Unsupervised Feature Selection by Heuristic Search with Provable Bounds on Suboptimality

Hiromasa Arai
Dept. of Computer Science
University of Texas (Dallas)
800 W Campbell Road
Richardson, TX 75080
Hiromasa.Arai@utdallas.edu

Crystal Maung
Dept. of Computer Science
University of Texas (Dallas)
800 W Campbell Road
Richardson, TX 75080
Crystal.Maung@gmail.com

Ke Xu
Dept. of Computer Science
University of Texas (Dallas)
800 W Campbell Road
Richardson, TX 75080
ke.xu5@utdallas.edu

Haim Schweitzer
Dept. of Computer Science
University of Texas (Dallas)
800 W Campbell Road
Richardson, TX 75080
hschweitzer@utdallas.edu

Abstract

Identifying a small number of features that can represent the data is a known problem that comes up in areas such as machine learning, knowledge representation, data mining, and numerical linear algebra. Computing an optimal solution is believed to be NP-hard, and there is extensive work on approximation algorithms. Classic approaches exploit the algebraic structure of the underlying matrix, while more recent approaches use randomization. An entirely different approach that uses the A* heuristic search algorithm to find an optimal solution was recently proposed. Not surprisingly it is limited to effectively selecting only a small number of features. We propose a similar approach related to the Weighted A* algorithm. This gives algorithms that are not guaranteed to find an optimal solution but run much faster than the A* approach, enabling effective selection of many features from large datasets. We demonstrate experimentally that these new algorithms are more accurate than the current state-of-the-art while still being practical. Furthermore, they come with an adjustable guarantee on how different their error may be from the smallest possible (optimal) error. Their accuracy can always be increased at the expense of a longer running time.

1 Introduction

Feature selection is a standard dimensionality reduction technique. Given data items described in terms of n features, the goal is to select $k<n$ features such that the reduced k dimensional vectors are useful for the task at hand. In supervised feature selection the features are selected for predicting values associated with each data item. These can be classification labels (e.g., (Guyon and Elisseeff 2003)), or multi-valued regression (e.g., (Maung and Schweitzer 2015)). In unsupervised feature selection the standard criterion is to select k features that can be used to approximate all n features. There are two common approaches for computing the unsupervised selection. The first is to partition the feature space into k clusters and then selects a representative from each cluster (e.g., (Dy et al. 2003; Li et al. 2012)). This assumes a feature to be redundant if another feature similar to it has already been selected. The second common approach assumes that a feature is redundant if it can be approximated by a linear combination of other features that have already been selected (e.g., (Jimenez-Rodriguez, Arzuaga-Cruz, and Velez-Reyes 2007; Wei and Billings 2007; Drineas, Lewis, and Paschou 2010; Dasgupta et al. 2007; Wang et al. 2015)). Thus, if the goal is to select features for a nearest neighbor algorithm the first approach seems most appropriate. But if the selected features are to be used in a Gaussian classifier, the second approach seems more appropriate (Duda, Hart, and Stork 2001). In this paper we consider only the second approach, which is formally described as follows.

Let X be an $m \times n$ data matrix of m items (rows), each described in terms of n features (columns). We wish to approximate X by a linear combination of k of its columns: $X \approx S_k A$. The matrix $S_k = (x_{s_1}, \ldots, x_{s_k})$ is formed by the selected columns, and A is the coefficients matrix. In Frobenius norm the approximation error is:

$$\text{Err}(S_k) = \min_A \|X - S_k A\|_F^2 \qquad (1)$$

Finding S_k that minimizes (1) is also known as the "*Column Subset Selection Problem*" (CSSP) in numerical linear algebra. Previous studies of the CSSP with the Frobenius norm error include (Wei and Billings 2007; Golub and Van-Loan 1996; Boutsidis, Mahoney, and Drineas 2009; Çivril and Magdon-Ismail 2012; Çivril 2014). Obtaining approximations in other norms, such as the spectral norm or the l_1 norm is considered to be harder (e.g., (Tropp 2004)).

Copyright © 2016, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Unsupervised feature selection gives a sparse approximation to the data matrix, and has found applications in many areas. These include the computation of stable and rank revealing QR factorizations (e.g. (Golub and Van-Loan 1996; Gu and Eisenstat 1996; Boutsidis, Mahoney, and Drineas 2009)), feature selection in machine learning (e.g. (Drineas, Lewis, and Paschou 2010; Maung and Schweitzer 2013; Wei and Billings 2007), remote sensing (e.g., (Jimenez-Rodriguez, Arzuaga-Cruz, and Velez-Reyes 2007; Wang et al. 2015)), and data mining and knowledge representation (e.g. (Dasgupta et al. 2007; Kumar, Mohri, and Talwalkar 2012; Drineas, Lewis, and Paschou 2010)).

