oASIS: Adaptive Column Sampling for Kernel Matrix Approximation

Raajen J. Patel[*], **Thomas Goldstein**[†], **Eva L. Dyer**[‡], **Azalia Mirhoseini**[*], **Richard G. Baraniuk**[*]
Tech Report TREE1402

Abstract

Computing with large kernel or similarity matrices is essential to many state-of-the-art machine learning techniques in classification, clustering, and dimensionality reduction. The cost of forming and factoring these kernel matrices can become intractable for large datasets. We introduce an an adaptive column sampling technique called Accelerated Sequential Incoherence Selection (oASIS) that samples columns without computing the entire kernel matrix. Numerical experiments demonstrate that oASIS has performance comparable to state-of-the-art adaptive sampling methods at a fraction of the cost.

1 Introduction

Many machine learning and data analysis frameworks 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], or nonlinear dimensionality reduction [2, 3] all require computing and storing an $n \times n$ kernel matrix, where n is the number of examples (points) in the dataset.

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* [4]. Methods for sampling the span of a matrix are broadly referred to as *column subset selection* (CSS) methods and have been applied successfully in applications ranging from image segmentation [5] to matrix factorization [6].

In general, it is not possible to determine which columns to select from a data matrix without explicitly forming the candidate columns (i.e., computing the distance between one example and the rest of the dataset). However, in the case of symmetric kernel matrices,

- oASIS does not use information about a column until after it has been sampled. For this reason, only the submatrix of sampled columns must be computed/stored.
- The total runtime of oASIS scales *linearly* with the matrix dimension n. oASIS is typically orders of magnitude faster than other greedy methods that have $O(n^2)$ runtime.

oASIS provides a tractable solution for approximating extremely large kernel matrices where other adaptive methods simply cannot be applied. In numerical experiments, we demonstrate that oASIS provides accuracy comparable to the best existing adaptive methods with dramatically reduced complexity and memory requirements.

This paper is organized as follows. In Section 2, we introduce the Nyström method and survey existing sampling methods. In Section 3, we introduce oASIS. In Section 4, we use 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].

[*]Department of Electrical and Computer Engineering, Rice University, Houston, Texas USA

[†]Department of Computer Science, University of Maryland, College Park, Maryland USA

[‡]Department of Physical Medicine and Rehabilitation, Rehabilitation Institute of Chicago, Northwestern University, Chicago, Illinois USA.

2 Background

In this section we describe the Nyström method and its variants and applications.

2.1 Notation

We will write matrices G and vectors x in upper and lowercase script, respectively. We also use $A^\dagger$ to denote the Moore-Penrose pseudo-inverse of A. We represent the element-wise product of matrices A and B and $A \circ B$, and $\mathrm{colsum}(A)$ denotes a row vector containing the sums of the columns of A. When describing algorithms, we use "Matlab" style indexing in which $G(i, j)$ denotes the (i, j)th entry of the matrix G, and $G(:, j)$ denotes its jth column.

2.2 The Nyström Method

Williams and Seeger presented the Nyström method in 2001 to improve the speed of kernel-based methods for classification [1]. 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. There is some matrix $X \in \mathbb{R}^{r \times n}$ such that $G = X^T X$. Suppose we sample $k \leq r$ columns of G and collect them into a matrix C_k. To simplify notation, we assume, without loss of generality, that we have sampled the k left-most columns of G. We can then form the partition $X = [X_1 \;\; X_2]$, where X_1 contains the first k columns of X, and X_2 contains the remaining columns. We can write G as

$$G = \begin{bmatrix} X_1^T \\ X_2^T \end{bmatrix} [X_1 \quad X_2] = \begin{bmatrix} W_k & X_1^T X_2 \\ X_2^T X_1 & X_2^T X_2 \end{bmatrix} \tag{1}$$

where $W_k = X_1^T X_1$ is a $k \times k$ symmetric matrix, and $C_k = \begin{bmatrix} W_k & X_1^T X_2 \end{bmatrix}^T$ is the $n \times k$ sampled columns. Then the Nyström approximation of G is now defined as

