An automated approach for determining the number of components in non-negative matrix factorization with application to mutational signature learning

IOP Publishing Mach. Learn.: Sci. Technol. 2 (2021) 015013 https://doi.org/10.1088/2632-2153/abc60a

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
2 July 2020

REVISED
1 October 2020

ACCEPTED FOR PUBLICATION
29 October 2020

PUBLISHED
24 December 2020

PAPER

An automated approach for determining the number of components in non-negative matrix factorization with application to mutational signature learning

Gal Gilad[1,2], **Itay Sason**[1,2] **and Roded Sharan**[1]

[1] School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel
[2] Equal author contribution.

E-mail: roded@tauex.tau.ac.il

Keywords: non-negative matrix factorization, factorization methods, mutational signatures, NMF

Abstract

Non-negative matrix factorization (NMF) is a popular method for finding a low rank approximation of a matrix, thereby revealing the latent components behind it. In genomics, NMF is widely used to interpret mutation data and derive the underlying mutational processes and their activities. A key challenge in the use of NMF is determining the number of components, or rank of the factorization. Here we propose a novel method, CV2K, to choose this number automatically from data that is based on a detailed cross validation procedure combined with a parsimony consideration. We apply our method for mutational signature analysis and demonstrate its utility on both simulated and real data sets. In comparison to previous approaches, some of which involve human assessment, CV2K leads to improved predictions across a wide range of data sets.

1. Introduction

Non-negative matrix factorization (NMF) provides a low rank approximation of a matrix. When non-negativity constraints are applicable, NMF often provides a more meaningful structure than other low-rank approximation methods [1].

One important question concerning the use of NMF is how to determine the number of components, or rank, of the factorization (K). For example, Brunet *et al* [2] propose the cophenetic correlation coefficient as a measure of stability for each K, and choose the first value for which the cophenetic coefficient begins to fall. Hutchins *et al* [3] select the value of K in which the reconstruction error curve shows an inflection point. Another approach is a Bayesian variant of NMF [4], which chooses K that balances between model's likelihood and complexity. It uses an automatic relevance determination technique to remove irrelevant components that do not help to explain the input data. Other approaches are based on enumerating K and testing the performance of the different solutions in a cross validation setting (cf [5]). Owen and Perry's Bi-Cross-Validation [6] generalizes Gabriel's two-dimensional holdout cross validation scheme [7], leaving out a sub-matrix on each fold and using the rest of the matrix to reconstruct it. A different cross-validation scheme by Wold [8] that is also adopted in [9, 10] leaves out random entries from the initial matrix.

In the field of mutational signature learning, the two common methods are SigProfiler (SP) [11] and SignatureAnalyzer (SA) [12]. SigProfiler uses a method that is similar to Brunet *et al* It clusters signatures to K clusters and uses the tightness of the clusters as a measure of stability. The number of signatures is chosen after human assessment of the stability and accuracy of the solutions for a range of K values. SignatureAnalyzer uses the above Bayesian NMF to infer the number of signatures K, but may output different values across different runs and also depends on a regularization hyper-parameter.

Here we propose an automated method, CV2K, to choose K that is applicable for the most general definition of NMF. CV2K is based on a detailed comparison of the performance of different solutions in reconstructing hidden values of the original matrix in a cross validation setting, combined with a parsimony

Table 2.1. Examples for widely used NMF variants.

Description	$\phi(x)$	$\phi''(x)$	$d_\phi(x,y)$
Frobenius norm	$\frac{x^2}{2}$	1	$\frac{(x-y)^2}{2}$
Kullback–Leibler divergence (KL)	$x\log x$	$\frac{1}{x}$	$x\log\frac{x}{y}-x+y$
Itakura-Saito divergence (IS)	$-\log x$	$\frac{1}{x^2}$	$\frac{x}{y}-\log\frac{x}{y}$

consideration. We demonstrate its utility using both simulated and real data sets, showing that it outperforms previous approaches over a wide range of cases.

2. Methods

2.1. Non-negative matrix factorization

NMF receives as input a matrix $V \in \mathbb{R}_{\geq 0}^{N\times M}$ and a number of factors K, and outputs $W \in \mathbb{R}_{\geq 0}^{N\times K}, H \in \mathbb{R}_{\geq 0}^{K\times M}$ such that

$$V \approx WH$$

In its most general formulation, NMF solves the following minimization problem:

$$\underset{W\in\mathbb{R}_{\geq 0}^{N\times K},H\in\mathbb{R}_{\geq 0}^{K\times M}}{\arg\min} D(V|WH) := \sum_{i,j} d(V_{ij},(WH)_{ij}) \tag{1}$$

for some cost function $d : \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$. A useful class of distance functions can be obtained by using Bregman divergence, i.e for a univariate convex smooth function ϕ

$$d_\phi(x,y) = \phi(x) - \phi(y) - \phi'(y)(x-y)$$

is a cost function. Some useful ϕ functions are given in table 2.1.

