Scalable Unsupervised Feature Selection with Reconstruction Error Guarantees via QMR Decomposition

Anonymous Author(s)

ABSTRACT

Unsupervised feature selection (UFS) methods have garnered significant attention for their capability to eliminate redundant features without relying on class label information. However, their scalability to large datasets remains a challenge, rendering common UFS methods impractical for such applications. To address this issue, we introduce QMR-FS, a greedy forward filtering approach that selects linearly independent features up to a specified relative tolerance, ensuring that any excluded features can be reconstructed from the retained set within this tolerance. This is achieved through the QMR matrix decomposition, which builds upon the well-known QR decomposition. QMR-FS benefits from linear complexity relative to the number of instances and boasts exceptional performance due to its ability to leverage parallelized computation on both CPU and GPU. Despite its greedy nature, QMR-FS achieves comparable classification and clustering accuracies across multiple datasets when compared to other UFS methods, while achieving runtimes approximately 10 times faster than recently proposed scalable UFS methods for datasets ranging from 100 million to 1 billion elements.

CCS CONCEPTS

• **Computing methodologies** → **Feature selection**; **Unsupervised learning**.

KEYWORDS

Unsupervised learning, Feature selection, Scalability, Linear independence

ACM Reference Format:

Anonymous Author(s). 2018. Scalable Unsupervised Feature Selection with Reconstruction Error Guarantees via QMR Decomposition. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Unsupervised feature selection (UFS) involves extracting a subset of features from a dataset by eliminating redundant ones without relying on task-specific information, such as class labels. The removal of redundant features offers several inherent benefits: it facilitates data visualization and understanding, reduces measurement and storage requirements, and decreases training and utilization times [10, 25]. Additionally, feature selection can enhance the performance of downstream tasks, such as clustering, by mitigating the curse of dimensionality [10, 13, 22].

UFS methods can be categorized into two main types: *wrapper* and *filter* methods [25]. Wrapper methods rely on an external task algorithm, such as a clustering algorithm, to evaluate feature importance, whereas filter methods use only the intrinsic properties of the features and dataset. Filter methods are considered faster and more scalable [10, 25, 26], making them the focus of this work.

Despite being labelled 'fast,' filtering UFS methods often become inefficient for large datasets [15] due to their quadratic time complexity with respect to the number of instances[11, 18, 31–34], meaning n in an $n \times d$ data matrix with d features. This quadratic complexity makes computation intractable for the datasets that would benefit most from feature selection in terms of reducing storage and utilization times. While some methods exhibit linear complexity in n [29, 30], they use iterative schemes that must run until convergence, which is slow in practice.

To address the lack of scalable UFS methods, we introduce QMR-FS, a fast feature selection algorithm. QMR-FS employs greedy forward feature selection, where features are considered one-by-one in a fixed order, and feature redundancy is measured only with respect to already selected features. Specifically, QMR-FS uses linear reconstruction error to measure redundancy, based on the premise that linearly reconstructable features are redundant [6]. Our QMR decomposition, an extension of the QR decomposition [12, Ch. 2.1], computes the reconstruction error efficiently by avoiding costly linear model fitting, ensuring linear complexity in n, and is optimized for parallel implementation on both CPU and GPU. Additionally, QMR-FS guarantees that reconstruction errors are smaller than a specified tolerance relative to the feature columns' L2-norm, allowing it to retain all features if redundancy criteria are not met. Thus, QMR-FS is suitable for scenarios where the potential for feature removal without information loss is uncertain.

We demonstrate the scalability of QMR-FS via runtimes around 30 seconds on datasets with up to 7.88 million instances and 1 billion elements, roughly 10 times faster than recent scalable UFS methods [15]. Furthermore, the features selected by QMR-FS exhibit comparable classification accuracy and clustering normalized mutual information (NMI) scores relative to both common and recent UFS methods, which require significantly longer runtimes. Code for QMR-FS and our experiments is available online[1].

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

[1] https://anonymous.4open.science/r/qmr-feature-selection-F3E8

2 METHOD

We now describe how our QMR-FS method performs scalable feature selection on a data matrix with n instances and d features, $\mathbf{X} \in \mathbb{R}^{n \times d}$. Since our focus is scalability to large n, we confine this paper to the case $n \geq d$ and relegate the opposite case to future work.

QMR-FS performs greedy forward feature selection, considering features from left to right in $\mathbf{X}$, with the decision to retain a feature based on the previously retained features. The guiding principle is that of linear independence. We say that a feature column $\mathbf{x}_j \in \mathbf{X}$ is *linearly dependent* on previous feature columns if it can be expressed as a linear combination of them, plus a constant term,

$$\mathbf{x}_j = \sum_{k=1}^{j-1} a_k \, \mathbf{x}_k + b\mathbf{1}_n, \tag{1}$$

where a_k and b are coefficients, and $\mathbf{1}_n$ is a length n vector of all ones. This is expressed concisely as $\mathbf{x}_j \in \text{span}(\mathbf{x}_1, \ldots, \mathbf{x}_{j-1}, \mathbf{1}_n)$. Conversely, if such a_k and b do not exist, $\mathbf{x}_j$ is *linearly independent*, $\mathbf{x}_j \notin \text{span}(\mathbf{x}_1, \ldots, \mathbf{x}_{j-1}, \mathbf{1}_n)$, and is to be retained.