$$G \approx \widetilde{G}_k = C_k W_k^\dagger C_k^T. \tag{2}$$

An approximate singular value decomposition (SVD) of G can then be obtained from the SVD of the smaller matrix W_k, which is written $W_k = U_W \Sigma_W U_W^T$. The singular values of the approximation $\widetilde{G}_k$ are simply a scaled multiple of Σ_W, and the singular vectors are given by [1]

$$\widetilde{U} = \sqrt{\frac{k}{n}} C U_W \widetilde{\Sigma}_W^{-1}.$$

Since W is $k \times k$, this computation is much faster than the full $n \times n$ SVD of G, where the complexity of the SVD step reduces from $\mathcal{O}(n^3)$ to $\mathcal{O}(k^3)$ with $k \ll n$.

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

2.3 Column Sampling Methods

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 under various norms can be found in [7]. Uniform random sampling methods can be improved using weighted probability distributions that consider the scaling of the columns of G [4]. Scaling can be done in $O(n^2)$ time, assuming the matrix is precomputed.

Non-deterministic Adaptive Sampling: Recently, leverage scores have been used as a distribution over the column draw [7]. 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. Scores are expensive to compute exactly, and require the low-rank approximate SVD of G to be pre-computed at $O(n^2)$ cost.

Deterministic Adaptive Sampling: Early adaptive methods use exhaustive search to find columns that minimize the Frobenius norm error $\|G - \widetilde{G}_k\|_F$ [8]. A more efficient adaptive scheme due to

Farahat [9] builds a Nyström approximation by selecting columns sequentially using a matrix of "residuals." On 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 in [10].

3 Accelerated Sequential Incoherent Selection (oASIS)

In this section, we introduce a new adaptive sampling method for column subset selection called oASIS and analyze its complexity.

3.1 Sequential Incoherence Selection

We now address the question of how to build a Nyström approximation by sequentially choosing columns from G. Suppose we have already formed a k-column Nyström approximation to G. Our goal is to select a new column that will improve the approximation. If a new column lies in the span of the columns we have already sampled, then adding this column has no effect on the Nyström approximation. Ideally, we would like to quantify how much each candidate column will change the Nyström approximation and select the column that will produce the most significant impact on $\widetilde{G}_k$. As this column would ideally not lie in the span of the already selected columns, we say the new column is *incoherent* with those already selected. While it is impossible to predict how much each column will impact $\widetilde{G}$ without examining all of the candidate columns, we show below that it is possible to calculate the impact of a column on the upper-left block of $\widetilde{G}$ even before that column has been selected. This provides a proxy measure of the incoherence.

Let $\widetilde{G}_k$ denote the k-column Nyström approximation of G. Recall that the matrix G can be written as $X^T X$, and that X_k denotes the first k columns of X. The upper-left $(k+1) \times (k+1)$ block of G can be written in terms of the matrix X_k as

$$X_k^T X_k = \begin{bmatrix} W_k & b_{k+1} \\ b_{k+1}^T & d_{k+1} \end{bmatrix} \tag{3}$$

where b_{k+1} contains the first k entries in column $G(:, k+1)$, and d_{k+1} denotes the diagonal entry $G(k+1, k+1)$.

We now consider the expansion of the Nyström approximation in (2). The upper left $k \times k$ block of $\widetilde{G}_k$ is simply $X_k^T X_k W_k^\dagger X_k^T X_k = X_k^T X_k$. Similarly, the upper left $(k+1) \times (k+1)$ block of $\widetilde{G}_k$ can be written in terms of $X_k^T X_k$ as follows:

$$\begin{bmatrix} X_k^T X_k & b_{k+1} \\ b_{k+1}^T & d_{k+1} \end{bmatrix} \begin{bmatrix} W_k^\dagger & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} X_k^T X_k & b_{k+1} \\ b_{k+1}^T & d_{k+1} \end{bmatrix} = \begin{bmatrix} X_k^T X_k & b_{k+1} \\ b_{k+1}^T & b_{k+1}^T W_k^\dagger b_{k+1} \end{bmatrix}. \tag{4}$$

