^T \) and then summing the resulting columns. If we have already formed \( C_k \) and \( R_k \), the matrix \( R_{k+1} = W_{k+1}^{-1}C_{k+1}^T \) needed on the next iteration is obtained by applying (3.3) to \( C_k^T \) to obtain

\[
R_{k+1} = W_{k+1}^{-1}C_{k+1}^T = W_{k+1}^{-1} \begin{bmatrix} C_k^T \\ c_{k+1}^T \end{bmatrix} = \begin{bmatrix} R_k + sq(q^Tc_k - c_{k+1}^T) \\ s(-q^Tc_k + c_{k+1}^T) \end{bmatrix}.
\]

(3.4)

Equation (3.4) forms \( R_{k+1} \) by updating the matrix \( R_k \) from the previous iteration. The update requires only matrix-vector and vector-vector products. The application of this fast update rule to the method described in Section 3.1 yields oASIS, detailed in Figure 3.1.
oASIS can be initialized with a small random subset of $k_0$ starting columns from $G$. Next, the starting matrices $C_k$, $W_k^{-1}$ and $R_k = W_k^{-1}C_k^T$ are formed. On each iteration of the algorithm, the vector of Schur complements $\Delta$ is formed by computing

$$\Delta = d - \text{colsum}(C_k \circ R_k)$$

Next, the largest entry in $\Delta$ is found, and its index is used to select the next column from $G$. The update formulas (3.3) and (3.4) are then used to form the matrices $W_{k+1}^{-1}$ and $R_{k+1}$ required for the next iteration.

### 3.3 Parallelized oASIS

For the case of kernel matrices $G$ made from a data matrix $Z$, oASIS does not need to explicitly store $G$. However, $Z$ may be itself very difficult to store in memory due to size. In addition, oASIS requires space to build $C$, $W^{-1}$, and $R$. In total, oASIS requires $O(mn + \ell^2 + 2\ell n)$ of memory. In cases where $Z$ is too large to fit, the matrix operations for oASIS can be distributed among $p$ separate processor cores. Each core stores a submatrix $Z_{(\text{core})}$ consisting of $n/p$ points of $Z$, a copy of $W^{-1}$, and the column entries of $C_k$ and $R_k$ corresponding to the results of the kernel function over the entries in $Z_{(\text{core})}$ with the entries in $Z_\Lambda$. When a new column index $i$ is selected, the core storing column vector $z_i$ is found and the column is broadcast to all of the cores. Each core can then calculate the appropriate new entries as needed. The memory requirements for each core then become $O(mn/p + \ell^2 + 2\ell n/p + \ell m)$, which makes performing oASIS over datasets with millions of points tractable.
Algorithm 1: oASIS

**Inputs:** Symmetric matrix $G \in \mathbb{R}^{n \times n}$,
- Diagonal elements of $G$, stored in $d$,
- Maximum number of sampled columns, $\ell$,
- Initial number of sampled columns, $k < \ell$,
- Non-negative stopping tolerance, $\epsilon$.

**Outputs:** The sampled columns $C$,
- The inverse of the sampled rows $W^{-1}$.

**Initialize:** Choose a vector $\Lambda \in [1,n]^\ell$ of $k$ random starting indices.

- $C_k = G(:, \Lambda)$
- $W_k^{-1} = G(\Lambda, \Lambda)^{-1}$
- $R_k = W_k^{-1}C_k^T$

while $k < K$ do

- $\Delta = d - \text{colsum}(C_k \odot R_k)$
- $i = \text{arg max}_{j \in \Lambda} |\Delta(j)|$
- if $|\Delta(i)| < \epsilon$ then

- return

- $b = G(\Lambda, i)$
- $d = d(i)$
- $s = 1/\Delta(i)$
- $q = R(:, i)$
- $C_{k+1} = [C_k, G(:, i)]$

- Form $W_{k+1}^{-1}$ using (3.3)

- Update $R_{k+1}$ using (3.4)

- $k \leftarrow k + 1$

- $\Lambda \leftarrow \Lambda \cup \{i\}$

end while

Figure 3.1: The oASIS algorithm. Note that $G$ need not be precomputed.

We call this method Parallelized oASIS or oASIS-P, and it is detailed in Figure 3.2.