To determine the independence of $\mathbf{x}_j$ without having to explicitly find a_k and b through expensive linear model fitting, we make use of the following lemma, central to our method.

Lemma 2.1. *Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\mathbf{R}_{ref} \in \mathbb{R}^{d \times d}$ fulfil $\mathbf{X} = \mathbf{U}\,\mathbf{R}_{ref}$ and $\mathbf{R}_{ref} = \mathbf{U}^\dagger\,\mathbf{X}$, where $\mathbf{R}_{ref}$ is in row echelon form (REF), and $\mathbf{U} \in \mathbb{R}^{n \times d}$ is an orthogonal basis with left-side inverse $\mathbf{U}^\dagger \in \mathbb{R}^{d \times n}$. Then $\mathbf{x}_j \in \text{span}(\mathbf{x}_1, \ldots, \mathbf{x}_{j-1})$ **iff** $\mathbf{r}_j \in \text{span}(\mathbf{r}_1, \ldots, \mathbf{r}_{j-1})$ for any $j > 1$ where $\mathbf{x}_j \in \mathbb{R}^n$ is the jth column in $\mathbf{X}$, and $\mathbf{r}_j \in \mathbb{R}^d$ is the jth column in $\mathbf{R}_{ref}$.*

Proof. We first show that $\mathbf{x}_j \in \text{span}(\mathbf{x}_1, \ldots, \mathbf{x}_{j-1}) \implies \mathbf{r}_j \in \text{span}(\mathbf{r}_1, \ldots, \mathbf{r}_{j-1})$. In this case, we can assume $\exists\, a_{i,j} \in \mathbb{R}$ s.t. $\mathbf{x}_j = \sum_{i=1}^{j-1} a_{i,j}\,\mathbf{x}_i$. From $\mathbf{X} = \mathbf{U}\,\mathbf{R}_{\text{ref}}$ know that $\mathbf{x}_i = \mathbf{U}\,\mathbf{r}_i$. Thus,

$$\mathbf{U}\,\mathbf{r}_j = \mathbf{x}_j = \sum_{i=1}^{j-1} a_{i,j}\,\mathbf{x}_i = \sum_{i=1}^{j-1} a_{i,j}\,\mathbf{U}\,\mathbf{r}_i = \mathbf{U} \sum_{i=1}^{j-1} a_{i,j}\,\mathbf{r}_i\,.$$

Multiplication with $\mathbf{U}^\dagger$ completes the first part of the proof,

$$\mathbf{r}_j = \mathbf{U}^\dagger\,\mathbf{U} \sum_{i=1}^{j-1} a_{i,j}\,\mathbf{r}_i = \sum_{i=1}^{j-1} a_{i,j}\,\mathbf{r}_i\,.$$

Next, $\mathbf{r}_j \in \text{span}(\mathbf{r}_1, \ldots, \mathbf{r}_{j-1}) \implies \mathbf{x}_j \in \text{span}(\mathbf{x}_1, \ldots, \mathbf{x}_{j-1})$ is proved with a block matrix representation of $\mathbf{X} = \mathbf{U}\,\mathbf{R}_{\text{ref}}$ as aid,

$$\begin{pmatrix} | & & | & | & | \\ \mathbf{x}_1 & \cdots & \mathbf{x}_{j-1} & \mathbf{x}_j & \cdots & \mathbf{x}_d \\ | & & | & | & | \end{pmatrix} = \begin{pmatrix} | & & | & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_h & \cdots & \mathbf{u}_d \\ | & & | & | \end{pmatrix} \begin{pmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,j-1} & r_{1,j} & \cdots & r_{1,d} \\ & r_{2,2} & \cdots & \vdots & \vdots & \vdots & \vdots \\ & & & r_{h,j-1} & r_{h,j} & \cdots & \vdots \\ & & & & & & \vdots \end{pmatrix}. \tag{2}$$

Here, h is the rank of the first $j-1$ columns in $\mathbf{X}$, which coincide with the number of rows with pivot elements before column j in $\mathbf{R}_{\text{ref}}$. Since $\mathbf{R}_{\text{ref}}$ is in REF, $h \leq j-1$, and by the assumption $\mathbf{r}_j \in \text{span}(\mathbf{r}_1, ..., \mathbf{r}_{j-1})$ we know that $\mathbf{r}_j$ cannot have pivot elements.