If the Nyström approximation $\widetilde{G}_k$ is an accurate approximation to G, then the upper left block (4) of $\widetilde{G}_k$ should match the upper-left block (3) of G. The discrepancy between these two blocks is given by $|d_{k+1} - b_{k+1}^T W_k^\dagger b_{k+1}|$. In the event that $d_{k+1} - b_{k+1}^T W_k^\dagger b_{k+1} = 0$, the matrix $W_{k+1}^\dagger$ behaves similarly to the matrix $W_k^\dagger$, and the new column does not impact the upper-left block of $\widetilde{G}_k$. However, if $|d_{k+1} - b_{k+1}^T W_k^\dagger b_{k+1}|$ is large, then this column is guaranteed to have a substantial impact on the Nyström approximation.

Based upon this observation, we develop the following sampling method for sequential incoherent selection (SIS). We assume the process has been seeded with a small set Ω_k of k column indices chosen at random (we suggest starting with $|\Omega_k| = 10$). Columns are then selected to be included in the approximation as follows:

1. Permute the rows and columns of G such that the first k columns correspond to the columns that we have already sampled. Form $W_k^\dagger$, the pseudoinverse of the upper left $k \times k$ block of G.
2. Let b_i denote the first k entries of column i, and let d_i denote the diagonal entry in this column. For each unsampled column, calculate the lower bound Δ_i on how much it changes

the Nyström approximation:

$$\Delta_i = d_i - b_i^T W_k^\dagger b_i.$$

3. Select the un-sampled column that maximizes $|\Delta_i|$ and add it to the Nyström approximation.
4. If the selected value of $|\Delta_i|$ is smaller than a user set threshold, then terminate. Otherwise, let $k \leftarrow k+1$, and return to Step 1.

3.2 oASIS

A naive implementation of the method in Section 3.1 is inefficient because each step requires a matrix inversion to form $W_{k+1}^\dagger$ in addition to calculating the errors Δ_i. Fortunately, this can be done efficiently by updating the results from the previous step using block matrix inversion formulas. This method is called oASIS.

We first consider the calculation of $W_{k+1}^\dagger$ after column $k+1$ is added to the approximation. We will assume throughout the rest of this section that W_{k+1} is invertible and thus $W_{k+1}^\dagger = W_{k+1}^{-1}$. We shall see later on that our column selection rule guarantees the invertibility of W_k. Using the notation of Section 3.1, we let b_k denote the first k rows of column $G(:, k+1)$ and d_k denote its diagonal. Using a block inversion formula, we get

$$W_{k+1}^{-1} = \begin{bmatrix} W_k & b_{k+1} \\ b_{k+1}^T & d_{k+1} \end{bmatrix}^{-1} = \begin{bmatrix} W_k^{-1} + s_{k+1} q_{k+1} q_{k+1}^T, & -s_{k+1} q_{k+1} \\ -s_{k+1} q_{k+1}^T, & s_{k+1} \end{bmatrix} \tag{5}$$

where $s_{k+1} = (d_{k+1} - b_{k+1}^T W_k^{-1} b_{k+1})^{-1} = \Delta_{k+1}^{-1}$ is the (scalar valued) Schur complement and $q_{k+1} = W_k^{-1} b_{k+1}$ is a column vector. This update formula allows 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 Δ_{k+1} (the Schur complement) is non-zero, which is guaranteed by our sampling rule: the algorithm terminates if $\Delta_{k+1} = 0$ in which case our approximation is exact.

