Scientific Research Publishing

Open Journal of Applied Sciences, 2025, 15(10), 3073-3093
https://www.scirp.org/journal/ojapps
ISSN Online: 2165-3925
ISSN Print: 2165-3917

# Optimal Predictive Modeling of Nonlinear Transformations: Innovative Applied Mathematics in an Artificial Intelligence System

**Jean Pacifique Nkurunziza[1,2*], Fulgence Nahayo[2], Ignat Anca[3], Thierry Nsabimana[2]**

[1]Doctoral School, Burundi University, Bujumbura, Burundi
[2]LURMISTA, ISTA, University of Burundi, Bujumbura, Burundi
[3]Faculty of Computer Science, University of Alexandru Ion Cuza, Iasi, Rumania
Email: *jean.nkurunziza@imsp-uac.org

**How to cite this paper:** Nkurunziza, J.P., Nahayo, F., Anca, I. and Nsabimana, T. (2025) Optimal Predictive Modeling of Nonlinear Transformations: Innovative Applied Mathematics in an Artificial Intelligence System. *Open Journal of Applied Sciences*, **15**, 3073-3093.
https://doi.org/10.4236/ojapps.2025.1510202

**Received:** September 5, 2025
**Accepted:** October 12, 2025
**Published:** October 15, 2025



## Abstract

In this paper, an Optimal Predictive Modeling of Nonlinear Transformations "OPMNT" method has been developed while using Orthogonal Nonnegative Matrix Factorization "ONMF" with the Kullback-Leibler "KL" divergence. Innovative algorithm includes a Term Frequency-Inverse Document Frequency score smoothing technique and incorporates parameters optimized through cross-validation using the Nelder-Mead method. This technique has numerous applications in Artificial Intelligence Systems (AIS), including extracting nonlinear features for document classification. To make our technique powerful, we apply nonlinear transformations such as logarithms, square roots, and hyperbolic tangents. Our results demonstrate that the *OPMNT-KL-ONMF* method yields significantly higher accuracy on average compared to untransformed datasets, underscoring the critical importance of selecting appropriate transformation functions to enhance the classification capabilities of the KL-ONMF model. Future work will involve integrating these transformation functions into a neural network framework to explore new techniques for improving performance.

## Keywords

Optimal Predictive Modeling, Nonlinear Transformation, Artificial Intelligence Systems, Parameter Optimization, KL-ONMF Model, Clustering, Cross-Validation

## 1. Introduction

The non-negative matrix factorisation (NMF) method, which was first used in the

paper [1], is a technique that allows a non-negative matrix $X$ to be decomposed into two non-negative factors given by $W$ and $H$, where

$$X \approx W \times H,$$

where $W$ is an $m \times r$ matrix with elements in $\mathbb{R}^+$, and $H$ is an $r \times n$ matrix with nonnegative elements. Here, $m$ represents the number of features, $n$ represents the number of observations or samples, while $r$ denotes the rank or dimensionality of the feature subspace of $X$.

Let $\{x_1, x_2, \cdots, x_n\} \in \mathbb{R}^m$ be a set of data. For any column vector $x_j$ in $X$, we can express it as

$$x_j \approx \sum_{k=1}^{r} w_k h_{kj},$$

indicating that $x_j$ can be approximated by a linear combination of the basis vectors (*i.e.*, all column vectors of $W$), with the components of $H$ acting as the weight coefficients. Thus, $H$ is also known as the coefficient matrix, allowing the feature vector to be computed as a linear combination of the basis vectors.

Originally proposed for feature extraction, the NMF nonnegative nature makes it suitable for various non-negative processing tasks, underpinned by strong theoretical foundations and interpretability [2]. In pattern recognition, dimensionality reduction is crucial for efficiently managing high-dimensional data and improving recognition accuracy. While the K-means algorithm is frequently used for clustering [3], its performance often degrades in high-dimensional spaces, underscoring the necessity for effective dimensionality reduction techniques.

In addition to pattern recognition, NMF has applications across various fields, including dimension reduction [4], image processing [5], speech processing [6] [7], spectral analysis [8] [9], DNA expression analysis [10] [11], microRNA-disease analysis [12] [13], social network analysis [14], text analysis [15], hyperspectral signal unmixing [16], and blind spectral unmixing [17] [18]. As data size increases, so does the demand for processing and storage. This has led researchers to develop new algorithms for linear dimensionality reduction (LDR), a data analysis method that reduces the dimensionality of information with minimal loss of information before analysis. The transformation of data is an effective means to improve the linear representation. The idea is to transform the data via a function $f$ to be learned, and use $f(X) = [f(\theta_1), f(\theta_2), \cdots, f(\theta_n)]$ and apply the factorization to $f(X)$ instead of $X$, *i.e.* $f(X) \approx W \times H$.