In practice, sometimes a weight matrix $O \in \mathbb{R}_{\geq 0}^{N\times M}$ is given to us, e.g. reflecting entries with available data or the confidence in each cell, leading to the following extension:

$$\underset{W\in\mathbb{R}_{\geq 0}^{N\times K},H\in\mathbb{R}_{\geq 0}^{K\times M}}{\arg\min} D_{\phi,O}(V|WH) := \sum_{i,j} o_{ij} d_\phi(V_{ij},(WH)_{ij}) \tag{2}$$

In order to optimize NMF, which is known to be NP hard [13], we start from a random guess for W, H and apply some iterative update rules. For the initialization we use the random scheme used in scikit-learn; for the updates we use a scalar block coordinate descent (sBCD) [14], which we found to be the simplest and most efficient algorithm for a general cost function d_ϕ. We adapt both the initialization (lines 3–5) and sBCD (lines 6, 9–16) to work with a weight matrix as in (2). To check for convergence, we use a minimal relative improvement threshold ε and a maximum iteration number T (lines 7–8, 17) parameters. This is summarized in algorithm 1, which was adapted from [14]. In practice, to avoid bad local minimum, we initialize the model many times, train each for a few hundred iterations and continue training the best performing model.

2.2. Data imputation

In order to score different values of K, we hide some of the original matrix entries and test how well we can recover them. To this end, we set the weight matrix O to be binary, acting as a mask on the matrix values. To score the imputation of W, H we first normalize W, H, V, then average the L^1 norm over all missing values. The entire scheme is summarized in algorithm 2. In lines 2–4 we move the weights from the rows of H to the columns of W without changing the product WH. To reduce the variance caused by high/low values to impute, we normalize the rows of V and W to sum to one (lines 5–6, 7–8).

2.3. Choosing the number of components

Our CV2K scheme is described in algorithm 3 and consists of two parts:

(a) (lines 3–8) Evaluate all applicable values for K using imputation with a fraction p of missing values by repeatedly running Algorithm 2 on random O matrices (line 5). Note we require O to have at least one non-zero entry in every row and column. We let K^* be the value of K attaining the best median error over all runs. An example application of this criterion is shown in figure 1.

Algorithm 1 NMF(V, O, K)

Input: $V, O \in \mathbb{R}_{\geq 0}^{N \times M}, K \leq M$
Parameters: ϕ convex and univariate function, $\varepsilon > 0, T =$ max iterations
$W \sim \mathcal{N}(0,1)^{N \times K}, H \sim \mathcal{N}(0,1)^{K \times M}$
$d = \sqrt{\frac{\sum_{i,j} o_{ij} v_{ij}}{K \sum_{i,j} o_{ij}}}$
$W = d|W|, H = d|H|$
$E = V - WH$
$t = 0, S_{curr} = D_{\phi,O}(V|V'), S_{prev} = \infty$
While $t < T, \frac{S_{prev} - S_{curr}}{S_{curr}} > \varepsilon$ **do**
    $B = \phi''(WH) \circ O$
    for $k = 1, \ldots, K$ **do**
        $V^{(k)} = E + W_{:,k} H_{k,:} = (e_{nm} + w_{nk} h_{km})_{nm}$
        for $n = 1, \ldots, N$ **do**
            $w_{nk} = \max\{0, \frac{\sum_{m=1}^{M} b_{nm} v_{nm}^{(k)} h_{km}}{\sum_{m=1}^{M} b_{nm} h_{km} h_{km}}\}$
        for $m = 1, \ldots, M$ **do**
            $h_{km} = \max\{0, \frac{\sum_{n=1}^{N} b_{nm} v_{nm}^{(k)} w_{nk}}{\sum_{n=1}^{N} b_{nm} w_{nk} w_{nk}}\}$
        $E = V^{(k)} - W_{:,k} H_{k,:} = (v_{nm}^{(k)} - w_{nk} h_{km})_{nm}$
    $t = t + 1, S_{curr} = D_{\phi,O}(V|V'), S_{prev} = S_{curr}$
return W, H

Algorithm 2 Imputation Error (V, O, W, H)