We know that $\mathbf{x}_j = \sum_{i=1}^{h} \mathbf{u}_i\, r_{i,j}$ (orange and brown in Eq. (2)), and we wish to express $\mathbf{u}_i$ using columns in $\mathbf{X}$. To do so, we look at the block equation highlighted in blue and brown blocks in Eq. (2),

$$\mathbf{X}_{[:,1:j-1]} = \mathbf{U}_{[:,1:h]}\,\mathbf{R}_{\text{ref}[1:h,1:j-1]}\,. \tag{3}$$

If $h > 0$, the block matrix $\mathbf{R}_{\text{ref}[1:h,1:j-1]} \in \mathbb{R}^{h \times j-1}$ has pivot elements in every row by construction (REF), meaning that it has full row rank. Therefore, it has the right-side inverse $\mathbf{P} \in \mathbb{R}^{j-1 \times h}$ such that $\mathbf{R}_{\text{ref}[1:h,1:j-1]}\,\mathbf{P} = \mathbf{I}_{h \times h}$ [23, Ch. 2.1]. Consequently, we can rewrite Eq. (3) by multiplying with $\mathbf{P}$ from the right:

$$\mathbf{X}_{[:,1:j-1]}\,\mathbf{P} = \mathbf{U}_{[:,1:h]}\,\mathbf{R}_{\text{ref}[1:h,1:j-1]}\,\mathbf{P} \implies \mathbf{U}_{[:,1:h]} = \mathbf{X}_{[:,1:j-1]}\,\mathbf{P}.$$

Thus, the first h columns in $\mathbf{U}$ can be expressed using the first $j-1$ columns in $\mathbf{X}$ as $\mathbf{u}_i = \sum_{k=1}^{j-1} \mathbf{x}_k\, P_{k.i}$, for $i \in \{1, \ldots, h\}$. Inserting this into $\mathbf{x}_j = \sum_{i=1}^{h} \mathbf{u}_i\, r_{i,j}$ gives

$$\mathbf{x}_j = \sum_{i=1}^{h} \sum_{k=1}^{j-1} \mathbf{x}_k\, P_{k,i} r_{i,j} = \sum_{k=1}^{j-1} \mathbf{x}_k \sum_{i=1}^{h} P_{k,i} r_{i,j} = \sum_{k=1}^{j-1} c_{k,j}\,\mathbf{x}_k, \tag{4}$$

where $c_{k,j} = \sum_{i=1}^{h} P_{k,i} r_{i,j}$. Thus, for $h > 0$ we have shown that $\mathbf{r}_j \in \text{span}(\mathbf{r}_1, \ldots, \mathbf{r}_{j-1}) \implies \mathbf{x}_j \in \text{span}(\mathbf{x}_1, \ldots, \mathbf{x}_{j-1})$.

The case $h = 0$ corresponds to a trivial case as it requires all columns of $\mathbf{R}_{\text{ref}}$ with index smaller than j to consist of zeros. Consequently, all columns vectors of $\mathbf{X}$ up to and including j must also consist of zeros, meaning that $\mathbf{x}_j \in \text{span}(\mathbf{x}_1, \ldots, \mathbf{x}_{j-1})$ trivially. □

By this lemma, the linearly independent columns in $\mathbf{X}$ correspond to the linearly independent columns in $\mathbf{R}_{\text{ref}}$, which are easy to determine by finding its pivot elements. However, the lemma requires a matrix decomposition of the form $\mathbf{X} = \mathbf{U}\,\mathbf{R}_{\text{ref}}$, where $\mathbf{U}$ has a left-side inverse and $\mathbf{R}_{\text{ref}}$ is in REF. The well-known QR decomposition $\mathbf{X} = \mathbf{Q}\,\mathbf{R}$ is a good starting point as the matrix $\mathbf{Q} \in \mathbb{R}^{n \times d}$ has orthonormal columns, so $\mathbf{Q}^\intercal$ is its left-side inverse [12, Ch. 2.1]. Yet $\mathbf{R} \in \mathbb{R}^{d \times d}$ is only upper-triangular, not in REF.

Therefore, we put forward the QMR decomposition. The matrix $\mathbf{R}$ is further decomposed as $\mathbf{R} = \mathbf{M}\,\mathbf{R}_{\text{ref}}$ using Gaussian elimination, such that $\mathbf{R}_{\text{ref}} \in \mathbb{R}^{d \times d}$ is in REF, and $\mathbf{M} \in \mathbb{R}^{d \times d}$ is an invertible matrix composed of inverted row reduction operations [9, Ch. 3.2]. That is, $\mathbf{M} = \mathbf{M}_K^{-1} \ldots \mathbf{M}_2^{-1}\,\mathbf{M}_1^{-1}$, where each $\mathbf{M}_k$ is one of three invertible operations: row swap, addition of two rows, and multiplication of a row with a nonzero value [9, Ch. 3.2.6]. Thus, we have $\mathbf{X} = \mathbf{Q}\,\mathbf{M}\,\mathbf{R}_{\text{ref}}$, with $\mathbf{U} = \mathbf{Q}\,\mathbf{M}$ having the left-side inverse $\mathbf{U}^\dagger = \mathbf{M}^{-1}\,\mathbf{Q}^\intercal$, fulfilling the criteria of Lemma 2.1. Computation of $\mathbf{M}$ and $\mathbf{R}_{\text{ref}}$ is implemented using outer product Gaussian elimination [9, Ch. 3.2.8], see lines 12 to 15 in Alg.1.