Computing the selection S_k that minimizes (1) is believed to be NP-hard (Çivril and Magdon-Ismail 2009; Çivril 2014). Most current algorithms compute an approximate solution, and the current state-of-the-art is reviewed in Section 2. The only exception we are aware of is a recent algorithm (Arai, Maung, and Schweitzer 2015) that uses the A* algorithm to compute a guaranteed optimal solution. The algorithm gives huge runtime improvements over exhaustive search. However, as implied by the NP-hardness of the problem, it can compute effectively only a small number of features from small to moderate size datasets. Therefore, this algorithm is not a competitor to the current state-of-the-art approximation algorithms.

In this paper we show that heuristics similar to those proposed in (Arai, Maung, and Schweitzer 2015) can be used to design much faster algorithms, that are more accurate than the current state-of-the-art. These new algorithms are not guaranteed to produce the optimal solution, but they can be applied to very large datasets and effectively select hundreds of features. Experimental evaluation shows that these algorithms are almost always more accurate than all other practical algorithms we are aware of. Furthermore, they come with an adjustable guarantee on how different their error may be from the smallest possible (optimal) error.

The paper is organized as follows. The current state-of-the-art is reviewed in Section 2. The algorithm of (Arai, Maung, and Schweitzer 2015) which forms the basis to our results is described in Section 2.3. Our main result, which extends this to the Weighted A* algorithm is described in Section 3. Experimental results are shown in Section 4.

2 The current state-of-the-art

Let S_k^* denote a (best) selection of k columns that minimizes the error in (1), and let $\text{Err}(S_k^*)$ be the corresponding (minimal) error. Recent studies suggest that computing S_k^* is most likely NP-hard (Çivril and Magdon-Ismail 2012; Çivril 2014). The eigenvalues of the matrix XX^T can be used to obtain bounds on $\text{Err}(S_k^*)$. Let Err_k^* denote the error of the best approximation of X in terms of a rank-k matrix. Then the following inequalities hold:

$$\text{Err}_k^* \leq \text{Err}(S_k^*) \leq (k+1)\text{Err}_k^* \qquad (2)$$

Calculating Err_k^* is easy, since the columns of the best rank-k matrix are the k dominant eigenvectors of XX^T, scaled by the corresponding eigenvalues. From this it follows that:

$$\text{Err}_k^* = \sum_{t=k+1}^{m} \lambda_t$$

where λ_t is the t-largest eigenvalue of XX^T. This fact, and the left-hand side inequality in (2) follow from the Courant Fischer theorem (Golub and Van-Loan 1996). The right-hand side inequality is a recent result (Deshpande et al. 2006; Deshpande and Rademacher 2010). The same references also show that the upper bound in (2) is tight in the following sense: If Err_k^* is given, then the constant $k+1$ in (2) is the best possible.

2.1 Algorithms from numerical linear algebra

Subset selection algorithms were developed in numerical linear algebra for computing a stable and rank-revealing QR factorization. The algorithm that we call **QRP** performs column subset selection by computing the QR factorization of X with column pivoting. It is numerically stable and efficient, but lacks in accuracy when compared to some of the more sophisticated algorithms. See (Golub and Van-Loan 1996; Gu and Eisenstat 1996) for details. A more accurate variant of this algorithms applies the QRP to the "leverage" vectors. These are truncated right singular vectors of the SVD decomposition of X. It was invented by Golub, Klema and Stewart and we refer to it as the **GKS**. See (Golub and Van-Loan 1996) for details.

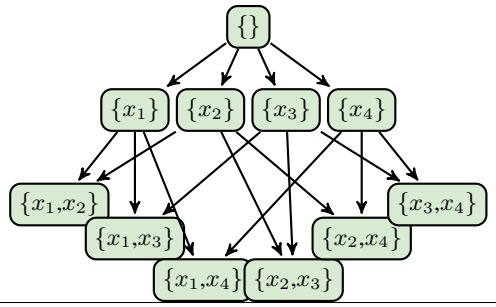
Among the many other algorithms that were developed in numerical linear algebra the algorithm of Gu and Eisenstat (Gu and Eisenstat 1996) that we call **GE** is considered the most accurate. It typically starts with a selection computed by the **QRP**, and then updates the selection, swapping columns to increase the "volume" (product of singular values squared) of the selection matrix.

2.2 Randomized algorithms

Randomized algorithms come with a small probability of failure. Some are very fast, and with high probability select a subset of size $O(k \log k)$ that can be shown to contain an accurate selection of size k. They can be converted into algorithms that select exactly k vectors by the approach of (Boutsidis, Mahoney, and Drineas 2009). The idea is to apply a randomized algorithm in the first stage to select more than k vectors, and then apply another deterministic algorithm on the (properly weighted) selection to obtain a reduced selection of size k.

The randomized algorithm described in (Frieze, Kannan, and Vempala 2004) selects columns with probabilities proportional to their squared norms. Using it for the randomized phase and the **GE** for the deterministic phase gives the algorithm that we call **FKV-GE**.