Input: $V \in \mathbb{R}_{\geq 0}^{N \times M}, O \in \{0,1\}^{N \times M}, W \in \mathbb{R}_{\geq 0}^{N \times K}, H \in \mathbb{R}_{\geq 0}^{K \times M}$
$D_H = diag(\sum_{m=1}^{M} h_{1\,\mathrm{m}}, \ldots, \sum_{m=1}^{M} h_{Km})$
$H = D_H^{-1} H$
$W = W D_H$
$D_V = diag(\sum_{m=1}^{M} v_{1\,\mathrm{m}}, \ldots, \sum_{m=1}^{M} v_{Nm})$
$\tilde{V} = D_V^{-1} V$
$D_W = diag(\sum_{k=1}^{K} w_{1k}, \ldots, \sum_{k=1}^{K} w_{Nk})$
$\tilde{W} = D_W^{-1} W$
$\tilde{V}' = \tilde{W} H$
return $\frac{1}{\sum_{n,m}(1 - o_{nm})} \sum_{n,m} (1 - o_{nm}) |\tilde{v}_{nm} - \tilde{v}'_{nm}|$

Algorithm 3 CV2K $(V, K_{\min}, K_{\max})$

Input: $V \in \mathbb{R}_{\geq 0}^{N \times M}, 1 \leq K_{\min} < K_{\max} \leq \min\{N, M\}$
Parameters: $p =$ fraction to hide, $T =$ number of runs for each K
for $k = K_{\min}, \ldots, K_{\max}$ **do**
    for $t = 1, \ldots, T$ **do**
        $O \sim \text{Bernoulli}(p)^{N \times M}$
        $W, H = \text{NMF}(V, O, k)$
        $S[k, t] = \text{ImputationError}(V, O, W, H)$
$\tilde{K} = \arg\min_k(\text{median}(S[k, :]))$
do
    $K^* = \tilde{K}$
    $\tilde{K} = \min\{K < K^* | \text{Wilcoxons-rank-sum}(S[K, :], S[K^*, :]) > 0.05\}$
While $\tilde{K} < K^*$
return K^*

(b) (lines 9–12) Use Wilcoxon rank-sum test to compare the obtained value K^* to smaller values, and reduce the value if they are not significantly different. This parsimony consideration is used to resolve cases in which the imputation error reaches a plateau, as exemplified in figure 2. Table 2 summarizes the effect of this part on the performance of our method.

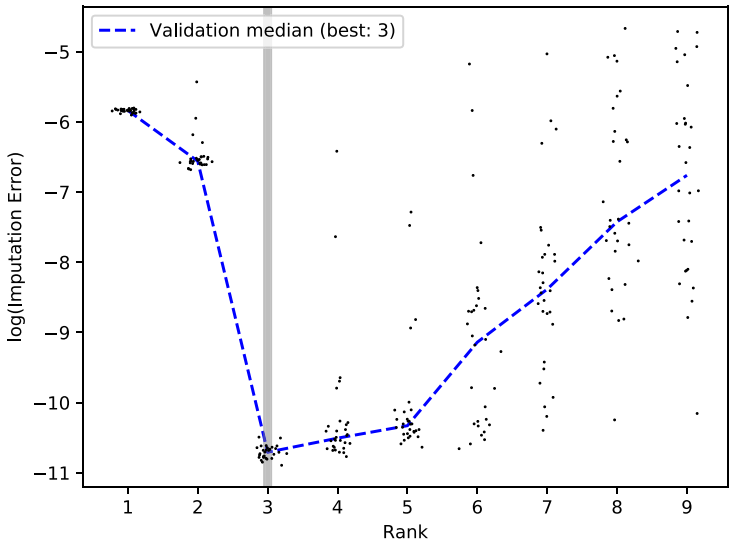

Figure 1. An example run of CV2K(0.1) on a simulated data set ((SP) vi). The minimum is obtained for $K = 3$ (as well as the final K, marked in gray). The true number of components is also 3.