We now consider the calculation of $\Delta_i = d_i - b_i^T W_k^\dagger b_i$ for all i. Note that on step k of the method, we have $C_k^T = [b_1, b_2, \cdots, b_N]$. We can evaluate all values of $b_i^T W_k^\dagger b_i$ 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 on iteration k, the matrix $R_{k+1} = W_{k+1}^{-1} C_{k+1}^T$ needed on the next iteration is obtained by applying (5) to C_{k+1}^T to get

$$R_{k+1} = W_{k+1}^{-1} C_{k+1}^T = W_{k+1}^{-1} \begin{bmatrix} C_k^T \\ c_{k+1}^T \end{bmatrix} = \begin{bmatrix} R_k + s_{k+1} q_{k+1} (q_{k+1}^T C_k^T - c_{k+1}^T) \\ s_{k+1} (-q_{k+1}^T C_k^T + c_{k+1}^T) \end{bmatrix}. \tag{6}$$

Equation (6) 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 Algorithm 1.

Algorithm 1 is initialized by choosing a small random subset of 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 Δ is formed by computing

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

where "$\circ$" represents entry-wise multiplication, and "colsum" returns the sums of the columns of its argument. Next, the largest entry in Δ is found, and its index is used to select the next column from G. The update formulas (5) and (6) are then used to form the matrices W_{k+1}^{-1} and R_{k+1} required for the next iteration.

3.3 Complexity of oASIS

The rate-limiting step of Algorithm 1 is the computation of R_{k+1} by updating R_k. Eq. (6) allows this to be performed by sweeping over the entries of R_k, which has dimensions $k \times n$. The complexity of a single iteration is thus $O(kn)$. If ℓ columns are sampled in total, then $\sum_{k=1}^{\ell} kn = \frac{1}{2}\ell(\ell+1)n$

Algorithm 1 oASIS

Inputs: A symmetric matrix G,
The diagonal elements of G, stored in d,
The maximum number of columns to sample, K,
The number of columns to initialize the algorithm $k < K$,
A non-negative stopping tolerance, ϵ.
Outputs: The sampled columns C,
The inverse of the sampled rows W^{-1}.
Initialize: Choose a vector $\Omega \in [1, N]^K$ of k random starting indices.
$C_k = G(:, \Omega)$
$W_k^{-1} = G(\Omega, \Omega)^{-1}$
$R_k = W_k^{-1} C_k^T$
while $k < K$ **do**
  $\Delta = d - \text{colsum}(C_k \circ R_k)$
  $i = \arg\max_{j \notin \Omega} |\Delta(j)|$
  if $|\Delta(i)| < \epsilon$ **then**
    return
  end if
  $b_k = G(\Omega, i)$
  $d_{k+1} = d(i)$
  $s_{k+1} = 1/\Delta(i)$
  $q_{k+1} = R(:, i)$
  $C_{k+1} = [C_k, G(:, i)]$
  Form W_{k+1}^{-1} using Eq. (5)
  Update R_{k+1} using Eq. (6)
  $k \leftarrow k + 1$
  $\Omega \leftarrow \Omega \cup \{i\}$
end while

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 [9] , which requires the computation of $n \times n$ residual matrices at each stage resulting in $O(\ell n^2)$ complexity. The method in [7] requires the approximate SVD of G, which also requires iterations of cost $O(n^2)$ over dense matrices. The low complexity of oASIS makes it practical for extremely large matrices where other adaptive sampling schemes are intractable.

4 Numerical Experiments

To evaluate the performance of oASIS, we measure the accuracy of Nyström approximations using small and large datasets. For each dataset $\{x_i\}_{i=1}^n$, we consider: (i) Gaussian kernel matrices where $G(i, j) = \exp(\|x_i - x_j\|_2^2/\sigma^2)$ and for small datasets we also consider (ii) diffusion distance matrices $M = D^{-1/2} G D^{-1/2}$ where G is a Gaussian kernel matrix and D is a diagonal matrix containing the row sums of G.

We compare oASIS to the following state-of-the-art CSS methods: uniform random sampling, leverage scores [7] (Section 2.3), and Farahat's greedy update method [9] (Section 2.3). For the large dataset we consider only oASIS and uniform random sampling, as the other adaptive methods become intractable when the matrix becomes too large to explicitly store. Specific datasets and experiments are described below.