The randomized algorithm described in (Boutsidis, Mahoney, and Drineas 2009) selects columns with probability proportional to the squared norm of the corresponding truncated singular vectors of the SVD decomposition (the leverage space). Using it for the randomized phase and the **GE** for the deterministic phase gives the algorithm that we call **LEV-GE**.



0. Put the root node into F.
1. While F is nonempty and no solution found:
  1.1 Pick n_i with the smallest $f'(n_i)$ from F.
  1.2 If n_i has k columns return it as the solution.
  1.3 Otherwise:
    1.3.1 Add n_i to C.
    1.3.2 Examine all children n_j of n_i.
    1.3.2.1 If n_j is in C or F do nothing.
    1.3.2.2 Otherwise put n_j in F.

Figure 1: Example of the subsets graph and the generic heuristic search algorithm. The algorithm maintains the fringe list F and the closed nodes list C. Several choices of $f'(n_i)$ are discussed in the text.

An algorithm that samples k-tuples of columns with probability proportional to their "volume" (product of singular values squared) was developed in (Deshpande et al. 2006; 2006; Deshpande and Rademacher 2010; Guruswami and Sinop 2012). Its accuracy can be shown to meet the upper bound specified in (2).

2.3 Heuristic search for the optimal solution

The algorithms reviewed in Sections 2.1,2.2 are fast, but typically do not produce an optimal solution. A recent study (Arai, Maung, and Schweitzer 2015) has shown how to model the CSSP as a graph search, and then use the classic A* algorithm (Hart, Nilsson, and Raphael 1968; Pearl 1984) to find the optimal solution. Their algorithm, that we call **CSSP-A***, is guaranteed to find the optimal solution, runs much faster than exhaustive search, but much slower than the other algorithms described in Section 2. In particular, it can only be used when k is very small and only on small to moderate size datasets.

The CSSP-A* operates on a graph, where nodes correspond to column subsets. There is an edge from subset s_1 to subset s_2 if adding one column to s_1 creates s_2. We call the node s_r the root of the subgraph consisting of all nodes s satisfying $s \supset s_r$. We call the empty set the root of the graph. An example is shown at the top of Figure 1. The algorithm shown at the bottom of Figure 1 is the classic A* algorithm applied to the subsets graph. The description leaves the value of $f'(n_i)$ unspecified.

2.4 A* and the Weighted A* algorithms

The standard formulation of the A* algorithm (e.g., (Pearl 1984; Russell and Norvig 2010)) involves the three functions $f(n_i), g(n_i), h(n_i)$, that are defined for each node n_i. The value of $g(n_i)$ is the length of the path from the root to n_i, the value of $h(n_i)$ is an estimate of the distance between n_i and the goal, and $f(n_i)$ is calculated as $f(n_i) = g(n_i) + h(n_i)$. Under some restrictions on $h(n_i)$ (it should be "consistent"), the value of $f(n_i)$ is non decreasing along any path. This property can be used to prove the optimality of the A* algorithm.

A standard approach for accelerating the running time of the A* algorithm is the "Weighted A* heuristic". Instead of taking $f'(n_i) = g(n_i) + h(n_i)$, the Weighted A* algorithm increases the weight of the $h(n_i)$ component by using $f'(n_i) = g(n_i) + (1+\epsilon)h(n_i)$. It can be shown ((Pearl 1984)) that with this choice of $f'(n_i)$ the A* algorithm gives a solution with error value g_{**} satisfying: $g_{**} \leq (1+\epsilon)g_*$, where g_* is the error of the optimal solution. Experimental results and theoretical analysis (e.g., (Pearl 1984; Likhachev, J.Gordon, and Thrun 2003)) show that the Weighted A* algorithm runs much faster than the A*. Observe that the bound on g_{**} is multiplicative.

3 Weighted A* algorithms for the CSSP

In this section we prove a theorem and use it to derive weighted A* algorithms for the CSSP. The theorem is stated in terms of the functions $f(n_i), g(n_i), h(n_i)$, that are defined for each node n_i. These functions are defined in Section 3.1 in terms of eigenvalues of a certain matrix. In the beginning of this section we give a non-computational definition that is sufficient for stating and proving the theorem.

Let n_i be a node in the subsets graph. Let the corresponding column subset S_i have j columns. Define:
$g(n_i) \triangleq$ Error of estimating X using the j columns in S_i.
$f(n_i) \triangleq$ Error of estimating X using the j columns in S_i and additional "best possible" $k-j$ vectors.
$h(n_i) \triangleq g(n_i) - f(n_i)$.
Observe that by definition $g(n_i)$ is monotonically decreasing along any path, and $f(n_i)$ is monotonically increasing along any path. From this it follows that $h(n_i)$ is monotonically decreasing along any path. In addition, at each leaf node q (a goal node of the search) j=k, so that $f(n_q) = g(n_q)$, and $h(n_q) = 0$.