In practice, exact linear independence can be too strict, resulting in more features being retained than desired. Therefore, QMR-FS removes features with small reconstruction errors. Let $\mathbf{x}_j = \boldsymbol{\chi}_j + \boldsymbol{\delta}_j$, where $\boldsymbol{\chi}_j = \sum_{k \in \bar{\mathbf{S}}} a_k\,\mathbf{x}_k$ is the reconstruction of $\mathbf{x}_j$ using the retained features in $\bar{\mathbf{S}}$, and $\boldsymbol{\delta}_j$ is the residual error vector orthogonal to $\boldsymbol{\chi}_j$. Then $\mathbf{x}_j$ is removed if $\|\boldsymbol{\delta}_j|_2 \leq \theta |\,\mathbf{x}_j\,\|_2$, where $\theta \in [0, 1]$ is the relative tolerance. Importantly, $\boldsymbol{\delta}_j$ can be computed without expensive least-squares fitting to calculate the coefficients a_k. Instead, a formula for $\boldsymbol{\delta}_j$ can be derived from the QMR decomposition $\mathbf{x}_j = \mathbf{U}\,\mathbf{r}_j$ by leveraging that $\mathbf{U} = \mathbf{Q}\,\mathbf{M}$ forms an orthogonal basis, and that $\mathbf{r}_j$ consists of three segments: elements constructing $\boldsymbol{\chi}_j$, pivot elements corresponding to $\boldsymbol{\delta}_j$, and zeros.

The derivation of the formula for $\boldsymbol{\delta}_j$ is best understood in the context of the pseudocode on lines 4 to 16 in Alg. 1. Consider the start of iteration j. Let $\mathbf{M}' = \mathbf{M}_{[:,1:p-1]}$ be the first $p-1$ columns of the current $\mathbf{M}$, and $\mathbf{R}' = \mathbf{R}_{[1:p-1,\bar{\mathbf{S}}]} \in \mathbb{R}^{(p-1) \times (p-1)}$ be the first $p-1$ rows of the columns in $\mathbf{R}$ with indices in $\bar{\mathbf{S}}$, where $|\,\bar{\mathbf{S}}\,| = p-1$. Then we know the following: 1) $\mathbf{R}'$ is in REF with full row rank, 2) $\mathbf{X}_{[:,\bar{\mathbf{S}}]} = \mathbf{Q}\,\mathbf{M}'\,\mathbf{R}'$, and 3) $\mathbf{Q}\,\mathbf{M}' = \mathbf{X}_{[:,\bar{\mathbf{S}}]}\,\boldsymbol{P}$, where $\boldsymbol{P}$ is the right-side inverse of $\mathbf{R}'$.

Statement 1) follows by construction of Alg. 1. Statement 2) is true for $j = 0$, and this relation is preserved for each iteration as $\mathbf{M}$, $\mathbf{R}$, p and $\bar{\mathrm{S}}$ get updated. To see this, consider the two cases of the if-statement on line 9. If the condition evaluates to `False`, Gaussian elimination proceeds normally, preserving Statement 2) by construction. If the condition evaluates to `True`, the elements $\mathbf{R}_{[p:j,j]}$ are set to zero, ensuring that column j has no pivot elements in $\mathbf{R}_{\text{ref}}$. Importantly, the elements in $\mathbf{R}_{[p:j,j]}$ are not used in subsequent iterations for updating $\mathbf{M}$, and since j was not added to $\bar{\mathrm{S}}$ in this case, the equality $\mathbf{X}_{[:,\bar{\mathrm{S}}]} = \mathbf{Q}\,\mathbf{M}'\,\mathbf{R}'$ remains unchanged. Thus, $\mathbf{X}_{[:,\bar{\mathrm{S}}]} = \mathbf{Q}\,\mathbf{M}'\,\mathbf{R}'$ holds for every iteration. Statement 3) follows from 2), since 1) ensures the existence of $\boldsymbol{P}$ [23, Ch. 2.1], using the same reasoning as in the transition from Eq. (3) to Eq. (4) in the proof above.

Given 1), 2) and 3), we can express $\mathbf{x}_j$ as $\boldsymbol{\chi}_j + \boldsymbol{\delta}_j$,

$$\begin{aligned}\mathbf{x}_j &= \mathbf{Q}\,\mathbf{M}_{[:,1:j]}\,\mathbf{R}_{[1:j,j]} = \mathbf{Q}\left(\mathbf{M}'\,\mathbf{R}_{[1:p-1,j]} + \mathbf{M}_{[:,p:j]}\,\mathbf{R}_{[p:j,j]}\right)\\ &= \mathbf{X}_{[:,\bar{\mathrm{S}}]}\,\boldsymbol{P}\,\mathbf{R}_{[1:p-1,j]} + \mathbf{Q}\,\mathbf{M}_{[:,p:j]}\,\mathbf{R}_{[p:j,j]} = \boldsymbol{\chi}_j + \boldsymbol{\delta}_j, \qquad (5)\end{aligned}$$

which provides us with the formula for $\boldsymbol{\delta}_j$ used on line 8 in Alg. 1.

Three additional aspects of QMR-FS require attention. First, due to its greedy selection approach, QMR-FS is biased towards the initial feature ordering in $\mathbf{X}$, with features further to the left being more likely to be retained. Therefore, QMR-FS benefits from a good initial ordering of the features. If prior knowledge about feature importance is available, it should be used to determine the initial ordering. In other cases, we propose to order features by their Shannon entropy [5, Ch. 2.1] (line 2). This heuristic is motivated by the various entropy measures that have been previously suggested for UFS [7, 10, 28]. However, we acknowledge that improving the initial ordering is an important direction for future work on QMR-FS. Second, to account for the constant term in Eq.1, we prepend $\mathbf{1}_n$ to $\mathbf{X}$ (line 3 in Alg. 1). Finally, the time complexity of QMR-FS is $O(nd^2)$, dominated by line 8 and the QR decomposition [9, Ch. 5.2.2].