We assume that each data point is associated with a label $y_1, \cdots, y_n$. In the most basic case, they have labels $y_i$ that are binary, *i.e.*, they can take two values representing two classes. For multi-class cases, the labels can be represented using one-hot encoding [19]. A prominent example of a non-linear transformation applied to count data in document classification is the term frequency-inverse document frequency (TF-IDF) transformation [20] [21]. In this context, $X(i, j)$ denotes the frequency of word $i$ appearing in document $j$, and the TF-IDF transformation maps $X$ to $f(X)$ where

$$X(i,j) = f\left(X(i,j)\right).$$

In [22] [23], document classification is achieved through NMF and KL-NMF. Additionally, [24] demonstrates the application of non-linear transformations to enhance the effectiveness of these classification methods. This motivates our exploration of improved transformations, which we aim to apply to various document datasets. This motivates our exploration of some nonlinear transformations, which we intend to apply to various document datasets to enhance the results obtained in [25].

Another effective method is the probabilistic IDF [26], which modifies the basic IDF calculation to incorporate probabilities, ensuring that terms that do not appear in a document are still assigned meaningful weights. Jelinek-Mercer smoothing [27] [28], initially developed for language modeling, can also be adapted for TF-IDF by interpolating maximum likelihood estimates with background distributions.

Moreover, Katz [29] and Witten-Bell [30] smoothing techniques provide further enhancements by allowing for interpolation between observed and unobserved frequencies, improving robustness in document classification tasks at language analysis. It is important to emphasize the use of these smoothing techniques to improve the performance of TF-IDF so that the weights assigned to terms indicate the general importance of the terms in the overall corpus.

**Outline and Contribution of the Paper**

This paper proposes a new method for document classification. As detailed in Section 2, KL-ONMF serves as the reference for our method. We will smooth the TF-IDF and incorporate parameters (see Section 3), which we will optimize using the fminsearch function in MATLAB with a machine learning approach. We apply non-linear transformations in order to capture non-linear characteristics better. We will show that this new approach is much better than KL-ONMF on the original data for document classification (see Section 4).

## 2. Alternating Optimization for ONMF with the KL Divergence

In this section, we focus on Alternating Optimization for Orthogonal Nonnegative Matrix Factorization (ONMF) with the Kullback-Leibler (KL) Divergence. The goal is to minimize the KL Divergence between the given matrix $X$ and the product of two matrices $W$ and $H$, while enforcing specific constraints on $H$.

The optimization problem can be formulated as follows:

$$\min_{W\in\mathbb{R}^{m\times r},H\in\mathbb{R}^{r\times n}} D_{\mathrm{KL}}\left(X, WH\right) \text{ subject to } H \geq 0 \text{ and } HH^{\top} = I_r.$$

For ONMF utilizing the KL Divergence, it is crucial that $X$ is componentwise nonnegative, reflecting the inherent properties of KL Divergence, which is only defined for non-negative matrices. This non-negativity is essential, as it ensures that the divergence measurement remains valid and interpretable in the context of probability distributions.

In contrast, ONMF using the Frobenius norm does not impose such restrictions, allowing for broader applications but potentially leading to less interpretable results in probabilistic terms. The choice of KL Divergence in this context emphasizes our focus on modeling non-negative data, making it particularly suitable for applications like document classification or image processing, where the underlying data naturally adheres to non-negativity constraints.

Moreover, the constraint $HH^\top = I_r$ ensures that the matrix $H$ retains orthogonality, which is vital for preserving the interpretability of the factors derived from the decomposition. This orthogonality condition aids in distinctly separating the components, allowing for clearer insights into the latent structures represented by the factors in $W$ and $H$.

This formulation highlights the methodological rigor behind ONMF with KL Divergence, ensuring that the results are both mathematically sound and practically applicable to real-world datasets. The update rules for $W$ and $H$ in ONMF with the KL Divergence highlight the different optimization strategies compared to Frobenius ONMF, making KL-ONMF less sensitive to outliers and more focused on data points with smaller norms.

**Algorithm 1** summarizes the alternating optimization scheme, known as KL-ONMF.

**Algorithm 1** KL-ONMF - Alternating optimisation algorithm for ONMF with KL divergence [25]

**Require:** Nonnegative matrix: $X \in \mathbb{R}_+^{m\times n}$, Initialization: $W \in \mathbb{R}_+^{m\times r}$, factorization rank: $r$, maximum number of iterations: maxiter, convergence criterion: $\delta \ll 1$, small parameter: $\epsilon \ll 1$.

**Ensure:** Matrices $W \in \mathbb{R}_+^{m\times r}$ and $H \in \mathbb{R}_+^{r\times n}$ with $HH^\top = I_r$ such that $D_{\mathrm{KL}}(X, WH)$ is minimized.