Let n_* be an optimal solution node so that $g(n_*)$ is the smallest possible error in Equation (1). Let n_{**} be a solution obtained by the algorithm in Figure 1 using a particular choice for $f'(n_i)$ with the (not necessarily optimal) error $g(n_{**})$. It was proved in (Arai, Maung, and Schweitzer 2015) that if $f'(n_i) = f(n_i) = g(n_i) - h(n_i)$ then $g(n_{**}) = g(n_*)$ (i.e, the solution found by the algorithm is optimal). As in Section 2.4, this suggests that adding weight to $h(n_i)$ would give a weighted A* algorithm with an improved running time. Suppose instead of $f'(n_i)$=$g(n_i)$–$h(n_i)$ we take:

$$f'(n_i) = g(n_i) - (1+\epsilon)h(n_i) \quad (3)$$

Observe that in this case $f'(n_i)$ is monotonically increasing, and it is possible to give proof similar to the one in (Pearl 1984) which shows that $g(n_{**}) \leq (1+\epsilon)g(n_*)$. We skip the details of this case because we have found experimentally that this algorithm is impractical. Its running time is even

worse than that of the CSSP-A*. Other attempts of defining a monotonically increasing $f'(n_i)$ in terms of $g(n_i), h(n_i)$ also failed to improve the running time. As we describe next, the solution we have found does not have $f'(n_i)$ monotonically increasing or decreasing, but it still gives exact bounds on suboptimality.

Theorem 1: Let n_* be an optimal solution node of the CSSP, and let $g(n_*)$ be the corresponding (smallest) error value. Suppose the function $v(n_i) \geq 0$ can be computed at each node n_i. Define:

$$f'(n_i) = g(n_i) - h(n_i) + \epsilon v(n_i), \quad \epsilon \geq 0$$

Then if $f'(n_i)$ is used by the algorithm in Figure 1, it will terminate with a solution node n_{**} satisfying:

$$g(n_{**}) \leq g(n_*) + \epsilon v_{\max}, \quad v_{\max} = \max_{n_i} v(n_i) \qquad (4)$$

Proof: Suppose the theorem is false. Then there is a case where the algorithm terminates at the solution node n_{**} with the associated error $g(n_{**})$ which satisfies: $g(n_{**}) > g(n_*) + \epsilon v_{\max}$. In Lemma 1 below we prove that under this condition if n_z is any node on the path from the root to n_* then $f'(n_z) < f'(n_{**})$. We proceed to show that this leads to a contradiction.

To see the contradiction observe that the fringe list always contains a node n_z on the path from the root to n_*. (The root is such a node, and if such a node is selected from the fringe, one of its children will be on the path from the root to n_*.) But then since $f'(n_z) < f'(n_{**})$ any such node n_z in the fringe will be selected before n_{**}, so that eventually n_* is selected before n_{**} contradicting the assumption. ■

Lemma 1: With the notation of Theorem 1 under the assumption $g(n_{**}) > g(n_*) + \epsilon v_{\max}$, if n_z is on the path to n_* then $f'(n_z) < f'(n_{**})$.

Proof:

$$\begin{aligned} f'(n_{**}) &= g(n_{**}) - h(n_{**}) + \epsilon v(n_{**}) = g(n_{**}) + \epsilon v(n_{**}) \\ &> g(n_*) + \epsilon v_{\max} + \epsilon v(n_{**}) \\ &\geq f(n_*) + \epsilon v_{\max} \geq f(n_z) + \epsilon v(n_z) = f'(n_z) \end{aligned}$$

In the above inequality chain we used the following: 1. $h(n_{**})=0$ and $f(n_*) = g(n_*)$, since both n_{**} and n_* are leaf (goal) nodes. 2. $f(n_*) \geq f(n_z)$ since n_z is on the path to n_* and $f(n_i)$ is monotonically increasing along any path. 3. $v_{\max} \geq v(n_z)$. ■

Theorem 1 is a useful tool for proving bounds on the suboptimality of the solution. But by itself it doesn't provide guidance as to what constitutes a useful function $v(n_i)$. The following corollary suggests that $v(n_i)$ should be chosen monotonically decreasing along any path. Under this condition the bound becomes tighter during the run of the algorithm.

Corollary 1: Let n_{**}, n_*, and $v(n_i)$ be as in Theorem 1. Suppose $v(n_i)$ is monotonically decreasing (non-increasing) along any path. Let n_r be a visited node in the path to the solution. Then:

$$g(n_{**}) \leq g(n_*) + \epsilon v(n_r) \qquad (5)$$

Proof: Apply Theorem 1 to the subgraph rooted at n_r. ■

The guarantee in the corollary may be much stronger then the guarantee provided by the theorem since $v(n_r)$ may be much smaller than the value of $v(n_i)$ at the root.

The suboptimality bounds on $g(n_{**})$ that can be proved from the theorem and the corollary are additive. This is weaker than the multiplicative bound of the classic WA* algorithm (e.g., (Pearl 1984). Multiplicative bounds are typically preferred because they are more accurate for small $g(n_*)$ values. In particular, when $g(n_*) = 0$ algorithms with multiplicative bounds on their suboptimality return the optimal solution regardless of the value of ϵ.