Algorithm 1: The QMR-FS algorithm with 1 based indexing. *Inputs:* Data matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ and tolerance threshold $\theta \in [0, 1]$.

```
1  def qmr_fs(X, θ):
2    X = sort_columns_by_entropy(X, 'descending')
3    X = [1_n, X]          # Prepend column with constant value.
4    Q, R = qr_decomposition(X)
5    M = I_{d×d}           # Initialize M as the identity matrix.
6    S̄ = list(), p = 1     # Initialize S̄ and p.
7    for j in range(1, R.shape[1]):
8      δ_j = Q M[:, p:j]R[p:j, j]  # Formula provided by Eq. (5).
9      if ‖δ_j‖_2 ≤ θ‖ x_j ‖_2:
10       R[p:j, j] = 0       # Ensure no privot elements.
11     else:
         # Gaussian elimination on column j from row p.
12       ρ = R[p+1:j, j]/R[p, j]          # [(j−p−1) × 1]
13       R[p+1:j, :] −= ρ ⊗ R[p, :]       # [(j−p−1) × d]
14       M[:, p] += M[:, p+1:j]·ρ         # [d × 1]
15       p += 1              # Increment pivot counter.
16       S̄.append(j)         # Add feature j to selected set.
17   S̄.remove(1)             # Remove constant value column.
18   return X[:, S̄]
```

3 EXPERIMENTS

We conduct two sets of experiments to demonstrate the effectiveness of QMR-FS. First, we showcase its scalability by applying it to four large datasets: US Census (1990) [21], GitHub MUSAE [24], SNAP patents [17], and KDDCUP (1999) [27]. The results are shown in Tbl. 1. Notably, even for KDDCUP, the largest dataset with over 1 billion elements, QMR-FS completes in just over 30 seconds on a CPU (16 vCPUs @ 2.2 GHz) and 20 seconds on a GPU (Nvidia L4), which is 10 times faster compared to recent scalable UFS methods [14, 15] with reported runtimes of 200 s and 300 s on a dataset with about half as many elements, n = 630K and d = 900 [15].

In our second experiment, we demonstrate that QMR-FS performs on par with other UFS methods despite its greedy approach. Following the setup outlined in a recent review and benchmark paper [25], we evaluate the selected features using SVM classification [4] and K-means clustering [2]. For classification, we compute average accuracies using 5-fold cross-validation with five different seeds, and for clustering, we compute average normalized mutual information (NMI) [20, Ch. 16.3] using 25 seeds. We use five datasets from [25], all from the UCI ML repository [16], and add the popular Isolet dataset [3, 8, 19]. Dataset details are provided in Tbl. 2.

Following [25], we use SVD-entropy [28], LS [11], SPEC [34], USFSM [26], UDFS [31], and NDFS [18] as comparison UFS methods. To compensate for excluding methods in [25] without readily available implementations, we add the recent methods CNAFS [32] and FMIUFS [33], available in the Matlab UFS toolbox [1]. Each of these methods outputs a feature ranking and expects a specific proportion of features to be selected, which differs from QMR-FS, which automatically selects the number of features based on the tolerance threshold θ. To make results comparable, we extract eight different feature sets from each ranking produced by the baseline UFS methods, corresponding to 20%-90% of the features. For Isolet, we use a fixed number of features ranging from 20 to 100 instead of percentages, as is customary for this dataset [19, 29, 30]. For QMR-FS, we use multiple values of θ to obtain a spread over the number of preserved features.

Examining Fig. 1, we find that QMR-FS is among the top methods for classification, but no single method excels across all settings. The average rank results in Tbl. 2 convey a similar story. QMR-FS achieves the highest classification ranking and an average clustering ranking within one standard deviation of the best method. Overall, the results support our claim that QMR-FS performs comparably to other UFS methods while offering superior scalability. Finally, the Isolet results in Fig. 1j are notable for QMR-FS's significant improvement in accuracy when increasing from 20 to 100 features, suggesting potential for enhancing the initial feature ordering.