We will write the command $\text{Broadcast}(data)$ to send data from one core to every core, and the command $\text{Gather}(variable)$ to concatenate variable in each core into a single variable.
Algorithm 2: oASIS-P

**Inputs:** Data matrix $Z \in \mathbb{R}^{m \times n}$
- Kernel function $g(z_i, z_j)$,
- Number of cores $p$,
- Maximum number of sampled columns, $\ell$,
- Initial number of sampled columns, $k < \ell$,
- Number of columns in $Z$, $n$,
- Non-negative stopping tolerance, $\epsilon$.

**Outputs:** The sampled columns $C$,
- The inverse of the sampled rows $W^{-1}$.

**Initialize:** Choose a vector $\Lambda \in [1, n]^\ell$ of $k$ random starting indices.
- Load separate $n/p$ column blocks of $Z$ into each core as $Z_{(core)}$.
- $Broadcast(Z(:, \Lambda))$ as $Z_{\Lambda}$.

Form $d_{(core)}$ from $Z_{(core)}$ and $g$
Form $C_{(core)}$ from $Z_{(core)}$ and $g$
Form $W^{-1}_{(core)}$ from $Z_{\Lambda}$ and $g$

Form $R_{(core)} = W^{-1}_{(core)} C_{(core)}^T$

while $k < K$ do
  $\Delta_{(core)} = d_{(core)} - \text{colsum}(C_{(core)} \circ R_{(core)})$
  $\Delta = Gather(\Delta_{(core)})$
  $i = \arg \max_{j \in \Lambda} |\Delta(j)|$
  if $|\Delta| < \epsilon$ then
    return
  end if
  $z_{k+1} = Broadcast(Z(:, i))$
  Form $c_{k+1}$ from $Z_{(core)}$, $z_{k+1}$, and $g$
  $C_{(core)} = [C_{(core)}, c_{k+1}]$
  Form $q$ from $Z_{\Lambda}$, $z_{k+1}$, and $g$
  $s = 1/\Delta$
  Update $W^{-1}_{(core)}$ using (3.3)
  Update $R_{(core)}$ using (3.4)
  $Z_{\Lambda} = [Z_{\Lambda}, z_{k+1}]$
  $k \leftarrow k + 1$
  $\Lambda \leftarrow \Lambda \cup \{i\}$
end while

$W^{-1} = W^{-1}_{(core)}$
$C = Gather(C_{(core)})$

Figure 3.2: oASIS-P, for data matrices too large for a single core.
Chapter 4

Properties and Applications of oASIS

4.1 Theoretical Guarantees of oASIS

For general PSD matrices $G$ of rank $r$, we can guarantee that oASIS will finish in $r$ steps. When $G$ is a Gram matrix with $X$ known, then we can use the sample index set $\Lambda$ found with oASIS to make additional guarantees on approximating $X$. We mention the guarantees briefly in Section 4.1.3, and they are described fully in [29].

We can also guarantee that oASIS will select linearly independent columns of $G$ at each step. This becomes very useful in practice, as $\tilde{G}$ is computed from the $W^\dagger$, where $W = X_A^T X_A$. If the selected columns of $G$ are not independent, then $W$ is a singular matrix. By selecting linearly independent columns of $G$, oASIS can guarantee that $W^\dagger = W^{-1}$, allowing for time and space saving calculation of this element when computing $\tilde{G}$. We discuss this further in Section 4.1.4.

4.1.1 Independent Selection Property of oASIS

Given a PSD matrix $G = X^T X$, rank($G$) = rank($X$) = $r$. In Lem. 1 below, we provide a sufficient condition that describes when oASIS will return a set of $r$ linearly independent columns. This is a similar condition as that provided in [29], although
we provide an alternate proof.

**Lemma 1.** At each step of Alg. 3.1, the $i^{th}$ column of the matrix $G$ is linearly independent from the previously selected columns provided that $\Delta(i) > 0$.

**Proof.** We prove this by construction of $X_\Lambda$. Consider adding a new column $x_i$ to $X_\Lambda$ with nonzero $\Delta(i) = \|(I - P_\Lambda)x_i\|_2^2$ and $i \in \Lambda^c$. Then $x_i$ is linearly independent of each column in $X_\Lambda$, and so $G_{k+1} = X^T x_i$ is linearly independent from any other $G_j = X^T x_j$. Therefore as long as $\Delta(i) > 0$ at each step, the column selected will be linearly independent from the previous columns selected.  