Since both $g(n_i)$ and $h(n_i)$, as defined in Section 3, are monotonically decreasing, we have experimented with them as possible choices for $v(n_i)$. (As discussed in Section 3.3 these choices give essentially the same algorithm.) In addition, we propose a third option, the function $b(n_i)$, which gives a multiplicative bound on the algorithm suboptimality.

The algorithms are summarized in the following table where we write f'_i, g_i, h_i, for $f'(n_i), g(n_i), h(n_i)$, respectively, g_0, h_0, b_0 for the values of $g(), h(), b()$ at the root, and g_*, g_{**} for $g(n_*), g(n_{**})$ respectively.

Name	f'_i	Suboptimality Bound
CSSP-WA*-g	$g_i - h_i + \epsilon g_i$	$g_{**} \leq g_* + \epsilon g_0$
CSSP-WA*-h	$g_i - h_i + \epsilon h_i$	$g_{**} \leq g_* + \epsilon h_0$
CSSP-WA*-b	$g_i - h_i + \epsilon b_i$	$g_{**} \leq g_* + \epsilon b_0$ $g_{**} \leq (1+\epsilon(k+1)) g_*$

3.1 Computing the heuristics

In calculating $f(n_i), g(n_i), h(n_i)$ we follow (Arai, Maung, and Schweitzer 2015). We briefly describe the main formulas. Let S_i be the columns at node n_i, and set $j = |S_i|$. Compute Q_i, an orthogonal matrix which forms a basis to S_i. Compute: $X_i = X - Q_i Q_i^T X$, and $B_i = X_i X_i^T$. Let $\lambda_1 \geq \ldots \geq \lambda_m$ be the eigenvalues of B_i. Then:

$$g(n_i) = \sum_{t=1}^{m} \lambda_t = \mathsf{Trace}(B_i) = \|X_i\|_F^2,$$

$$h(n_i) = \sum_{t=1}^{j} \lambda_t, \quad f(n_i) = g(n_i) - h(n_i) = \sum_{t=j+1}^{m} \lambda_t$$

3.2 A multiplicative bound on suboptimality

We begin by sharpening the right hand size of Equation (2): $\mathrm{Err}(S_k^*) \leq (k+1)\mathrm{Err}_k^*$. With the notation used in this section this can be written as: $g(n_*) \leq (k+1) f(n_0)$, where n_0 is the root. Consider a node n_i corresponding to a subset S_i of j columns, where k–j columns still need to be selected. Let S_i^* be the best solution satisfying $S_i^* \supset S_i$. We have:

$$g(n_*) \leq \mathrm{Err}(S_i^*) \leq (k\text{–}j\text{+}1) f(n_i) = (k\text{–}j\text{+}1)\Big(\sum_{t=k-j+1}^{m} \lambda_t\Big)$$

where the eigenvalue calculations are as defined in Section 3.1. Now consider adding a zero vector to S_i. This does not change the eigenvalues, but increases the value of j. Therefore, $(k-j)(\sum_{t=k-j}^{m} \lambda_t)$ is also an upper bound. Since this process can be repeated by adding l zero vectors,

for $l = 0, \ldots, k-j$, we get the following sharper upper bound:

$$g(n_*) \leq \text{Err}(S_i^*) \leq \min_{l=0,\ldots,k-j} (k\text{–}j\text{+}1\text{–}l)(\sum_{t=k-j+1-l}^{m} \lambda_t)$$

We define the right hand side as $b(n_i)$:

$$b(n_i) = \min_{l=0,\ldots,k-j} (k\text{–}j\text{+}1\text{–}l)(\sum_{t=k-j+1-l}^{m} \lambda_t)$$

As an upper bound $b(n_i)$ is expected to decrease (but not necessarily monotonically). For n_0 we have:

$$b(n_0) = \min_{l=0,\ldots,k} (k\text{+}1\text{–}l)(\sum_{t=k+1-l}^{m} \lambda_t)$$
$$\leq (k+1)(\sum_{t=k+1}^{m} \lambda_t) = (k+1)g(n_*)$$

This shows that the additive upper bound $g_{**} \leq g_* + \epsilon b_0$ implies the multiplicative upper bound $g_{**} \leq (1+\epsilon(k+1))g_*$.

3.3 The relationship between the three algorithms

All three algorithms reduce to the CSSP-A* for $\epsilon = 0$. As shown above, the CSSP-WA*-b has a multiplicative bound on its suboptimality while the other two have only additive bounds. This implies that when g_* is very small, the CSSP-WA*-b is more accurate than the other two. Our experiments support this observation.