$t = 1$, $H = 1$, $H^{(p)} = 0$.
**while** $t \le$ maxiter and $\|H - H^{(p)}\|_F \ge \delta$ **do**
  *% Update $H$ for $W$ fixed*
  $H^{(p)} \leftarrow H$. *% $H^{(p)}$ is the previous iterate to monitor convergence.*
  Normalized $W_n$: $W(:,k) \leftarrow W(:,k)/e^\top W(:,k)$ for all $k$.
  Compute $A = \log(W_n + \epsilon)^\top X \in \mathbb{R}^{r\times n}$.
  Let $\mathcal{K}_k = \{j \mid A(k,j) > A(k',j) \text{ for all } k' \ne k\}$.
  Update $H(k,j) \leftarrow \frac{e^\top X(:,j)}{e^\top W(:,k)}$ for all $k$ and $j \in \mathcal{K}_k$.
  Scale $H$: $H(k,:) \leftarrow \frac{H(k,:)}{\|H(k,:)\|_2}$ for all $k$.
  *% Update $W$ for $H$ fixed*
  $W(:,k) \leftarrow \frac{X(:,\mathcal{K}_k)e}{H(k,:)e}$ for all $k$.
  $t \leftarrow t + 1$
**end while**

It's a simple yet effective and highly scalable alternating optimization algorithm. It's an alternating optimization algorithm, which is simple but efficient and highly scalable, running in $O(\mathrm{nnz}(X)r)$ operations where $\mathrm{nnz}(X)$ is the number of non-zero entries in the data matrix, and $r$ is the number of clusters. Applied on documents and hyperspectral images, KL-ONMF demonstrates its performance over ONMF using the Frobenius norm by providing better clustering results, and the convergence time is always smaller on average.

## 3. Optimal Predictive Modeling of Nonlinear Transformations

In this section, we develop smoothing techniques to enhance the calculations of Term Frequency-Inverse Document Frequency (TF-IDF). These techniques ad-

dress limitations associated with raw term frequency counts, ensuring a more balanced representation of term importance.

### 3.1. TF-IDF Modeling

In [31], the authors define TF-IDF as a combination of two statistics: term frequency (TF) and inverse document frequency (IDF). The term frequency $tf(i,j)$ quantifies how often term $i$ appears in document $j$ [32], and is expressed as:

$$tf(i,j) = \frac{f_{i,j}}{\sum_{i' \in j} f_{i',j}}. \tag{1}$$

where $f_{i,j}$ is the raw count of the term in the document, and $\sum_{i' \in j} f_{i',j}$ is the total number of terms in document $j$. Inverse document frequency measures the amount of information provided by a word, indicating whether it is present in all documents, and is defined as the inverse fraction, on a logarithmic scale, of documents that contain the word. It is given by:

$$idf(i) = \log\left(\frac{N}{\left|\{j \in J : i \in j\}\right|}\right). \tag{2}$$

where $N$ is the total number of documents in the corpus, and $|j|$ is the total number of terms in document $j$. This approach effectively balances the frequency and rarity of terms, making it a powerful tool for information retrieval and text analysis. In the next section, we will introduce smoothing techniques to enhance the TF-IDF representation, aiming to improve the robustness and performance of our classification method.

### 3.2. Smoothing Technique for TF-IDF in Optimal Predictive Modeling

To address the limitations of raw term frequency counts and ensure a more balanced representation of term importance, we introduce smoothing techniques in the calculation of both term frequency (TF) and inverse document frequency (IDF). The term frequency is recalculated with a smoothing term to prevent zero counts from disproportionately affecting the results. Similarly, the inverse document frequency is adjusted to ensure that terms common across many documents do not dominate the weighting.

The term frequency $\mathrm{TF}(i,j)$ is computed as:

$$\mathrm{TF}(i,j) = \frac{f_{i,j} + \text{smoothing}}{\sum_{i' \in j} f_{i',j} + \text{smoothing}}, \tag{3}$$

where smooth = 1 for Laplace smoothing or add-1 smoothing [33]. The inverse document frequency $\mathrm{IDF}(i)$ is calculated to measure the importance of the term across the document corpus:

$$\mathrm{IDF}(i) = \log\left(\frac{N + \text{smoothing}}{\mathrm{DF}(i) + \text{smoothing}}\right). \tag{4}$$

where $\mathrm{DF}(i)$ represents the number of documents containing the term $i$. This formulation helps to mitigate the impact of terms that appear in a large number of documents, ensuring that the IDF value remains informative. The TF-IDF weighting of a term is therefore the product of its TF term frequency Equation (3) and its IDF document inverse frequency Equation (4). It is defined by Equation (5):

$$X(i,j) = \mathrm{TF}(i,j) \times \mathrm{IDF}(i). \tag{5}$$

In the following, we start with a matrix $X$ representing word occurrences, where each element $X(i,j)$ indicates how many times word $i$ appears in document $j$. The algorithm computes the Term Frequency (TF) for each word in a document, reflecting its relative importance, and the Inverse Document Frequency (IDF) measures how common or rare a word is across all documents. The resulting TF-IDF scores highlight the significance of words in the context of the documents. The detailed algorithm is described below.