**Remark.** This result guarantees that oASIS will return a set of $r$ linearly independent columns in $r$ steps as long as the selection criterion $\Delta(i) \neq 0$ holds before exact reconstruction occurs. Unfortunately, in the pathological case in which the algorithm fails with $\Delta(i) = 0$ before $r$ columns have been selected, the algorithm may terminate early. While it is possible to construct pathological matrices where this occurs, we have not observed this early termination in practice.

### 4.1.2 Exact Matrix Recovery

We now will prove that when $r$ columns are selected from $G$ with oASIS, then $\tilde{G} = G$.

**Theorem 1.** If oASIS chooses $r$ columns from a PSD matrix $G$ with $\text{rank}(G) = r$, the Nyström approximation $\tilde{G} = G$.

**Proof.** As $X$ is $\text{rank}(r)$, and oASIS has chosen $r$ linearly independent columns, then at the next step all $\Delta(i) = 0$ as $\|(I - P_\Lambda)x_i\|_2^2 = 0 \forall i$. Therefore $\|(I - X_\Lambda X_\Lambda^\dagger)x_i\|_2^2 = 0$,
or $\|X - X_{\Lambda}X_{\Lambda}^\dagger X\|_F = 0$, $X = \begin{bmatrix} X_{\Lambda} & X_{\Lambda^c} \end{bmatrix}$, and therefore $\|X_{\Lambda^c} - X_{\Lambda}X_{\Lambda}^\dagger X_{\Lambda^c}\|_F = 0$.

or $X_{\Lambda^c} = X_{\Lambda}X_{\Lambda}^\dagger X_{\Lambda^c}$. Expanding $\tilde{G}$ in terms of $X_{\Lambda}$ and $X_{\Lambda^c}$,

$$
\tilde{G} = CW^\dagger C^T = \begin{bmatrix} X_{\Lambda}^TX_{\Lambda} & X_{\Lambda}^TX_{\Lambda^c} \\ X_{\Lambda^c}^TX_{\Lambda} & X_{\Lambda^c}^TX_{\Lambda^c} \end{bmatrix} = \begin{bmatrix} X_{\Lambda}^TX_{\Lambda} & X_{\Lambda}^TX_{\Lambda^c} \\ X_{\Lambda^c}^TX_{\Lambda} & X_{\Lambda^c}^TX_{\Lambda}X_{\Lambda^c}^\dagger X_{\Lambda^c}^TX_{\Lambda^c} \end{bmatrix}.
$$

We examine the lower right block of this expansion as the others exactly match that of $G$ in (2.1).

By Lemma 1 $X_{\Lambda}$ is full rank, and so

$$
X_{\Lambda^c}^TX_{\Lambda}X_{\Lambda}(X_{\Lambda}^TX_{\Lambda})^\dagger X_{\Lambda}^TX_{\Lambda^c} = X_{\Lambda^c}^TX_{\Lambda}X_{\Lambda}^{-1}X_{\Lambda}^TX_{\Lambda^c} = X_{\Lambda^c}^T(X_{\Lambda}X_{\Lambda}^\dagger)X_{\Lambda^c} = X_{\Lambda^c}^TX_{\Lambda^c}
$$

and so then the expansion of $\tilde{G}$ is equal to the expansion of $G$ in (2.1).

4.1.3 Guarantees of oASIS when $G$ is a Gram Matrix

In the case where $G = Z^T Z$, with $Z$ a data matrix of rank $r$, oASIS selects a set $|\Lambda| = r$ such that $Z = P_{\Lambda}(Z)$ exactly. This property is useful in solving a more general CSS problem than Nyström, precisely formulated as

$$
\min_{|\Lambda| = L} \|Z - P_{\Lambda}Z\|_F, \tag{4.1}
$$
Figure 4.1: For a PSD matrix $G = Z^T Z$, with the dataset $Z$ as in (a), we compare approximation errors for $\tilde{G}$ formed using oASIS or uniform random sampling in (b), and the rank of $\tilde{G}$ with either method in (c). We terminate trials at exact recovery. See Section 4.1.4 for details.

where $Z$ is an $m \times n$ matrix. This problem has combinatorial complexity, and many of the selection schemes for Nyström arise from attempting to solve this more general problem. Indeed, the adaptive selection methods in [5, 10], and others, can also apply to this problem. We have developed guarantees on oASIS’s ability to exactly recover $X$ so that we can use the columns in $X_\Lambda$ in developing a self-expressive decomposition of $X$. Although a full treatment of SEED is described in [29], we briefly describe this
extension in Section 2.5.