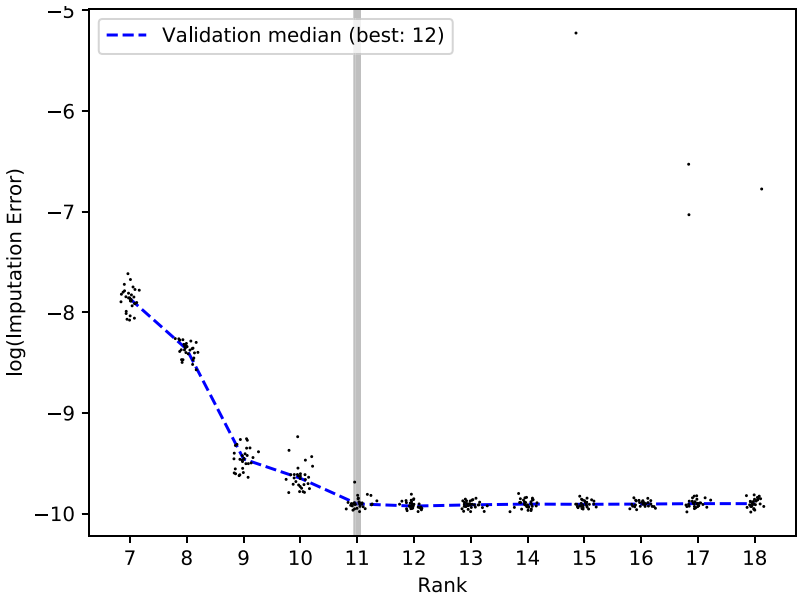

Figure 2. An example run of CV2K(0.01) on a simulated data set ((SP) iii). Dots represent runs, with their median denoted using a dashed line. The minimum is obtained for $K = 12$, and the final chosen K is marked in gray (11). The true number of components for this simulation is also 11.

As we show below, the results of CV2K are quite robust to p across a wide range, yet we hypothesize that fractions between $1/N$ and $1/M$ will work best. This requires further investigation in more size-varying data sets. In order to efficiently identify K^*, we use a binary search algorithm, starting from an arbitrary guess of 20. Once we find K^*, we scan a small window (± 3) around it to test for a better value. If there is a close K with a better median, we adopt it and repeat, otherwise we use Wilcoxon rank sum test to check the possibility of decreasing K and repeat.

2.4. CV2K-x: Eliminating the fraction hyperparameter

To eliminate the fraction hyperparameter, we propose a variant (CV2K-x) which is described in Algorithm 4 (showing only the difference from Algorithm 3). The idea is to combine cross validation on the rows of V to learn the H matrix (signatures) and cross validation on the columns to learn the W matrix (exposures; for the latter we use SciPy's non-negative least squares [15]). Finally, reconstruction errors are summed up from each column and row folds to produce a score for a given rank. When applying this method with number of

Algorithm 4 CV2K-x $(V, K_{\min}, K_{\max})$—replaces Algorithm 3: lines 2–7

2: Parameter: $F =$ number of sample folds
3: Randomly split V samples to F groups
4: **for** $k = K_{\min}, \ldots, K_{\max}$ **do**
5: **for** $f = 1, \ldots, F$ **do**
6: $V_{train} = V[\neg f]$
7: $V_{test} = V[f]$
8: $\_, H = \mathrm{NMF}(V_{train}, 1, k)$
9: $D_H = diag(\sum_{m=1}^{M} h_{1\,\mathrm{m}}, \ldots, \sum_{m=1}^{M} h_{Km})$
10: $H = D_H^{-1} H$
11: **for** $m = 1, \ldots, M$ **do**
12: $H_{train} = H[\neg m]$
13: $H_{train} = D_{H_{train}^{-1}} H_{train}$
14: $V_{curr} = V_{test}[\neg m]$
15: $W_{train} = \mathrm{nnls}(V_{curr}, H_{train})$
16: $H_{test} = H[m]$
17: $S[k, f] += \mathrm{ImputationError}(V_{test}, 0, W_{train}, H_{test})$

folds ranging from 5 to 30, we noticed that the results were the same or deviated by at most 1 across this range. Henceforth, we used 10-fold cross validation ($F = 10$).

2.5. Comparison to related methods

Two previous methods, Bi-CV [6] and scattered holdout [9], have explored the use of cross validation for learning the number of components in NMF. To demonstrate the utility of our new method, we also tested it against these two previous methods. We found that the Bi-CV method is not stable, producing results with no single local minimum. For example, on a sample simulated data set ((SA) ii]) in which the true number of components is 39, the Bi-CV method gives multiple local minima that are far from the true value (for $K =$ 15, 64, 76, 52, 29, in this order). To test the scattered holdout approach we used Alex Williams' implementation (http://alexhwilliams.info/itsneuronalblog/2018/02/26/crossval/). We adjusted the implementation (abbreviated scatCV below) to use median values (rather than mean) to score each rank, thus decreasing the problem of outliers.

Lin *et al* [10] also suggests using imputation to choose K. Their approach, implemented in the NNLM package, is based on masking 30% of the input matrix entries in cross validation and picking K to minimize the reconstruction error. See https://rdrr.io/cran/NNLM/f/inst/doc/Fast-And-Versatile-NMF.pdf for additional details.