### 3.3. Parameterization of TF-IDF Score with Custom Adjustments

To further refine the TF-IDF score, we introduce adjustable parameters $a$, $b$, and $c$. These parameters enable customization based on the dataset's characteristics, thereby enhancing the overall performance of clustering algorithms. The TF-IDF score incorporating parameters to be optimized can be expressed as follows:

$$X(i,j) = a \cdot \left( X(i,j)^b \right) + c \cdot X(i,j). \tag{6}$$

We note that for the classic TF-IDF, we have $a = b = 1$ and $c = 0$ or $a = b = 0$ and $c = 1$. In our new model, each parameter plays a crucial role in fine-tuning the model to capture the underlying patterns in the data: parameter $a$ scales the contribution of the transformed term frequencies. By adjusting $a$, we can amplify the significance of certain terms based on their frequency, ensuring that important terms are more prominently represented in the final feature set. Parameter $b$ serves as the exponent applied to the term frequency matrix $X$. This exponentiation can accentuate the differences among term frequencies, effectively emphasizing more frequent terms while diminishing the influence of less common ones, and parameter $c$ adjusts the influence of the original TF-IDF scores.

### 3.4. Innovative Nonlinear Transformations: Boosting Predictive Performance in AI Applications

In this section, we explore three nonlinear transformations that enhance the model's ability to capture complex relationships within data. These transformations are inspired by principles from neural networks and include the logarithm function, the square root function, and the hyperbolic tangent function. Each transformation serves a specific purpose in refining the feature representation of TF-IDF scores.

**Logarithm Function**

We prefer the logarithm function due to its effectiveness in reducing skewness in data distributions. Many real-world datasets exhibit right skewness; applying a logarithmic transformation can render the data more symmetric, thereby improving the performance of clustering algorithms. The logarithmic transformation can be expressed as:

$$f(x) = \log(1+x). \tag{7}$$

Moreover, logarithmic scaling emphasizes smaller differences in low-value ranges, which is particularly beneficial for TF-IDF scores. This allows for a more nuanced understanding of infrequent terms that may still hold significant semantic weight. Additionally, by compressing the range of values, logarithmic transformations mitigate the influence of outliers, making clustering algorithms less sensitive to extreme values.

**Square Root Function**

The square root function is particularly advantageous for processing non-negative TF-IDF scores, as it emphasizes larger values while preserving the non-negativity of the data. This property is crucial for highlighting important terms that appear more frequently within the dataset, thereby amplifying their significance in the analysis. We use the expression given by:

$$f(x) = \sqrt{x}. \tag{8}$$

Moreover, the square root transformation effectively mitigates the impact of extremely high values, enabling clustering algorithms to concentrate on the overall structure of the data rather than being skewed by a few dominant terms. Since many clustering methods rely on distance measures, such as Euclidean distance, the square root transformation helps to maintain the relative distances between data points. This results in more accurate and meaningful clustering outcomes, enhancing the model's ability to identify distinct groups within the data.

**Hyperbolic Tangent Function**

The hyperbolic tangent function outputs values in the range of −1 to 1, which is particularly beneficial for clustering algorithms that assume centered data. This symmetry facilitates the model's ability to learn balanced representations of both positive and negative influences within the dataset.

$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{9}$$

Additionally, the tanh function introduces strong nonlinearity, which aids in capturing complex relationships among data points. This capability is crucial in clustering, where relationships may not be linearly separable. In the context of neural networks, the tanh function can also lead to more efficient gradients during backpropagation, potentially improving convergence rates in learning models.

In summary, we have chosen these transformations because they introduce nonlinearity into the model, allowing it to better learn from the underlying patterns in the data. The effectiveness of these nonlinear transformations is further

enhanced by the previously optimized parameters $a$, $b$, and $c$. By refining the representation of term importance through parameter tuning, we ensure that these transformations can operate on the optimized TF-IDF scores $x$, resulting in a more relevant and expressive feature space. This integrated approach—comprising smoothing, parameter optimization, and nonlinear transformations—aims to enhance clustering accuracy and improve document classification and retrieval outcomes.

### 3.5. Algorithm: Optimizing TF-IDF Score with Smoothing Techniques

**Algorithm 2** summarizes the alternating optimization scheme, which we refer to as KL-ONMF. The index $i$ refers of terms in the TF-IDF vector (ranging from 1 to $n$), and $j$ is the index of documents (for each term $t_i$, $j$ varies based on the number of documents).

**Algorithm 2** : OPMNT-KL-ONMF: Optimizing TF-IDF Score with Smoothing

**Require:** A vector of TF-IDF scores TF_IDF $= \{t_1, t_2, \ldots, t_n\}$, parameter smoothing, $a$: scaling coefficient, $b$: exponent, $c$: adjustment factor.