4.1.4 Comparison with other theory

oASIS can guarantee exact matrix recovery in an information theoretically efficient number of steps. We show a synthetic example in Figure 4.1. Using a dataset $Z$ consisting of points drawn from a 2D Gaussian distribution centered on $(0, 0)$ and points drawn from a 3D Gaussian distribution centered on $(0, 0, 1)$, we compute a PSD matrix $G = Z^T Z$ with $\text{rank}(G) = 3$. oASIS selects columns linearly independent from previously selected columns at each step, increasing the rank of the approximation each time. This allows oASIS to use an iterative update of $W^{-1}$ instead of computing $W^\dagger$ after all columns have been selected. At 3 steps, oASIS terminates, with $\tilde{G} = G$ within machine tolerance.

Random or adaptive random sampling techniques have theoretical guarantees that $\tilde{G}$ will be close to a rank-$k$ approximation of $G$ after a number of iterations [5, 14]. However, the lack of guarantees on column selection make for redundant sampling. As an illustration, we include separate trials of uniform random sampling in Figure 4.1. Uniform random sampling does not necessarily select columns linearly independent from previously selected columns at each step, and as a result the approximation is not particularly good until enough columns have been selected to form a rank 3 $\tilde{G}$. In cases of very large data, this becomes a practical concern in computing both $C$ and $W^\dagger$. 
Other greedy algorithms tend not to have these guarantees [11], and rely on experimental verification of accuracy.

4.2 Complexity of oASIS

The rate-limiting step of oASIS in Figure 3.1 is the computation of $R_{k+1}$ by updating $R_k$. Equation (3.4) enables this to be performed by sweeping over the entries of $R_k$, which has dimension $k \times n$. The complexity of a single iteration is thus $O(kn)$. If $\ell$ columns are sampled in total, then $\sum_{k=1}^{\ell} kn = \frac{1}{2} \ell(\ell + 1)n$ entries must be updated. The resulting complexity of the entire algorithm is $O(\ell^2 n)$. In practice, the number of sampled columns $\ell \ll n$. This makes oASIS considerably more efficient than adaptive methods such as Farahat’s [11], which requires the computation of $n \times n$ residual matrices at each stage resulting in $O(\ell n^2)$ complexity. oASIS is also more efficient than Leverage scores [14], as the scores use the SVD of $G$ at worst cost $O(n^2)$ over dense matrices. oASIS is about as efficient as $K$-means Nyström, with complexity $O(\ell n)$. However, $K$-means does not provide a column selection method, and instead forms the full $\tilde{G}$ from the low-dimensional remapping. As a result, while $K$-means is useful in Nyström, it may not be as useful for more general CSS methods. oASIS, in contrast, can be used in more general CSS problems via the Gram matrix.

If we only compare the speed in finding $\Lambda$, oASIS is much slower than uniform random sampling and its $O(1)$ sampling speed. But oASIS also computes $C$ and $W^{-1}$ along the way, while these still need to be computed after selecting the columns to be
used under uniform random sampling. In large data regimes, these become practical considerations that make implementation of uniform random sampling less effective, as we discuss in Section 4.3.

4.3 Complexity of oASIS-P

The low complexity of oASIS makes it practical for extremely large matrices where other adaptive sampling schemes are intractable. For oASIS-P, the computational complexity of oASIS is divided by the $p$ cores, such that each core has $O(\ell^2 n/p)$ complexity. At first blush, oASIS-P is still slower than uniform random sampling and its $O(1)$ sampling speed. However, in regimes where the data matrix can’t be loaded entirely in memory, two practical considerations make uniform random sampling less competitive. First, forming $\ell$ columns from a data matrix $Z$ with $m$-dimensional entries takes $O(\ell n)$ time. Depending on the dataset, $\ell_{random}$ may be much larger than $\ell_{oASIS}$. Second, uniform random sampling does not guarantee an invertible $W$. Computing $W^\dagger$ is an additional $O(\ell^3)$ in time. While uniform sampling is still more efficient, it is not as showstopping as it first appears. In fact, communication of data vectors among cores becomes the bottleneck, and oASIS-P and random uniform sampling appear competitive in time, as shown in Table 5.3.
Chapter 5

Numerical Experiments

To evaluate the performance of oASIS, we measure the accuracy of Nyström approximations for three size classes of kernel matrices. We first consider matrices where we can directly measure the approximation error of the Nyström method. Second, we consider larger problems for which forming the entire kernel matrix is impractical.