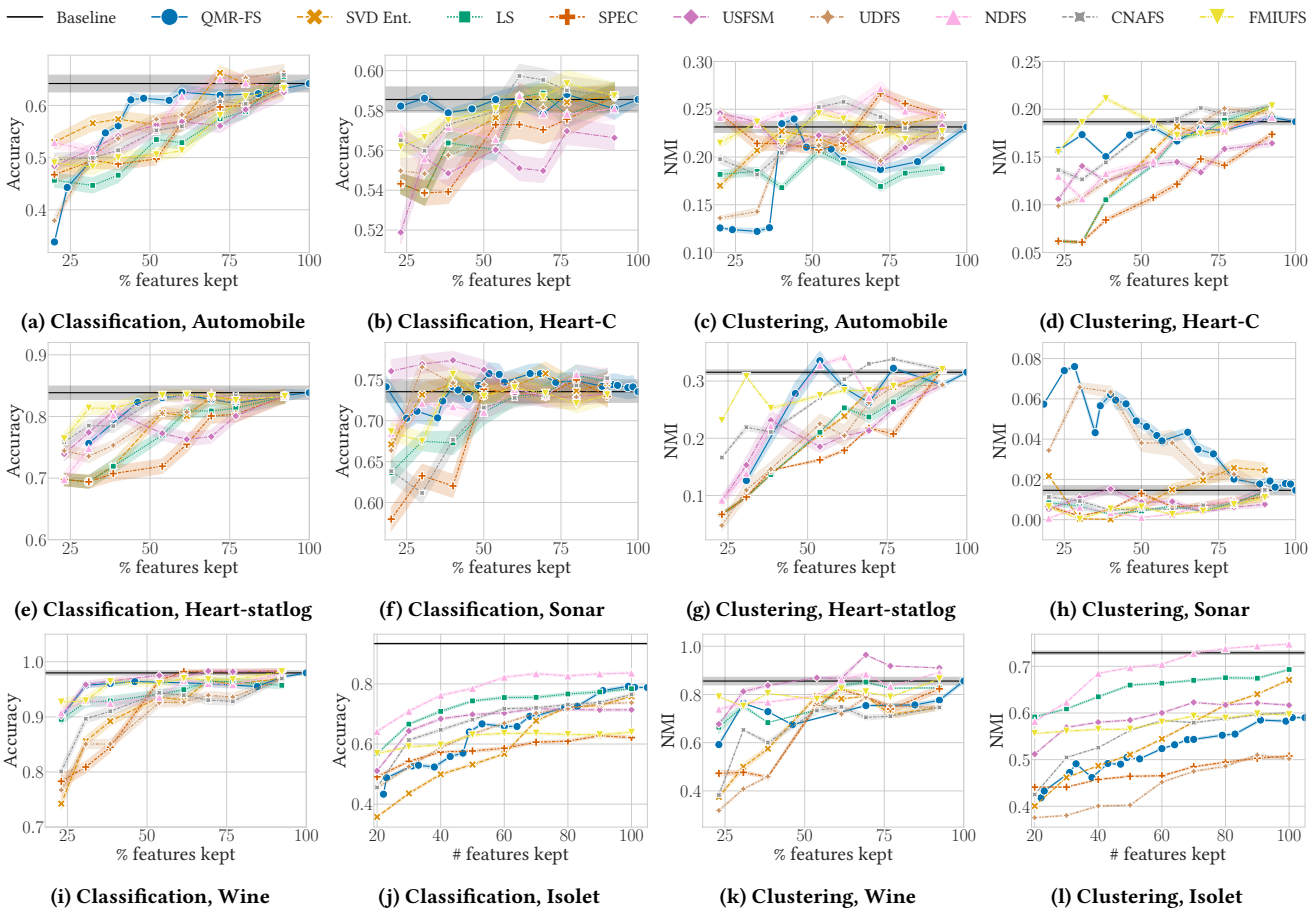
Table 1: QMR-FS runtimes in seconds on four large datasets using $\theta = 0.1$, resulting in d_{fs} selected features.

	Num. instances and dims.			Runtime (s)	
Dataset	n	d	d_{fs}	CPU	GPU
US Census (1990)	2.46M	68	66	4.16 ± 0.16	2.75 ± 0.11
GitHub MUSAE	37.7K	4006	3799	28.5 ± 0.18	13.9 ± 0.45
Snap Patents	2.92M	269	259	26.4 ± 0.25	13.7 ± 0.06
KDDCUP (1999)	7.88M	127	111	33.9 ± 0.12	20.2 ± 0.06

Table 2: Dataset details (left), and summary of benchmark results (right) using 40% and 60% kept features following [25] (#50 and #100 for Isolet). The average ranks and standard deviations for classification and clustering are computed over the 6 datasets. The time complexities are simplified under the assumption $n \geq d$, and $*$ indicates methods which are iterated until convergence. Relative runtimes are displayed for Isolet, the largest dataset the baseline UFS methods scale to.

Dataset	n	d	# classes
Automobile	205	25	6
Heart-C	303	13	5
Heart-statlog	270	13	2
Sonar	208	60	2
Wine	178	13	3
Isolet	1560	617	26

	QMR-FS	SVD Ent.	LS	SPEC	USFSM	UDFS	NDFS	CNAFS	FMIUFS
Clsif. avg. rank (40%)	2.8 ± 1.8	5.5 ± 2.8	5.8 ± 2.6	8.3 ± 0.5	3.0 ± 2.2	5.5 ± 1.8	4.3 ± 1.9	5.0 ± 1.4	3.7 ± 2.6
Clsif. avg. rank (60%)	2.5 ± 1.6	5.0 ± 1.8	5.7 ± 2.1	6.5 ± 2.6	5.8 ± 2.8	4.7 ± 2.8	3.2 ± 1.7	5.3 ± 3.1	5.2 ± 3.1
Clstr. avg. rank (40%)	3.0 ± 2.2	6.7 ± 2.1	6.5 ± 2.6	7.0 ± 1.7	3.7 ± 2.1	6.0 ± 2.8	3.3 ± 2.3	5.5 ± 1.6	2.8 ± 1.3
Clstr. avg. rank (60%)	5.5 ± 2.9	4.3 ± 2.0	4.8 ± 2.4	7.2 ± 1.3	4.8 ± 2.5	5.5 ± 2.6	3.0 ± 2.7	4.2 ± 3.4	4.8 ± 2.3
Time complexity	$O(nd^2)$	$O(nd^3)$	$O(n^2d)$	$O(n^2d)$	$O(n^3d)$	$O(n^2d)^*$	$O(n^2d)^*$	$O(n^2d)^*$	$O(n^2d)$
Runtime (Isolet)	137ms	×310	×1.2	×16	×22764	×11	×44	×2950	×13635