**Ensure:** A vector of optimized TF-IDF scores TF_IDF_optimized $= \{o_1, o_2, \ldots, o_n\}$

**Initialization:**
  Create an empty vector TF_IDF_optimized of dimension $n$.
**Calculate Term Frequency with Smoothing:**
**for** each term $t_i$ where $i \in \{1, 2, \ldots, n\}$ **do**
  **for** each document $j$ where $j \in \{1, 2, \ldots, m\}$ **do**
    Calculate $TF(i, j)$:

$$TF(i,j) = \frac{f_{i,j} + \text{smoothing}}{\sum_{i' \in j} f_{i',j} + \text{smoothing}}$$

  **end for**
**end for**
**Calculate Inverse Document Frequency with Smoothing:**
**for** each term $t_i$ where $i \in \{1, 2, \ldots, n\}$ **do**
  Calculate $IDF(i)$:

$$IDF(i) = \log\left(\frac{N + \text{smoothing}}{DF(i) + \text{smoothing}}\right)$$

  **for** each document $j$ where $j \in \{1, 2, \ldots, m\}$ **do**
    Calculate $TF - IDF(i, j)$:

$$X(i,j) = TF(i,j) \times IDF(i)$$

  **end for**
**end for**
**Apply Parameters:**
**for** each term $t_i$ where $i \in \{1, 2, \ldots, n\}$ **do**
  **for** each document $j$ where $j \in \{1, 2, \ldots, m\}$ **do**
    Calculate the optimized score:

$$o_{ij} = a \cdot (X(i,j)^b) + c \cdot X(i,j)$$

  **end for**
**end for**
**Apply Nonlinear Transformations:**
**for** each optimized score $o_{ij}$ in TF_IDF_optimized **do**

$$o_{ij} = f(o_{ij})$$

**end for**
**Store Transformed Scores:**
Update TF_IDF_optimized[$ij$] with the result of the nonlinear transformation.
**Clustering Step:**
Use $H$ to obtain final cluster labels:
**for** each document $j$ where $j \in \{1, 2, \ldots, m\}$ **do**
  Assign cluster label $C_j$ as:

$$C_j = \arg\max_i H(i,j)$$

**end for**
Return the vector TF_IDF_optimized and cluster labels $C$.

**Algorithm 2** is designed to enhance the traditional TF-IDF scoring method by incorporating smoothing and nonlinear transformations. It takes as input a vector of TF-IDF scores along with several parameters, including smoothing and scaling coefficient $a$, exponent $b$, and adjustment factor $c$, with the goal to produce an optimized vector of TF-IDF scores and corresponding cluster labels.

### 3.6. Parameter Optimization Using Regularization

To identify optimal values for parameters $b$ and $c$, we leverage the fminsearch function in MATLAB, which performs unconstrained optimization. During this process, the parameter $a$ is fixed at a baseline value of 0.0001. Initial values for $b$ and $c$ are generated randomly within a defined range of 0 to 2, enabling a thorough exploration of the parameter space and facilitating the discovery of improved configurations.

In our optimization function, we incorporate a regularization term to mitigate the risk of overfitting. This term penalizes excessively large values of the parameters $b$ and $c$ by introducing a quadratic penalty, represented as $\lambda\sum_{i=1}^{k}\theta_i^2$, where $k=2$ denotes the number of parameters. Specifically, this is an L2 regularization [34] component that helps to prevent overfitting by penalizing large parameter values, with each $\theta_i$ representing a parameter being optimized. By doing so, we encourage the model to prioritize simpler configurations that generalize better to unseen data. The inclusion of this regularization term helps maintain a balance between fitting the training data and preserving the model's ability to perform well on validation sets, ultimately leading to more robust and reliable results.

### 3.7. Data Separation

In order to evaluate the robustness of the optimized methodology, it is essential to separate the data into training and test sets. This separation is crucial for assessing the model's ability to generalize to unseen data and is performed by allocating 80% of the documents to the training set and 20% to the validation set.

The 80/20 split is strategically chosen as it strikes a balance between providing ample data for training the model and retaining a substantial portion for evaluation. This division ensures that the model can effectively learn from the training data while maintaining the capability to accurately assess its performance on independent data. Such a balance is crucial for developing a robust model that can generalize well to new, unseen instances.

In this context, "training" refers to the phase in which the model actively learns from the training set, adjusting its parameters to minimize prediction errors among terms and their corresponding clusters. Conversely, "validation" is the process of evaluating the model's performance on unseen data, which is essential for understanding its generalization capabilities. This two-phase approach allows us to ensure that the model is not only fitting the training data well but is also capable of making accurate predictions in real-world scenarios.