2.6. Data description

2.6.1. Simulated data

The simulated data was generated and described in detail in [16] to evaluate SigProfiler and SignatureAnalyzer. Here, for each of the 12 data sets we evaluate our method on two sets of 'realistic' synthetic data: SP-realistic, based on SP's reference signatures and attributions, and SA-realistic, based on SA's reference signatures and attributions. For each of the (i) to (x) tests, the synthetic data sets were generated based on observed statistics for each signature in each cancer type. Different data sets can differ by the number of signatures, number of active signatures per samples (sparsity), number of mutations per sample (whole exome/genome sequencing), whether they reflect single cancer type or multiple types, and similarity between signatures. All these factors affect the difficulty of determining the number of components.

2.6.2. Real data

We analyzed breast cancer (BRCA), chronic lymphocytic leukemia (CLLE), and malignant lymphoma (MALY) mutation datasets from whole-genome sequences (WGS) from the International Cancer Genome Consortium (ICGC). We downloaded mutations from the ICGC data portal (https://docs.icgc.org/). For BRCA, we used release 22 to match Nik-Zainal *et al* [17], while for the other datasets we used the most recent release (release 27). These data sets had 560, 100 and 100 samples, respectively, and more than 1000 mutations for most samples.

In addition, we downloaded mutations from whole-exome-sequencing (WXS) data of 411 ovarian cancer (OV) patients from The Cancer Genome Atlas [18]. There are about 100 mutations per sample in this collection. We extracted pancreas (PANC) and prostate (PROST) mutations from WGS PCAWG data set

Table 2. The effect of Wilcoxon rank sum test on the performance of CV2K(1e-3) on simulated data. The third column presents the value of K* chosen by CV2K before and after the parsimony step (one value implies that the parsimony criterion did not affect the result.

Dataset	Truth	CV2K(1e-3)
(SP) i	11	14->11
(SA) i	20	20->19
(SP) ii	21	21->19
(SA) ii	39	40->38
(SP) iii	11	13->11
(SA) iii	19	20
(SP) iv	3	4->3
(SA) iv	3	3
(SP) v	11	11
(SA) v	26	29->24
(SP) vi	3	3
(SA) vi	3	5->3
(SP) vii.a	2	4->2
(SA) vii.a	2	2
(SP) vii.b	4	4
(SA) vii.b	4	7->4
(SP) vii.c	4	6->4
(SA) vii.c	4	5->4
(SP) viii	30	33->27
(SA) viii	30	34->32
(SP) ix	21	14->11
(SA) ix	39	20->14
(SP) x	21	22->19
(SA) x	39	41->38
Percent dev.		22.09->7.10
Avg. distance		2.36->2

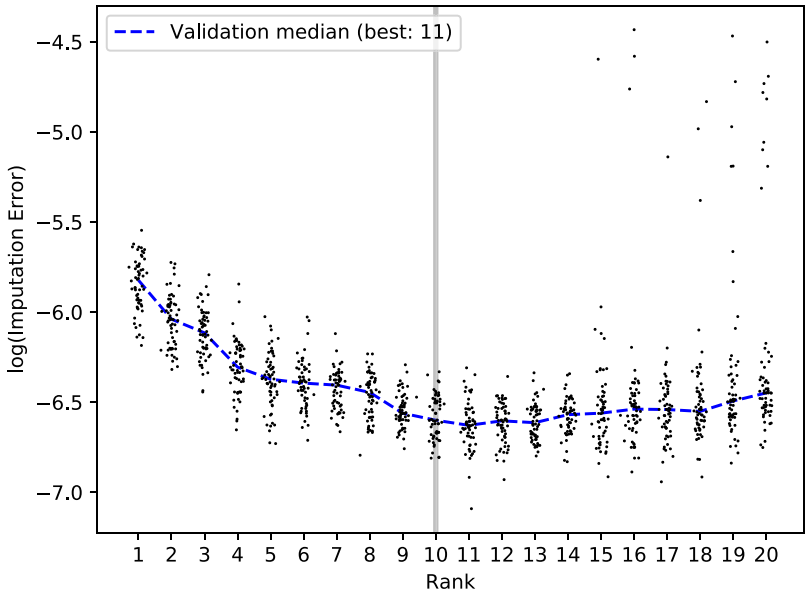

Figure 3. An example run of CV2K(0.01) on a real data set (MALY). The minimum is on $K = 11$, and the final K is marked in gray (10). The COSMIC number of signatures is 6.

linked from COSMIC. These data sets have 326 and 286 samples, respectively, with more then 1000 mutations for most samples.