4.1 Small datasets: Two Moons and Abalone

Convergence curves are generated by forming $\widetilde{G}_k$ for increasing k, and then calculating the approximation error defined by $\|\widetilde{G}_k - G\|_F / \|G\|_F$.

Two Moons: A common synthetic dataset for clustering algorithms, this data consist of 2-dimensional points arranged in two interlocking moons. The data matrix is 2000×2. We set the kernel σ equal to 5% of the maximum Euclidean distance between points.

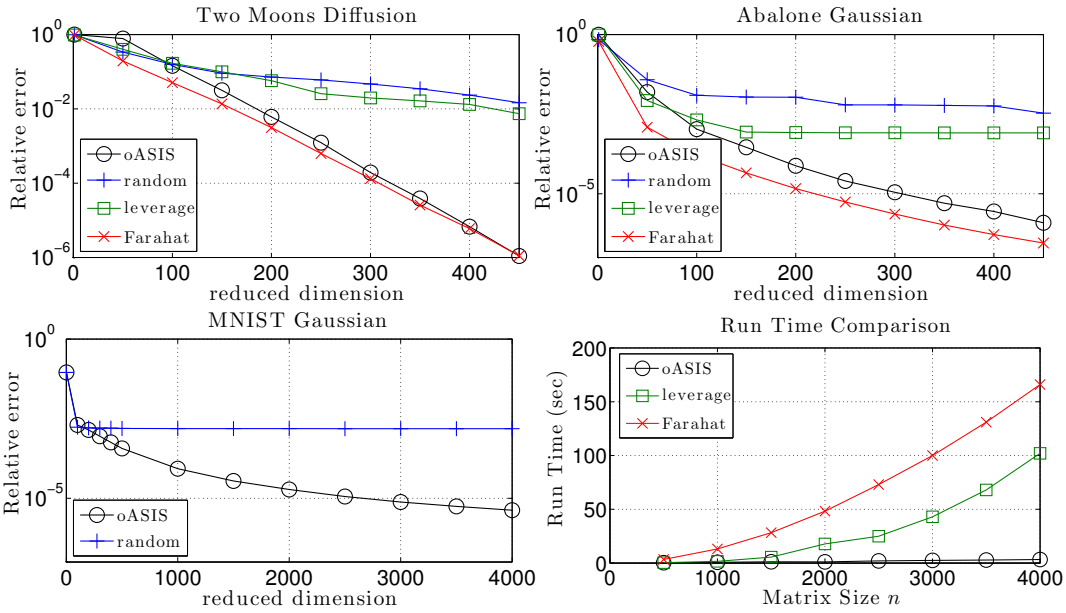

Figure 1: Nyström approximation error curves for selected experiments, comparing our proposed oASIS with state-of-art random and adaptive methods. See Section 4 for details. In addition, we also plot a run time comparison of oASIS with other adaptive methods. Note that oASIS scales well to large problem sizes due to its low runtime complexity (see Section 3.3).

Table 1: Normalized Nyström approximation errors (and column selection runtime in seconds) for a variety of matrices. For each problem we present the results for both Gaussian maps (top) and diffusion maps (bottom). For "small" problems we report results after sampling 450 columns. For MNIST we report results with 4000 columns.

Problem	n	oASIS	Random	Lev. Scores	Farahat
Two Moons	2,000	$1.00e{-}6$ (2.34)	$2.57e{-}3$ (0.01)	$1.08e{-}3$ (16.64)	$8.30e{-}7$ (57.96)
		$1.10e{-}6$ (2.07)	$1.47e{-}2$ (0.04)	$7.44e{-}3$ (18.26)	$1.11e{-}6$ (58.28)
Abalone	4,177	$1.23e{-}6$ (4.61)	$3.40e{-}3$ (0.01)	$8.17{-}4$ (129.047)	$2.80{-}7$ (243.73)
		$1.62e{-}6$ (4.59)	$3.05e{-}1$ (0.01)	$4.11e{-}1$ (130.66)	$5.61e{-}7$ (251.24)
MNIST	50,000	$7.42e{-}6$ (16387)	$1.54e{-}3$ (1861)	-	-
		-	-	-	-

Abalone: A dataset collected by marine scientists consisting of physical characteristics of abalone [11]. The data matrix is 4177×8. We set the kernel σ equal to 5% of the maximum Euclidean distance between points.