Furthermore, the use of labels in this unsupervised task is limited strictly to evaluation purposes. During the training phase, labels are not utilized; instead, they serve as a benchmark for measuring the accuracy and effectiveness of the model's clustering outcomes. This methodology emphasizes our commitment to an unbiased evaluation process, ensuring that the model's performance is assessed based solely on its ability to identify and group similar documents without any prior knowledge of their labels.

To enhance the evaluation process, we also employ **cross-validation**. Specifically, we set the number of folds for cross-validation to *num_folds* and the number of repetitions to *num_repetitions*.

Cross-validation is a resampling technique that helps in assessing how the results of a statistical analysis will generalize to an independent dataset. By splitting the training set into several subsets (folds), we can train the model on a portion of the data while validating it on the remaining parts. This method is repeated for a specified number of folds, providing a more comprehensive evaluation of model performance.

1) Number of Folds: Setting *num_folds* = 5 means that the training data will be split into five subsets. The model will be trained on one subset and validated on the other, and this process will be repeated for each fold.

2) Number of Repetitions: With *num_repetitions* = 5, the entire cross-validation process will be repeated five times, ensuring that the model's performance is robust and less sensitive to the specific data split.

By using cross-validation, we can achieve a more reliable assessment of our model's performance, reducing the risk of overfitting and ensuring that the enhancements made to the TF-IDF calculations yield tangible benefits in practical applications.

## 4. Numerical Experiments

In this section, we use nonlinear functions for document clustering, and we compare the performance of KL-ONMF (Algorithm developed in [25]) applied to the original data with that of KL-ONMF applied to the transformed data using (Algorithm). All experiments were run on a LAPTOP 11$^{th}$ Gen Intel® Core™ i5-1135G7 @ 2.40GHz × 8 16,0 Go RAM.

**Initialization:**

Similar to *k*-means, ONMF algorithms can be initialized in a variety of ways. We take the approach suggested in [35] to simplify the presentation. The successive nonnegative projection algorithm (SNPA) [36] is used in this method to initialize $W$. It finds a subset of $r$ columns from $X$ that accurately represent the dataset's evenly distributed data points.

To refine the parameters during the optimization phase, we utilize the Nelder-Mead optimization algorithm [37]. For this algorithm, we initialize the parameters by generating random values. The Nelder-Mead algorithm iteratively refines these parameters based on the objective function (Equation (10)), which in our case is

designed to minimize the difference between the transformed data and the original data, while also incorporating a regularization term. By combining the SNPA for the initialization of $W$ and the Nelder-Mead method for parameter optimization, we aim to achieve robust and efficient convergence in the ONMF framework.

**Parameterization of Functions:**

To minimize the error rate of the objective functions and enhance accuracy, we begin by setting a fixed initial parameter value for $a$ at 0.0001. This choice ensures consistency across our optimization process, allowing us to focus on adjusting the other parameters, $b$ and $c$. The values for $b$ and $c$ are randomly initialized (see Section 3.6), specifically rounded to three decimal places to maintain precision. This randomness introduces variability in the initial conditions, which can help the optimization algorithm explore different configurations effectively. To find the optimal values of $b$ and $c$, we employ the fminsearch algorithm, which utilizes the Nelder-Mead method. This optimization technique is particularly beneficial for multidimensional problems where derivatives may not be easily computable. It iteratively adjusts the parameters based on the evaluation of the objective function, seeking to minimize the error rate. By systematically refining the values of $b$ and $c$, the algorithm converges towards a local parameterization that improves the model's accuracy in its predictions. Overall, this approach ensures a robust framework for optimizing the model's performance based on the given dataset.

**Objective Function:**

The objective function aims to minimize the sum of the squared deviations between the calculated and observed outflows. This approach ensures that the model's predictions closely align with the actual data. Let $\Theta$ be the set of parameters defined as:

$$\Theta = \{\theta_1, \theta_2\}$$

where $\theta_1$ represents the first coefficient influencing the model's predictions, and $\theta_2$ represents the second coefficient. The parameters in $\Theta$ are crucial for minimizing $J(\theta_1, \theta_2)$ while adhering to the constraints that require both $\theta_1$ and $\theta_2$ to be non-negative. This ensures that the model parameters remain within a physically meaningful range, ultimately leading to a more accurate representation of the relationship between the inputs and outputs.