(a) Classification, Automobile (b) Classification, Heart-C (c) Clustering, Automobile (d) Clustering, Heart-C

(e) Classification, Heart-statlog (f) Classification, Sonar (g) Clustering, Heart-statlog (h) Clustering, Sonar

(i) Classification, Wine (j) Classification, Isolet (k) Clustering, Wine (l) Clustering, Isolet

Figure 1: Feature selection benchmark results. Each marker shows average SVM classification accuracy (left) or K-means NMI (right). The shaded areas show the standard error. The solid black lines show the score when 100% of features are used.

4 CONCLUSION AND FUTURE WORK

In this paper, we have presented QMR-FS, a scalable unsupervised feature selection (UFS) method, demonstrating its ability to achieve classification and clustering accuracies comparable to existing UFS methods while performing efficiently on large datasets. Despite these strengths, there are several avenues for further improvement. Enhancing the initial feature ordering, to which QMR-FS is biased, holds significant potential. This could be achieved by refining the feature ordering heuristic or by running QMR-FS with multiple random orderings and selecting the best order based on total reconstruction error. Given the speed of QMR-FS, this approach would not impose a substantial computational overhead.

Another research direction involves reducing the bias associated with feature ordering. For example, QMR-FS could be run iteratively, altering the feature order between iterations to reveal feature reconstruction possibilities not considered with a single ordering. Additionally, the differentiable nature of the QMR decomposition opens up exciting possibilities for applications in the deep learning domain, which we are eager to explore.

REFERENCES

[1] Farhad Abedinzadeh Torghabeh, Yeganeh Modaresnia, and Seyyed Abed Hosseini. 2023. Auto-UFSTool: An Automatic Unsupervised Feature Selection Toolbox for MATLAB. *Journal of AI and Data Mining* 11, 4 (2023), 517–524. https://doi.org/10.22044/jadm.2023.12820.2434

[2] David Arthur and Sergei Vassilvitskii. 2007. K-Means++: The Advantages of Careful Seeding. In *SODA '07*. ACM, 1027–1035. https://dl.acm.org/doi/10.5555/1283383.1283494

[3] Deng Cai, Xiaofei He, and Jiawei Han. 2011. Speed up kernel discriminant analysis. *The VLDB Journal* 20 (2011), 21–33. https://doi.org/10.1007/s00778-010-0189-3

[4] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20 (1995), 273–297. https://doi.org/10.1007/BF00994018

[5] Thomas M. Cover and Joy A. Thomas. 2005. *Entropy, Relative Entropy, and Mutual Information*. John Wiley & Sons, Ltd. 13–55 pages. https://doi.org/10.1002/047174882X.ch2

[6] S.K. Das. 1971. Feature Selection with a Linear Dependence Measure. *IEEE Trans. Comput.* C-20, 9 (1971), 1106–1109. https://doi.org/10.1109/T-C.1971.223412

[7] M. Dash, H. Liu, and J. Yao. 1997. Dimensionality reduction of unsupervised data. In *Proceedings Ninth IEEE International Conference on Tools with Artificial Intelligence*. IEEE, IEEE, 532–539. https://doi.org/10.1109/TAI.1997.632300

[8] Mark Fanty and Ronald Cole. 1990. Spoken Letter Recognition. In *Advances in Neural Information Processing Systems*, Vol. 3. Morgan-Kaufmann, 220–226. https://proceedings.neurips.cc/paper/1990.

[9] Gene H Golub and Charles F Van Loan. 2013. *Matrix computations* (fourth ed.). JHU press. https://doi.org/10.56021/9781421407944

[10] Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *Journal of machine learning research* 3, Mar (2003), 1157–1182. https://dl.acm.org/doi/10.5555/944919.944968

[11] Xiaofei He, Deng Cai, and Partha Niyogi. 2005. Laplacian score for feature selection. In *NIPS'05*. MIT Press, 507–514. https://dl.acm.org/doi/abs/10.5555/2976248.2976312

[12] Roger A. Horn and Charles R. Johnson. 2012. *Matrix Analysis* (2 ed.). Cambridge University Press. https://doi.org/10.1017/9781139020411

[13] Michael E. Houle, Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. 2010. Can Shared-Neighbor Distances Defeat the Curse of Dimensionality?. In *Scientific and Statistical Database Management*. Springer Berlin Heidelberg, 482–500. https://doi.org/10.1007/978-3-642-13818-8_34

[14] Haojie Hu, Rong Wang, Feiping Nie, Xiaojun Yang, and Weizhong Yu. 2018. Fast unsupervised feature selection with anchor graph and ℓ2,1-norm regularization. *Multimedia Tools Appl.* 77, 17 (sep 2018), 22099–22113. https://doi.org/10.1007/s11042-017-5582-0

[15] Haojie Hu, Rong Wang, Xiaojun Yang, and Feiping Nie. 2019. Scalable and Flexible Unsupervised Feature Selection. *Neural Computation* 31, 3 (2019), 517–537. https://doi.org/10.1162/neco_a_01163

[16] Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. 2024. The UCI Machine Learning Repository. https://archive.ics.uci.edu.