4.2 Large dataset: MNIST digits

As the resulting G would be too large to explicitly calculate and store, the approximation error is estimated using a random sampling of 100,000 entries of G, and computing the Frobenius-norm discrepancy between the corresponding entries in $\widetilde{G}_k$.

MNIST: A benchmark dataset for classification methods [12], MNIST contains 50,000 images of handwritten digits with $28 \times 28 = 784$ 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 σ equal to 50% of the maximum Euclidean distance between points.

4.3 Discussion of Results

Selected convergence curves (normalized error vs. number of columns sampled) are shown in Figure 1. Figure 1 also presents a plot of column selection runtime vs. matrix size for a variety of methods. Results at the largest sample sizes are shown in Table 1. For the experiments run, oASIS

achieves lower approximation error than both uniform random sampling and leverage scores. In addition, it is competitive with Farahat's method in terms of accuracy, while having substantially faster run times.

We emphasize that the primary advantage of oASIS over other adaptive schemes is its speed. In fact, as discussed in Section 3.3, the runtime of oASIS is linear in the matrix dimension. This gives oASIS a considerable advantage over the other adaptive schemes, which require $O(n^2)$ runtime. For this reason, oASIS easily scales up to extremely high dimensional problems where other adaptive methods are intractable.

5 Conclusions

In this paper, we have introduced oASIS, a new adaptive sampling algorithm that can be used in the Nyström method for 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. The speed and efficacy of the method was demonstrated by accurately approximating large matrices using a single processor. Future work will focus on parallel and distributed implementations, and evaluating the efficacy of oASIS for specific tasks, such as manifold learning and spectral clustering.

References

[1] C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688, Denver, 2001. MIT Press.

[2] R. R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1), 2006.

[3] A. Talwalkar, S. Kumar, M. Mohri, and H. Rowley. Large-scale svd and manifold learning. *Journal of Machine Learning Research*, 14:3129–3152, 2013.

[4] P. Drineas and M. W. Mahoney. On the Nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

[5] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225, 2004.

[6] L. W. Mackey, A. Talwalkar, and M. I. Jordan. Divide-and-conquer matrix factorization. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, editors, *NIPS*, pages 1134–1142, 2011.

[7] A. Gittens and M. W. Mahoney. Revisiting the Nyström method for improved large-scale machine learning. In *International Conference on Machine Learning (3)*, pages 567–575. JMLR.org, 2013.

[8] A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In P. Langley, editor, *Proceedings of the International Conference on Machine Learning*, pages 911–918, San Francisco, 2000. Morgan Kaufmann Publishers.

[9] A. K. Farahat, A. Ghodsi, and M. S. Kamel. A novel greedy algorithm for Nyström approximation. In G. J. Gordon, D. B. Dunson, and M. Dudk, editors, *Artificial Intelligence and Statistics*, pages 269–277. JMLR.org, 2011.

[10] A. Deshpande, L. Rademacher, S. Vempala, and G. Wang. Matrix approximation and projective clustering via volume sampling. In C. Stein, editor, *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 1117–1126, New York, NY, USA, 2006. ACM.

[11] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[12] Y. LeCun, L. D. Jackel, L. Bottou, A. Brunot, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of learning algorithms for handwritten digit recognition. In F. Fogelman and P. Gallinari, editors, *International Conference on Artificial Neural Networks*, pages 53–60, Paris, 1995. EC2 & Cie.