Figure 5.1: Nyström Approximation Error Curves and Run Time Comparison. See Section 5 for details. Note that oASIS is accurate and scales well to large problem sizes due to its low runtime complexity (see Section 4.2).
Figure 5.2: Nyström Error curves and number of samples based on time. See Section 5 for details. Note that oASIS is accurate and scales well to large problem sizes due to its low runtime complexity (see Section 4.2).

Third, we consider problems so large that the data matrix will not fit in memory. For each dataset \( \{z_i\}_{i=1}^n \) in all classes, we consider Gaussian kernel matrices where 

\[
G(i,j) = \exp\left(\frac{\|z_i - z_j\|^2}{\sigma^2}\right).
\]

For datasets in the first class we also consider diffusion
distance matrices $M = D^{-1/2}GD^{-1/2}$ where $D$ is a diagonal matrix containing the row sums of $M$, and $G$ is a Gaussian kernel matrix [2]. For each dataset, we tune $\sigma$ to provide good matrix approximation for any sampling method.

We compare oASIS to the following state-of-the-art Column Subset Selection (CSS) methods: (i) uniform random sampling, (ii) Leverage scores [14] (Section 2.4.2), (iii) Farahat’s greedy update method [11] (Section 2.4.3). We also compare with the $K$-means Nyström approximation [15] (Section 2.4.4). For methods (i), (ii), and $K$-means, we repeat experiments 10 times and average the results. For the second class of matrices we consider oASIS, uniform random sampling, and $K$-means since the other methods become intractable when the matrix becomes too large to explicitly store. For the largest class of matrices we only consider oASIS and uniform random sampling. Specific datasets and experiments are described below.

5.1 Full Kernel Matrices

Here, we consider datasets for which the kernel matrices can entirely fit in memory, making all the sampling methods tractable. Convergence curves are generated by forming $\tilde{G}_k$ for increasing $k$ and then calculating the approximation error defined by $\|\tilde{G}_k - G\|_F/\|G\|_F$. We consider the following datasets, run using MATLAB on an iMac with a 2.7 GHz processor and 16GB of memory. Results at the largest sample sizes are shown for full matrices in Table 5.1. Figure 5.1 shows selected convergence curves (normalized error vs. number of columns sampled). Figure 5.1 also presents a plot of
column selection runtime vs. matrix size for a variety of methods. Figure 5.2 shows convergence curves (normalized error vs. number of seconds runtime) and column sampling rates (number of columns sampled vs. number of seconds runtime) for all adaptive methods using the Gaussian kernel. These curves allow for a fair assessment of approximation error achieved after a set run time for various adaptive methods, as methods will select columns at different rates.

**Two Moons** We consider a common synthetic dataset for clustering algorithms that consists of 2-dimensional points arranged in two interlocking moons. The data matrix is $2000 \times 2$. We set the kernel $\sigma$ equal to 5% of the maximum Euclidean distance between points.

**Abalone** The Abalone dataset is a collection of physical characteristics of abalone [35], which are analyzed to find the age of the abalone without direct measurement. The data matrix is $4177 \times 8$. We set the kernel $\sigma$ equal to 5% of the maximum Euclidean distance between points.

**Binary Organization of Random Gaussians (BORG)** The BORG assimilates sets of points clustered tightly around each vertex of an $n$-dimensional cube. The points around each vertex $v$ are distributed as $\mathcal{N}(v, \sigma^2 I)$ with $\sigma^2 = 0.1$. This data set is constructed to be pathologically difficult, with many clusters and many points per cluster. Many columns are needed from $G$ to ensure sampling from each cluster. With 256 clusters of 30 points each, the data matrix is $7680 \times 8$. We set the kernel
\( \sigma \) equal to 12.5\% of the maximum Euclidean distance between points.