We proceed to show that the CSSP-WA*-g and the CSSP-WA*-h are essentially the same algorithm corresponding to different choices of ϵ. The f' values of these algorithms can be expressed as:

$$f'_g(n_i) = (1+\epsilon_g)g(n_i) - h(n_i)$$
$$f'_h(n_i) = g(n_i) - (1-\epsilon_h)h(n_i)$$

Since an arbitrary scaling of the f' values does not affect the behavior of the algorithm we have the following equivalent form of $f'_g(n_i)$:

$$f''_g(n_i) = \frac{f'_g(n_i)}{1+\epsilon_g} = g(n_i) - \frac{h(n_i)}{1+\epsilon_g}$$

Comparing this with the expression for $f'_h(n_i)$ shows that the two algorithms are equivalent under the condition:

$$\epsilon_h = \frac{\epsilon_g}{1+\epsilon_g}$$

This shows the two algorithms to be equivalent whenever $\epsilon_h < 1$.

4 Experimental results

We implemented and tested the three CSSP-WA* algorithms that were described in Section 3. The experiments described here compare our algorithms to the current state-of-the-art on several large publicly available datasets.

The state-of-the-art algorithms considered here (see Section 2) are the GKS (see (Golub and Van-Loan 1996)) and the GE ((Gu and Eisenstat 1996)) from numerical linear algebra, and the randomized LEV-GE two-stage-method of Boutsidis et.al (Boutsidis, Mahoney, and Drineas 2009). The datasets are shown in the following table:

Name	Size	availability
Madelon	$2,000 \times 500$	UCI
Isolet5	$1,559 \times 618$	UCI
CNAE-9	$1,080 \times 857$	UCI
TechTC01	$163 \times 29,261$	Technion
Day1	$20,000 \times 3,231,957$	UCI

4.1 Evaluating accuracy and running time

The results of accuracy evaluation are shown in the table of Figure 2. Observe that the WA* algorithms always come as most accurate. On the other hand, the WA* algorithms are significantly slower than the other algorithms.

4.2 Dependency on ϵ

The results of experiments for evaluating the performance of the WA* algorithms as a function of ϵ are shown in Figure 3. The experiments were performed on the TechTC01 dataset with $k = 50$. The upper left panel shows runtime as a function of ϵ. Although the results for the WA* are clearly worse than the results of the other algorithms, they are still measured in minutes. As a comparison, running the CSSP-A* on the same dataset is impractical, as the results may take hundreds of years. The bottom left panel shows the results only for the WA* algorithms. It clearly shows that the runtime improves with the increase of ϵ value but only up to a certain point. Increasing ϵ above 0.4 gives little or no improvement in running time.

The upper right panel shows accuracy as a function of ϵ. Observe that the WA* algorithms are significantly more accurate than the other algorithms for all values of ϵ in the $0-1$ range that we experimented with. The same comparison showing only the WA* algorithms is shown in the lower right panel. In terms of accuracy the results are less clear, although in general the results seem to improve for smaller ϵ values.

5 Concluding remarks

This paper describes new algorithms for the column subset selection problems that are based on the classic Weighted A* approach in heuristic search. Our algorithms run slower than the current state-of-the-art, but are still practical even for large datasets. Their advantage is having better accuracy than the current state-of-the-art. Their accuracy and running time can be fine tuned with the ϵ parameter. For $\epsilon = 0$ they all find the optimal solution, but this will most likely take exponential time. Our claim of practicality is for larger ϵ values where our experiments indicate superior accuracy at an "acceptable" running time. We are not aware of any other algorithm in the literature that is practical, and can have similar accuracy levels.

Since the algorithms presented here are still much slower than some of the algebraic/randomized techniques, it is questionable whether they should be an appropriate tool for big data. (See, for example, the runtime results on the Day1

k	GKS/time	GE/time	LEV_GE/time	WA*-h/time	WA*-g/time	WA*-b/time
			Madelon			
20	6.97E+08/0.01	7.00E+08/0.003	6.52E+08/0.30	**6.05E+08**/0.58	**6.05E+08**/0.59	**6.05E+08**/0.58
60	4.82E+08/0.01	4.86E+08/0.003	5.27E+08/0.87	**4.59E+08**/2.11	**4.59E+08**/2.39	**4.59E+08**/2.36
100	3.48E+08/0.02	3.47E+08/0.006	4.21E+08/1.52	**3.38E+08**/4.47	**3.38E+08**/4.92	**3.38E+08**/4.86
			Isolet5			
20	7.72E+04/0.01	9.30E+04/0.002	8.01E+04/0.45	**6.58E+04**/0.97	**6.58E+04**/0.92	6.60E+04/0.80
60	4.01E+04/0.01	4.57E+04/0.008	4.10E+04/1.78	3.72E+04/3.63	3.72E+04/3.50	**3.70E+04**/6.77
100	2.62E+04/0.02	2.88E+04/0.016	2.74E+04/3.94	2.48E+04/6.90	2.48E+04/6.75	**2.47E+04**/6.78
			CNAE-9			
20	4.28E+03/0.01	4.31E+03/0.002	4.97E+03/0.29	**4.27E+03**/1.82	4.28E+03/1.84	4.28E+03/1.83
60	2.58E+03/0.01	2.56E+03/0.002	3.38E+03/0.89	**2.55E+03**/6.15	**2.55E+03**/6.27	**2.55E+03**/6.29
100	1.77E+03/0.02	1.79E+03/0.003	2.43E+03/1.55	**1.76E+03**/10.80	**1.76E+03**/10.96	**1.76E+03**/10.97
			TechTC01			
20	7.14E+05/0.02	7.68E+05/0.006	8.68E+05/0.67	6.93E+05/0.75	**6.92E+05**/0.79	**6.92E+05**/0.80
60	1.52E+05/0.04	1.59E+05/0.011	1.65E+05/1.93	1.29E+05/5.65	**1.28E+05**/5.68	**1.28E+05**/5.68
100	2.78E+04/0.06	3.18E+04/0.350	2.80E+04/3.15	2.12E+04/14.50	**2.07E+04**/14.54	**2.07E+04**/14.47
			Day1			
20	4.82E+05/0.50	-	5.01E+05/10.67	4.60E+05/31.25	**4.59E+05**/31.24	**4.59E+05**/31.40
60	3.72E+05/1.50	-	3.97E+05/35.18	3.50E+05/140.68	**3.49E+05**/140.57	**3.49E+05**/140.59
100	3.20E+05/2.84	-	3.48E+05/63.01	3.02E+05/272.58	**3.01E+05**/297.58	-