[17] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. SNAP Patents. Stanford Large Network Dataset Collection. https://snap.stanford.edu/data/cit-Patents.html.

[18] Zechao Li, Yi Yang, Jing Liu, Xiaofang Zhou, and Hanqing Lu. 2012. Unsupervised feature selection using nonnegative spectral analysis. In *AAAI'12*. AAAI Press, 1026–1032. https://doi.org/10.1609/aaai.v26i1.8289

[19] Yanfang Liu, Dongyi Ye, Wenbin Li, Huihui Wang, and Yang Gao. 2020. Robust neighborhood embedding for unsupervised feature selection. *Knowledge-Based Systems* 193 (2020), 105462. https://doi.org/10.1016/j.knosys.2019.105462

[20] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*. Cambridge University Press. https://nlp.stanford.edu/IR-book/information-retrieval-book.html

[21] Chris Meek, Bo Thiesson, and David Heckerman. 2001. US Census Data (1990). UCI Machine Learning Repository. https://doi.org/10.24432/C5VP42.

[22] Pabitra Mitra, Chivukula Anjaneya Murthy, and Sankar K. Pal. 2002. Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, 3 (2002), 301–312. https://doi.org/10.1109/34.990133

[23] C. Radhakrishna Rao. 1971. *Generalized inverse of matrices and its applications*. Wiley-Interscience. This book is not available online. Readers who cannot access a physical copy can also look in this online document for the same statement: `MIT course document`.

[24] UCI Machine Learning Repository. 2019. GitHub MUSAE. UCI Machine Learning Repository. https://doi.org/10.24432/C5Z02B.

[25] Saúl Solorio-Fernández, J Ariel Carrasco-Ochoa, and José Fco Martínez-Trinidad. 2020. A review of unsupervised feature selection methods. *Artificial Intelligence Review* 53, 2 (2020), 907–948. https://doi.org/10.1007/s10462-019-09682-y

[26] Saúl Solorio-Fernández, José Fco. Martínez-Trinidad, and J. Ariel Carrasco-Ochoa. 2017. A new Unsupervised Spectral Feature Selection Method for mixed data: A filter approach. *Pattern Recognition* 72 (2017), 314–326. https://doi.org/10.1016/j.patcog.2017.07.020

[27] Salvatore Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Philip Chan. 1999. KDD Cup 1999 Data. UCI Machine Learning Repository. https://doi.org/10.24432/C51C7N.

[28] Roy Varshavsky, Assaf Gottlieb, Michal Linial, and David Horn. 2006. Novel Unsupervised Feature Filtering of Biological Data. *Bioinformatics* 22, 14 (07 2006), e507–e513. https://doi.org/10.1093/bioinformatics/btl214

[29] Shiping Wang, Witold Pedrycz, Qingxin Zhu, and William Zhu. 2015. Subspace learning for unsupervised feature selection via matrix factorization. *Pattern Recognition* 48, 1 (2015), 10–19. https://doi.org/10.1016/j.patcog.2014.08.004

[30] Shiping Wang, Witold Pedrycz, Qingxin Zhu, and William Zhu. 2015. Unsupervised feature selection via maximum projection and minimum redundancy. *Knowledge-Based Systems* 75 (2015), 19–29. https://doi.org/10.1016/j.knosys.2014.11.008

[31] Yi Yang, Heng Tao Shen, Zhigang Ma, Zi Huang, and Xiaofang Zhou. 2011. ℓ2,1-norm regularized discriminative feature selection for unsupervised learning. In *IJCAI'11*. AAAI Press, 1589–1594. https://dl.acm.org/doi/10.5555/2283516.2283660

[32] Aihong Yuan, Mengbo You, Dongjian He, and Xuelong Li. 2022. Convex Non-Negative Matrix Factorization With Adaptive Graph for Unsupervised Feature Selection. *IEEE Transactions on Cybernetics* 52, 6 (2022), 5522–5534. https://doi.org/10.1109/TCYB.2020.3034462

[33] Zhong Yuan, Hongmei Chen, Pengfei Zhang, Jihong Wan, and Tianrui Li. 2022. A Novel Unsupervised Approach to Heterogeneous Feature Selection Based on Fuzzy Mutual Information. *IEEE Transactions on Fuzzy Systems* 30, 9 (2022), 3395–3409. https://doi.org/10.1109/TFUZZ.2021.3114734

[34] Zheng Zhao and Huan Liu. 2007. Spectral feature selection for supervised and unsupervised learning. In *ICML'07*. ACM, 1151–1157. https://doi.org/10.1145/1273496.1273641

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009