### 5.2 Implicit Kernel Matrices

Here, we consider datasets for which the resulting kernel matrix would become impractical to explicitly calculate and store. Rather, columns from \( G \) are generated “on the fly” at the time they are sampled. Since a full representation of \( G \) is no longer available, we estimate the approximation error as the Frobenius-norm discrepancy between 100,000 randomly sampled entries in \( G \) and the corresponding entries in \( \widetilde{G}_k \). Because the Leverage scores [14] and Farahat [11] schemes require a full representation of \( G \) (which is intractable for these problem instances), we compare with uniform random sampling and \( K \)-means. We consider the following datasets, run using MATLAB on an iMac with a 2.7 GHz processor and 16GB of memory. Results at the largest sample sizes are shown for implicit matrices in Table 5.2.
Table 5.1: Error and Runtime Results for Gaussian (top) and Diffusion (bottom) Kernel Matrices.

<table>
<thead>
<tr>
<th>Problem</th>
<th>$n$</th>
<th>$\ell$</th>
<th>oASIS</th>
<th>Random</th>
<th>Leverage scores</th>
<th>$K$-means</th>
<th>Farahat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two Moons</td>
<td>2,000</td>
<td>450</td>
<td>$1.00e-6$ (1.20)</td>
<td>$2.14e-3$ (0.01)</td>
<td>$9.46e-4$ (3.96)</td>
<td>$1.05e-3$ (0.38)</td>
<td>$8.31e-7$ (19.7)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$1.10e-6$ (1.16)</td>
<td>$1.22e-2$ (0.01)</td>
<td>$7.45e-3$ (4.00)</td>
<td>$5.49e-3$ (1.21)</td>
<td>$1.11e-6$ (19.6)</td>
</tr>
<tr>
<td>Abalone</td>
<td>4,177</td>
<td>450</td>
<td>$1.23e-6$ (2.60)</td>
<td>$2.65e-3$ (0.01)</td>
<td>$5.23e-4$ (35.8)</td>
<td>$1.73e-3$ (0.84)</td>
<td>$2.85e-7$ (64.8)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$1.62e-6$ (2.51)</td>
<td>$3.76e-1$ (0.01)</td>
<td>$1.47e-1$ (35.9)</td>
<td>$3.24e-1$ (8.26)</td>
<td>$5.61e-7$ (64.7)</td>
</tr>
<tr>
<td>BORG</td>
<td>7,680</td>
<td>450</td>
<td>$5.30e-2$ (4.71)</td>
<td>$3.90e-1$ (0.01)</td>
<td>$4.31e-1$ (252)</td>
<td>$8.89e-2$ (2.53)</td>
<td>$2.75e-2$ (176)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$6.29e-2$ (4.73)</td>
<td>$3.90e-1$ (0.01)</td>
<td>$4.23e-1$ (244)</td>
<td>$7.70e-2$ (48.3)</td>
<td>$2.78e-2$ (174)</td>
</tr>
</tbody>
</table>
Table 5.2: Error and Runtime Results for Implicit Kernel Matrices.

<table>
<thead>
<tr>
<th>Problem</th>
<th>n</th>
<th>$\ell$</th>
<th>oASIS</th>
<th>Random</th>
<th>$K$-means</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>50,000</td>
<td>4,000</td>
<td>$1.53e-6$ (8260)</td>
<td>$7.48e-5$ (946)</td>
<td>$7.30e-7$ (188)</td>
</tr>
<tr>
<td>Salinas</td>
<td>54,129</td>
<td>5,000</td>
<td>$2.22e-5$ (13372)</td>
<td>$1.61e-4$ (819)</td>
<td>$1.33e-5$ (1120)</td>
</tr>
<tr>
<td>Light Field</td>
<td>82,265</td>
<td>4,000</td>
<td>$7.10e-6$ (13600)</td>
<td>$2.54e-5$ (989)</td>
<td>$9.44e-6$ (1890)</td>
</tr>
</tbody>
</table>

Table 5.3: Error and Runtime Results for Parallelized Implicit Kernel Matrices.