2.7. Evaluation

For evaluation, we use three figures of merit: (i) Average percent deviation is the average over $\frac{|\text{Truth}-\text{Predicted}|}{\text{Truth}} * 100$, emphasizing errors in data sets with a small number of signatures. (ii) Average distance is the average over $|\text{Truth} - \text{Predicted}|$, emphasizing large deviations from the true value especially in data

Table 3. Performance evaluation on simulated data. Best values in each row are highlighted in bold. Comparison between SignatureAnalyzer (SA), SigProfiler (SP), Akaike information criterion (AIC), Alex Williams' implementation (scatCV), NNLM package and our CV2 method.

Dataset	Truth	SA	SP	AIC	scatCV	NNLM	CV2K(1e-3)	CV2K(0.01)	CV2K-x
(SP) i	11	**11**	**11**	**11**	**11**	8	**11**	**11**	**11**
(SA) i	20	17	19	17	**20**	12	19	19	**20**
(SP) ii	21	19	19	19	18	1	19	**20**	17
(SA) ii	39	32	30	32	15	6	**38**	32	35
(SP) iii	11	10	10	9	**11**	1	**11**	**11**	14
(SA) iii	19	16	17	16	21	17	**20**	**20**	22
(SP) iv	3	**3**	2	**3**	**3**	26	**3**	**3**	**3**
(SA) iv	3	**3**	**3**	**3**	**3**	31	**3**	**3**	**3**
(SP) v	11	**11**	**11**	**11**	10	1	**11**	**11**	**11**
(SA) v	26	23	20	**24**	7	1	**24**	19	19
(SP) vi	3	4	**3**	**3**	**3**	1	**3**	**3**	**3**
(SA) vi	3	**3**	2	**3**	**3**	1	**3**	**3**	**3**
(SP) vii.a	2	**2**	**2***	**2**	**2**	6	**2**	**2**	**2**
(SA) vii.a	2	**2**	**2***	**2**	**2**	5	**2**	**2**	**2**
(SP) vii.b	4	**4**	**4***	**4**	**4**	1	**4**	**4**	**4**
(SA) vii.b	4	**4**	**4***	**4**	**4**	3	**4**	**4**	**4**
(SP) vii.c	4	**4***	**4**	**4**	5	1	**4**	**4**	**4**
(SA) vii.c	4	**4***	**4**	**4**	**4**	**4**	**4**	**4**	**4**
(SP) viii	30	26	21	25	20	1	**27**	22	26
(SA) viii	30	25	21	24	22	5	**32**	34	27
(SP) ix	21	12	8	11	**18**	1	11	11	13
(SA) ix	39	11	5	11	11	1	14	**16**	13
(SP) x	21	19	16	19	19	1	19	**20**	18
(SA) x	39	32	28	32	13	2	**38**	36	35
Percent dev.		11.8	17	11.15	17.33	144.03	**7.10**	9.12	10.50
Avg. distance		3	4.2	3.08	5.08	14.04	**2**	2.64	2.76
Overall wins		11	10	13	13	1	**19**	17	13

sets with a high number of signatures. (iii) Overall wins is the number of datasets where the predicted rank is closest to the true one. In addition, we use a signed version of the deviation, which is the average over $\frac{\text{Predicted}-\text{Truth}}{\text{Truth}} * 100$, to examine whether we over- or under-estimate the true rank.

3. Results

In the following we describe the application of CV2K to simulated and real mutations data sets. For brevity, we use CV2K(p) to denote the application of CV2K with a fraction p of hidden values in its cross validation procedure, where CV2K-x corresponds to the variant without the fraction hyperparameter. For CV2K, we used the Kullback–Leibler divergence (KL) variant of NMF, to produce the results mentioned below.

3.1. Simulated data

Our first set of results is focused on simulated data from [16] of different characteristics, accounting for different numbers of signatures and different similarity relationships between the signatures. We examine our performance on simulated data over a range of fractions p of hidden entries, as summarized in table 4, observing that the results are relatively robust across a wide range of values for this parameter and that—as a general rule—our method tends to under-estimate K, with smaller fraction values leading to closer estimations compared to larger values. We further observe that the use of the Wilcoxon rank sum step improves the performance of the algorithm, as summarized in table 2.

We compare our performance against two widely used tools for mutational signatures extraction, SigProfiler (SP) and SignatureAnalyzer (SA). We report the results of the authors of these tools on the simulated data, also published in [16]. As SigProfiler and SignatureAnalyzer miss results on few data sets (4 and 2, respectively), to aid the comparison we fill all missing values with the optimal value (marked with an asterisk in table 3). We further compare to a general model selection approach, the Akaike information criterion (AIC), as well as to the scatCV and NNLM methods reviewed above. The results are summarized in table 3 and depicted in figure 4 (left panel). They show the superiority of our approach (regardless of the choice of the hidden fraction p to use) over the existing ones.