Figure 2: Error and time comparison. The most accurate result for each experiment is underlined. Some results are missing because they are too slow. The WA* algorithms use $\epsilon = 0.5$ on Madelon, CNAE-9 and TechTC01, and $\epsilon = 0.9$ on the other datasets. Time values are in minutes.

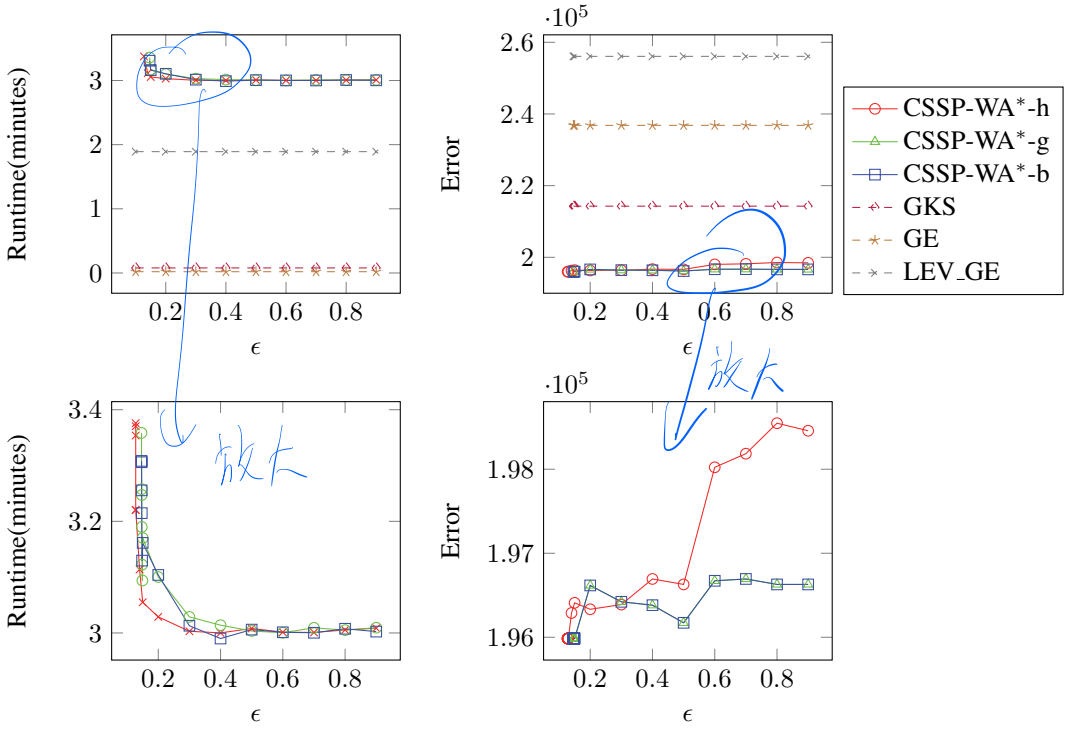


Figure 3: Performance comparison as a function of ϵ on TechTC01, for k=50.

dataset.) But for most other applications they give practical solutions with superior accuracy to currently available methods.

6 Acknowledgments

The authors would like to thank an anonymous referee for pointing out the equivalence between the CSSP-WA*-g and the CSSP-WA*-h, as discussed in Section 3.3. The first author would like to thank the Japan Air Self-Defence Force for supporting his studies.

References

Arai, H.; Maung, C.; and Schweitzer, H. 2015. Optimal column subset selection by A-Star search. In *Proceedings of the 29th National Conference on Artificial Intelligence (AAAI'15)*, 1079–1085. AAAI Press.