<table>
<thead>
<tr>
<th>Problem</th>
<th>n</th>
<th>$\ell$</th>
<th>oASIS-P</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two Moons</td>
<td>1,000,000</td>
<td>1,000</td>
<td>$5.10e-6$ (108)</td>
<td>$5.90e-4$ (279)</td>
</tr>
<tr>
<td>Tiny Images</td>
<td>1,000,000</td>
<td>4,500</td>
<td>$2.76e-4$ (10672)</td>
<td>$2.99e-6$ (28225)</td>
</tr>
<tr>
<td>Tiny Images</td>
<td>4,000,000</td>
<td>4,500</td>
<td>$5.20e-4$ (14013)</td>
<td>$1.70e-5$ (26566)</td>
</tr>
</tbody>
</table>

**MNIST** Correctly classifying handwritten digits is a classic machine learning task, and MNIST is a benchmark dataset used to test new methods in machine learning [36]. MNIST training data contains 50,000 images of $28 \times 28 = 784$ pixels each. Similarity matrices formed from the digits are known to have low-rank structure, because there are only 10 different numerical digits. We set the kernel $\sigma$ equal to 50% of the maximum Euclidean distance between points.
**Salinas** While consumer cameras generally take 3 channel RGB images, hyperspectral cameras can take images at many spectral bands. The Salinas dataset is a hyperspectral image taken in 1998 by the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) over Salinas Valley, CA. The image is of size $512 \times 217$, over 204 spectral bands, and can be used to classify various areas of crops. Each pixel is assigned to one of 16 classes according to a groundtruth image. Classes represent crops such as broccoli or lettuce. We consider all pixels assigned to a nonzero class, for a total number of 54,129 data points. We set the kernel $\sigma$ equal to 10.

**Light Field** Light fields are 4-dimensional data sets describing both the intensity and directionality of light as it travels through a plane. We consider patches taken from the “chessboard” dataset of the Stanford Multi-Camera Array [37]. The samples are 85,265 vectorized 4-dimensional “patches,” each with $4 \times 4$ spatial resolution and $5 \times 5$ angular resolution for a total data matrix size of $85,265 \times 400$. We set the kernel $\sigma$ equal to 50% of the maximum Euclidean distance between points.

### 5.3 Implicit Kernel Matrices with oASIS-P

We now consider datasets that cannot be in memory. As such, the data set is split onto a variety of cores and the kernel matrix is approximated using oASIS-P as described in Figure 3.2. At this size, we only compare with uniform random sampling. We choose a Gaussian kernel for all data sets in this class.

Since a full representation of $G$ is no longer available, we estimate the approxi-
mation error as the Frobenius-norm discrepancy between 100,000 randomly sampled entries in $G$ and the corresponding entries in $\tilde{G}_k$. We consider the following datasets, run using OpenMPI with the Eigen C++ library [38] over 192 cores on the DaVinCi cluster at Rice University, with each 2.83 GHz processor core having 4GB of memory. Results at the largest sample sizes are shown for parallelized implicit matrices in Table 5.3. These sample sizes were chosen at the limit of available run time on the cluster when using uniform random sampling.

Two Moons This dataset is as described in Section 5.1, but the number of data points has been increased to 1,000,000 points. Since calculating the maximum Euclidean distance among all points becomes expensive, we set the kernel $\sigma$ equal to $0.5 \times \sqrt{3}$ based on other $\sigma$ found to provide good approximation error with smaller Two Moons sets, for any sampling method. For this experiment, oASIS was run to an error tolerance of $1e-4$, and random sampling was performed for $k \in \{20, 50, 100, 200, 250, 500, 1000\}$ samples.

Millions of Tiny Images To show the capability of oASIS to approximate kernel matrices over very large datasets, we select two random subsets of the 80 Million Tiny Images dataset [39], consisting of 1,000,000 and 4,000,000 RGB images of size $32 \times 32$. We compute the approximation over one color channel of the images as we consider the number of data points to be the prime focus of this experiment. Since calculating the maximum Euclidean distance among all points becomes expensive, we
set the kernel $\sigma$ equal to 20 based on other $\sigma$ found to provide good approximation error with smaller Tiny Images sets, for any sampling method.

## 5.4 Discussion of Results

For the experiments run, oASIS achieves lower approximation error than both uniform random sampling and Leverage scores. In addition, it is competitive with both $K$-means Nyström and Farahat’s method in terms of accuracy while having substantially faster run times.

In addition, oASIS’s strength as a deterministic method allows for $\ell$ to not be known, but instead for oASIS to run for a set length of time. For adaptive random schemes, as it is not known a priori how many columns either $K$-means or Leverage scores can sample, one must guess at the appropriate $K$ or $\ell$ to use given a certain amount of time (as long as $\Delta(i) > 0$). Our experiments found the appropriate parameters through exhaustive search, resetting the clock and increasing $\ell$ for each trial until the time limit was reached.