Table 4. Performance of CV2K on simulated data, for different fractions p of hidden values in the cross validation procedure. Best values in each row are highlighted in bold.

p	1e−3	5e−3	0.01	0.05	0.1	x
Percent dev.	7.10	**6.67**	9.12	9.42	11.07	10.50
Avg. distance	2	**1.92**	2.64	2.68	3.2	2.76
Signed dev.	−6.11	−5.39	−7.57	−8.22	−9.87	−6.91

Table 5. Performance evaluation on real data. Comparison between SignatureAnalyzer (SA), Akaike information criterion (AIC), Alex Williams' implementation (scatCV), NNLM package and our CV2K method.

Dataset	COSMIC	SA	AIC	scatCV	NNLM	CV2K(1e−3)	CV2K(0.01)	CV2K−x
BRCA	12	16	23	3	2	9	**11**	9
OV	3	**3**	4	2	9	**3**	**3**	1
CLLE	5	7–8	11	1	1	2	**4**	**4**
MALY	6	15	22	8	4	**7**	10	9
PANC	6+	90	26	4	1	**7**	11	8
PROST	3	77	18	**3**	**3**	**3**	4	**3**
Percent dev.		681.67	224.17	42.50	80	**19.72**	35.28	32.50
Avg. distance		28.83	11.50	3	4.50	**1.33**	2	1.83
Overall wins		1	0	1	1	**4**	3	2

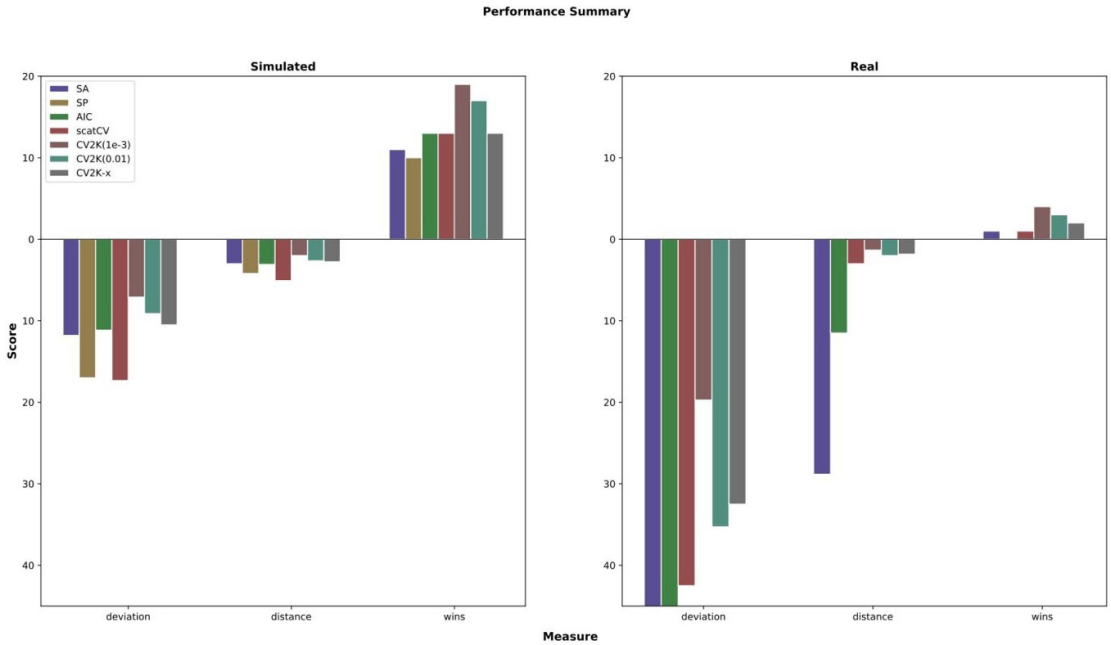

Figure 4. Summary of performance evaluation on simulated and real data. Comparison between SignatureAnalyzer (SA), SigProfiler (SP), Akaike information criterion (AIC), Alex Williams' implementation (scatCV) and our CV2K method.

3.2. Real data

Next, we evaluated the different methods on six real mutation data sets described above. As the true number of components or signatures in these applications is not known, we used the number of signatures suggested in COSMIC [19] as a surrogate. For SA we report in these tests the most frequent choice across 10 runs. SP was not used in these tests due to its high running time (using the original matlab code, we could not execute the new python code on our servers) and lack of automation. The results are summarized in table 5 and depicted in figure 4 (right panel), showing that our method yields numbers of signatures that are in higher agreement with the current knowledge on the corresponding data sets as represented by COSMIC.