Boutsidis, C.; Mahoney, M. W.; and Drineas, P. 2009. An improved approximation algorithm for the column subset selection problem. In Mathieu, C., ed., *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, 968–977. SIAM.

Çivril, A., and Magdon-Ismail, M. 2009. On selecting a maximum volume sub-matrix of a matrix and related problems. *Theoretical Computer Science* 410(47-49):4801–4811.

Çivril, A., and Magdon-Ismail, M. 2012. Column subset selection via sparse approximation of SVD. *Theoretical Computer Science* 421:1–14.

Çivril, A. 2014. Column subset selection problem is ughard. *Journal of Computer and System Sciences* 80(4):849–859.

Dasgupta, A.; Drineas, P.; Harb, B.; Josifovski, V.; and Mahoney, M. W. 2007. Feature selection methods for text classification. In Berkhin, P.; Caruana, R.; and Wu, X., eds., *KDD*, 230–239. ACM.

Deshpande, A., and Rademacher, L. 2010. Efficient volume sampling for row/column subset selection. In *FOCS*, 329–338. IEEE Computer Society Press.

Deshpande, A.; Rademacher, L.; Vempala, S.; and Wang, G. 2006. Matrix approximation and projective clustering via volume sampling. *Theory of Computing* 2(12):225–247.

Drineas, P.; Lewis, J.; and Paschou, P. 2010. Inferring geographic coordinates of origin for europeans using small panels of ancestry informative markers. *PLoS ONE* 5(8):e11892. doi:10.1371/journal.pone.0011892.

Duda, R. O.; Hart, P. E.; and Stork, D. G. 2001. *Pattern Classification*. John Wiley & Sons, second edition.

Dy, J. G.; Brodley, C. E.; Kak, A.; Broderick, L. S.; and Aisen, A. M. 2003. Unsupervised feature selection applied to content-based retrieval of lung images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25(3):373–378.

Frieze, A. M.; Kannan, R.; and Vempala, S. 2004. Fast Monte-Carlo algorithms for finding low-rank approximations. *Journal of the ACM* 51(6):1025–1041.

Golub, G. H., and Van-Loan, C. F. 1996. *Matrix computations*. The Johns Hopkins University Press, third edition.

Gu, M., and Eisenstat, S. C. 1996. Efficient algorithms for computing a strong rank-revealing QR factorization. *SIAM J. Computing* 17(4):848–869.

Guruswami, V., and Sinop, A. K. 2012. Optimal column-based low-rank matrix reconstruction. In Rabani, Y., ed., *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, 1207–1214. SIAM.

Guyon, I., and Elisseeff, A. 2003. An introduction to variable and feature selection. *Journal of Machine Learning Research* 3:1157–1182.

Hart, P. E.; Nilsson, N.; and Raphael, B. 1968. A formal basis for the heurisitc determination of minimal cost paths. *IEEE Transactions on Systems Science and Cybernetics* 4(2):100–107.

Jimenez-Rodriguez, L. O.; Arzuaga-Cruz, E.; and Velez-Reyes, M. 2007. Unsupervised linear feature-extraction methods and their effects in the classification of high-dimensional data. *IEEE Transactions on Geoscience and Remote Sensing* 45(2):469–483.

Kumar, S.; Mohri, M.; and Talwalkar, A. 2012. Sampling methods for the nystrom method. *Journal of Machine Learning Research* 13:981–1006.

Li, Z.; Yang, Y.; Liu, J.; Zhou, X.; and Lu, H. 2012. Unsupervised feature selection using nonnegative spectral analysis. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*. AAAI Press.

Likhachev, M.; J.Gordon, G.; and Thrun, S. 2003. ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in Neural Information Processing Systems 16 (NIPS-03)*, 767–774.

Maung, C., and Schweitzer, H. 2013. Pass-efficient unsupervised feature selection. In *Advances in Neural Information Processing Systems (NIPS)*, volume 26, 1628–1636.

Maung, C., and Schweitzer, H. 2015. Improved greedy algorithms for sparse approximation of a matrix in terms of another matrix. *IEEE Transactions on Knowledge and Data Engineering* 27(3):769–780.

Pearl, J. 1984. *Heuristics : intelligent search strategies for computer*. Reading, Massachusetts: Addison-Wesley.

Russell, S., and Norvig, P. 2010. *Artificial Intelligence - A Modern Approach*. Pearson Education.

Tropp, J. A. 2004. Greed is good: algorithmic results for sparse approximation. *IEEE Transactions on Information Theory* 50(10):2231–2242.

Wang, C.; Gong, M.; Zhang, M.; and Chan, Y. 2015. Unsupervised hyperspectral image band selection via column subset selection. *IEEE Geoscience and Remote Sensing Letters* 12(7):1411–1415.

Wei, H., and Billings, S. A. 2007. Feature subset selection and ranking for data dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29(1):162–166.