The primary advantage of oASIS over random selection schemes is approximation accuracy. In the Abalone example in Figure 5.1, uniform random sampling does not provide better accuracy as more columns are sampled, while oASIS continues to find columns that can add significantly to the accuracy of the approximation. In Figure 5.2, we observe that oASIS achieves exact matrix recovery with Two Moons at about 30 seconds with around 1000 columns sampled. This is an example of the
efficiency of oASIS’s column sampling. This efficient accuracy is necessary for the subsequent dimensionality reduction critical to most kernel machine learning tasks.

The primary advantage of oASIS over other adaptive schemes is its efficiency. oASIS saves both time and space. In the runtime results shown in Figure 5.2, we observe fast, accurate approximation of both Two Moons and Abalone with fewer columns sampled. For the BORG dataset, oASIS is second only to $K$-means, which is as expected given that BORG’s dataset containing of spherical clusters of equal variance exactly fits the inherent data model that $K$-means clusters. When the data do not fit that model, as in Abalone or Two Moons, we can see the efficiency and accuracy gains of oASIS. We observe in Table 5.1 that while $K$-means Nyström runs faster for a single sample size $\ell$, it needs to be run multiple times for consistency. For example, while running a single $K$-means Nyström approximation on Two Moons Gaussian with $n = 2000$ and $\ell = 450$ would take 0.38 seconds, the 10 runs used for consistency takes 3.8 seconds. Furthermore, $K$-means approximations performed for $\ell$ samples provides no remapping for any samples fewer than $\ell$, nor an index set $\Lambda$ of columns for CSS.

The advantages of oASIS-P over uniform random selection become clear in its application. First, oASIS-P can guarantee an invertible $W$. Uniform random sampling can not guarantee that $W$ will be invertible, so we must calculate $W^\dagger$ to compute $\tilde{G}$. Indeed, preliminary experiments frequently showed $W_k$ to be rank-deficient. This is most likely due to the birthday problem - as more columns are selected, the chances
that any two columns are of the same direction grows surprisingly fast. As oASIS can iteratively compute $W^{-1}$, it does not need to invert an $\ell \times \ell$ matrix. Note that 1% of a 1M point dataset still results in a $10k \times 10k$ $W$ matrix. Each of DaVinCi’s cores had 4GB of memory, and uniform random sampling became infeasible after approximately 4,500 columns. Second, while the complexity of oASIS-P appears much higher than uniform random sampling, in practice oASIS-P is faster than uniform random sampling. Computing columns from indices takes the same amount of time regardless of the method chosen, and in large data regimes communication of data vectors between cores becomes the computational bottleneck. Computing an iterative $W^{-1}$ is faster than computing $W^\dagger$, and so for very large data regimes oASIS-P becomes faster than uniform random sampling.

**Benefits for Sparse Matrices**

In addition to its low runtime complexity, oASIS is also capable of benefiting from sparse matrix structure. For such matrices, adaptive methods like the one in [11] requires the computation of $n \times n$ “residuals,” which may be dense even in the case that $G$ is extremely sparse. In contrast, oASIS requires only the storage of much smaller $\ell \times n$ matrices. This benefit of oASIS is highly relevant for extremely large datasets where sparse approximations to similarity matrices are formed using $K$-nearest-neighbor algorithms that only store the most significant entries in each matrix column. Further analysis is necessary, however, as fast approximation methods have
been developed specifically for sparse kernel matrices [40].
Chapter 6

Conclusion

In this paper, we introduced oASIS, a novel adaptive sampling algorithm for Nyström based low-rank approximation. oASIS combines the high accuracy of adaptive matrix approximations with the low runtime complexity and memory requirements of inexpensive random sampling schemes. Furthermore, we are able to achieve exact matrix recovery in an optimal number of columns selected. The speed and efficacy of the method is demonstrated by accurately approximating large matrices using a single processor. In addition, we parallelized oASIS so it could be run over data matrices of arbitrary size. This allows oASIS to be the only adaptive greedy method available in large data regimes. In addition, numerical experiments show oASIS to be competitive with random schemes at this level, in both accuracy and speed. Using a data set of 1 million examples we are able to achieve 1% of the approximation error of random sampling methods. oASIS has already been applied to sparse subspace clustering, and future work will focus on evaluating the efficacy of oASIS for other machine learning tasks, such as manifold learning and spectral clustering, for other applications requiring PSD matrix approximations, such as molecular relaxation, and for outlier detection.
Bibliography