Mathematically, the objective function can be expressed as:

$$\min J(\theta_1, \theta_2) = \|X_{\text{train}} - Y\|_F^2 + \begin{cases} 1\times 10^6 & \text{if } \theta_1 < 0 \text{ or } \theta_2 < 0 \\ 0 & \text{otherwise} \end{cases} + \lambda\left(\theta_1^2 + \theta_2^2\right) \quad (10)$$

where $J(\theta_1, \theta_2)$ measures how well the model's predictions (calculated outflows) match the actual observed outflows. The goal is to find the parameters $\theta_1$ and $\theta_2$ that minimize this function. The Frobenius norm $\|.\|_F^2$ quantifies the difference between the predicted and actual values. Here, $X_{\text{train}}$ represents the observed outflows, while $Y$ denotes the calculated outflows based on the model

and parameters $\theta_1$ and $\theta_2$. The calculated outflows $Y$ can be expressed as:

$$Y = W \cdot H \tag{11}$$

where $W$ is the matrix of features and $H$ is the matrix of coefficients derived from the parameters $\theta_1$ and $\theta_2$. The Frobenius norm computes the squared difference between these two sets of values, aggregating the errors across all observations. To discourage invalid parameter values, a penalty term is added, ensuring that only non-negative parameters are considered in the optimization process. To prevent overfitting by penalizing large values of the parameters, we add a regularization coefficient $\lambda$ that controls the strength of this penalty, balancing the fit of the model to the data against the complexity of the model.

**Document Data Sets:**

We cluster the 14 document data sets from [38] using KL-ONMF. Table 1 provides not only the names of the data sets but also their dimensions (where $m$ represents the number of words and $n$ the number of documents) and the number of clusters $r$.

Table 1. Summary of text datasets (for each dataset, $n$ is the total number of documents, $m$ the total number of words, $r$ the number of classes, $\overline{n}_c$ the average number of documents per class, and *Balance* the size ratio of the smallest class to the largest class).

| Data | $n$ | $m$ | $r$ | $\overline{n}_c$ | Balance |
|---|---|---|---|---|---|
| ngsim | 2998 | 15,810 | 3 | | |
| classic | 1073 | 30 | 20 | | |
| ohscal | 11,162 | 11,465 | 10 | 1116 | 0.437 |
| k1b | 2340 | 21,839 | 6 | 390 | 0.043 |
| hitech | 2301 | 10,080 | 6 | 384 | 0.192 |
| reviews | 4069 | 18,483 | 5 | 814 | 0.098 |
| sports | 8580 | 14,870 | 7 | 1226 | 0.036 |
| la1 | 3204 | 31,472 | 6 | 534 | 0.290 |
| la12 | 6279 | 31,472 | 6 | 1047 | 0.282 |
| la2 | 3075 | 31,472 | 6 | 513 | 0.274 |
| tr11 | 414 | 6429 | 9 | 46 | 0.046 |
| tr23 | 204 | 5832 | 6 | 34 | 0.066 |
| tr41 | 878 | 7454 | 10 | 88 | 0.037 |
| tr45 | 690 | 8261 | 10 | 69 | 0.088 |

**Accuracy:**

Accuracy is defined as the proportion of correctly classified data points relative to the total number of data points. In the setting of document clustering, the accuracy of a computed disjoint clustering $\left\{\tilde{C}_i\right\}_{i=1}^{r}$ compared to the true disjoint clusters $C_i \subset \{1,2,\cdots,n\}$ for $1 \le i \le r$ is defined as follows:

$$\text{accuracy}\left(\left\{\tilde{C}_i\right\}_{i=1}^{r}\right)=\min_{\pi\in[1,2,\cdots,r]}\frac{1}{n}\sum_{i=1}^{r}\left|C_i\cap\tilde{C}_{\pi(i)}\right|. \tag{12}$$

The correctly classified data points are those that have been assigned to the same cluster as their original data. The clustering process was carried out using the *K*-means algorithm.

**Running time:**

Table 2 summarizes the execution times associated with the objective function evaluated through the Nelder-Mead optimization algorithm across various datasets.

Table 2. Execution times of datasets.

| Dataset | Min Time (s) | Max Time (s) | Mean Time (s) |
|---|---|---|---|
| ng3sim | 1.40 | 2.57 | 1.84 |
| classic | 172.11 | 215.82 | 193.40 |
| ohscal | 4.45 | 31.01 | 13.46 |
| k1b | 30.78 | 31.28 | 31.05 |
| hitech | 1.58 | 1.80 | 1.65 |
| reviews | 4.60 | 48.19 | 19.15 |
| sports | 3.96 | 6.90 | 5.48 |
| la1 | 2.56 | 2.78 | 2.66 |
| la12 | 4.39 | 8.58 | 6.30 |
| la2 | 2.12 | 55.91 | 20.07 |
| tr11 | 0.91 | 1.18 | 1.01 |
| tr23 | 0.73 | 0.75 | 0.74 |
| tr41 | 1.14 | 1.50 | 1.31 |
| tr45 | 1.53 | 1.67 | 1.60 |

The results indicate substantial variability in processing times, with the "classic" dataset exhibiting the highest execution times, ranging from 172.11 to 215.82 seconds. This suggests that this dataset may present more complex optimization challenges or larger data sizes. Conversely, datasets like "ng3sim" and "hitech" show significantly lower mean execution times of 1.84 and 1.65 seconds, respectively, indicating efficient convergence during optimization. The "la2" dataset, with a maximum time of 55.91 seconds, points to potential outliers that may require further analysis to understand the underlying causes of increased processing time. Meanwhile, datasets such as "tr11," "tr23," "tr41," and "tr45" demonstrate consistent execution times, which may reflect similar optimization landscapes. Overall, these execution times underscore the impact of dataset characteristics on the performance of the Nelder-Mead optimization algorithm, highlighting opportunities for further refinement in handling more computationally intensive datasets.