4. Conclusion

In summary, we proposed a fully automated method, CV2K, to select the number of components in NMF. In the context of mutational signatures, we showed that CV2K outperforms state-of-the-art methods on simulated and real data.

The success of our method, in particular in comparison to previous methods that are based on similar cross validation principles, can be attributed to several factors: (i) a comprehensive framework for NMF optimization based on block coordinated descent coupled with multiple initializations to avoid local minima; (ii) the use of a parsimony consideration to resolve cases in which the imputation error reaches a plateau, making CV2K less prone to overestimation of K; and (iii) an exploration of a range of values for the fraction of hidden entries and a novel approach to eliminate this hyper-parameter.

Although the number of components is the only obvious hyper-parameter in NMF, there is in fact another regularization parameter that is often used. This regularization parameter can also affect the number of signatures. For future work we suggest expanding the model search to include this parameter as well.

Acknowledgments

This study was supported in part by a fellowship from the Edmond J. Safra Center for Bioinformatics at Tel-Aviv University. RS was supported by the Koret-UC Berkeley-Tel Aviv University Initiative in Computational Biology and Bioinformatics.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/GalGilad/CV2K.

ORCID iD

Gal Gilad https://orcid.org/0000-0003-1913-8918

References

[1] Lee D D and Seung H S 1999 Learning the parts of objects by non-negative matrix factorization *Nature* **401** 10
[2] Brunet J P, Tamayo P, Golub T R and Mesirov J P 2004 Metagenes and molecular pattern discovery using matrix factorization *Proc. Natl. Acad. Sci. USA* **101** 4164–9
[3] Hutchins L N, Murphy S M, Singh P and Graber J H 2008 Position-dependent motif characterization using non-negative matrix factorization *Bioinformatics* **24** 2684–90
[4] Tan V Y F and Févotte C 2013 Automatic relevance determination in nonnegative matrix factorization with the /spl beta/-divergence *IEEE Trans. Pattern Anal. Mach. Intell.* **35** 1592–605
[5] Bro R, Kjeldahl K, Smilde A K and Kiers H A L 2008 Cross-validation of component models: A critical look at current methods *Anal. Bioanal. Chem.* **390** 1241–51
[6] Owen A B and Perry P O 06 2009 Bi-cross-validation of the SVD and the nonnegative matrix factorization *Ann. Appl. Stat.* **3** 564–94
[7] Gabriel K R 2002 Le biplot—outil d'exploration de données multidimensionnelles *J. Soc. Française Stat.* **143** 5–55
[8] Wold S 1978 Cross-validatory estimation of the number of components in factor and principal components models *Technometrics* **20** 397–405
[9] Kanagal B and Sindhwani V 2010 Rank Selection in Low-rank Matrix Approximations: A Study of Cross-Validation for NMFs *Proc. Conf. Adv. Neural Inf. Process* **1** 01
[10] Lin X and Boutros P 2020 Optimization and expansion of non-negative matrix factorization *BMC Bioinform.* **21** 12
[11] Alexandrov L B, Nik-Zainal S, Wedge D C, Campbell P J and Stratton M R 2013 Deciphering signatures of mutational processes operative in human cancer *Cell Rep* **3** 246–59
[12] Haradhvala N J *et al* 05 2018 Distinct mutational signatures characterize concurrent loss of polymerase proofreading and mismatch repair *Nat. Commun.* **9** D777–83
[13] Vavasis S A 2010 On the complexity of nonnegative matrix factorization *SIAM J. Optim.* **20** 1364–77
[14] Li L, Lebanon G and Park H 2012 Fast Bregman divergence NMF using Taylor expansion and coordinate descent *Proc. of the 18th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining* KDD'12 (*New York, NY, USA*: Association for Computing Machinery) pp 307–15
[15] Lawson C L and Hanson R J 1995 *Solving Least Squares Problems* (Philadelphia, PA: Society for Industrial and Applied Mathematics) (https://doi.org/10.1137/1.9781611971217)
[16] Alexandrov L *et al* 02 2020 The repertoire of mutational signatures in human cancer *Nature* **578** 94–101
[17] Nik-Zainal S *et al et al* 2016 Landscape of somatic mutations in 560 breast cancer whole-genome sequences *Nature* **534** 47
[18] Tomczak K, Czerwinska P and Wiznerowicz M 2015 The cancer genome atlas (TCGA): An immeasurable source of knowledge *Contemp. Oncol. (Pozn)* **19** A68–77
[19] Forbes S A *et al* 11 2016 COSMIC: somatic cancer genetics at high-resolution *Nucleic Acids Res.* **45** D777–83