### Optimal Parameters

In this section, we present the optimal parameters identified for each dataset through cross-validation. The table below summarizes these parameters, which play a crucial role in enhancing the performance of our model. The first column of the table lists the datasets. To ensure stability and prevent excessive fluctuations during optimization, we fixed the parameter $a$ at 0.0001 rather than allowing it to be optimized alongside $b$ and $c$. This choice is guided by the observation that large variations in $a$ can lead to significant distortions in the TF-IDF scores, potentially overshadowing the contributions of the other parameters. By maintaining $a$ at a low constant value, we provide a baseline influence that stabilizes the model while allowing $b$ and $c$ to be fine-tuned for better performance.

In a sensitivity analysis, we examined the effects of varying $a$ around its fixed value. The results indicated that small deviations from 0.0001 did not substantially alter the overall performance of the clustering algorithms. However, larger adjustments (e.g., setting $a$ to 0.01 or higher) resulted in noticeable degradation in clustering quality, highlighting the importance of keeping $a$ within a narrow range. This analysis supports our decision to fix $a$ at 0.0001, ensuring that the model remains robust while still allowing $b$ and $c$ the flexibility needed to adapt to specific dataset characteristics. The remaining columns represent the optimized parameters $b$ and $c$, which vary for each dataset. The parameter $b$ influences the model's responsiveness to changes in the data, while $c$ affects the overall scaling of the model's output. Finally, the "Best Smooth" column indicates the optimal smoothing value achieved during cross-validation, which reflects the model's performance in terms of clustering accuracy. The varying values in this column highlight how different datasets require tailored parameter settings to achieve the best results.

**Table 3.** Optimal parameters for transformations.

| Dataset | Log | | | Sq | | | HTan | | |
|---|---|---|---|---|---|---|---|---|---|
| | *b* | *c* | BS | *b* | *c* | BS | *b* | *c* | BS |
| ng3sim | 2.7809 | 9.1445 | 50.1700 | 2.7951 | 9.1290 | 7.0700 | 2.3249 | 9.1467 | 18.3700 |
| classic | 0.0682 | 1.0919 | 25.0000 | 0.0682 | 1.0919 | 15.0000 | 0.0682 | 1.0918 | 15.0000 |
| ohscal | 3.0457 | 16.4311 | 0.1700 | 2.4546 | 16.5128 | 25.0000 | 3.6363 | 16.3386 | 17.0300 |
| k1b | 2.2467 | 21.8494 | 16.7300 | 1.6804 | 22.9310 | 0.1000 | 2.7789 | 22.8170 | 0.8700 |
| hitech | 3.1803 | 23.7720 | 34.0000 | 3.1843 | 23.8005 | 8.5700 | 2.6590 | 23.7885 | 0.2000 |
| reviews | 3.0342 | 27.0628 | 66.6700 | 3.0326 | 27.0573 | 13.3700 | 3.0317 | 27.0648 | 83.3300 |
| sports | 2.5719 | 20.4878 | 8.5000 | 2.5922 | 20.4682 | 35.0000 | 3.0420 | 20.4616 | 66.6700 |
| la1 | 2.7167 | 17.7621 | 0.4000 | 3.2277 | 17.6894 | 46.6700 | 2.2066 | 17.8179 | 7.1700 |
| la12 | 2.2671 | 18.6777 | 16.7000 | 2.7673 | 18.6036 | 8.3700 | 2.7632 | 18.6087 | 41.6700 |
| la2 | 2.8408 | 17.7540 | 23.3400 | 3.3745 | 17.6619 | 1.7000 | 2.8439 | 17.7351 | 0.2300 |
| tr11 | 2.4381 | 5.7950 | 50.6700 | 1.2637 | 5.3720 | 40.0300 | 2.4959 | 6.1245 | 8.3700 |

**Continued**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| tr23 | 1.8971 | 3.3695 | 0.1000 | 2.2632 | 3.5174 | 0.1000 | 2.2537 | 3.5558 | 0.1000 |
| tr41 | 2.5543 | 16.7309 | 13.3700 | 2.2009 | 16.4067 | 0.7300 | 2.1465 | 15.7258 | 18.3700 |
| tr45 | 2.3656 | 9.2114 | 16.7300 | 2.3873 | 8.8119 | 0.1000 | 1.5483 | 8.2574 | 17.6700 |

Table 3 presents parameter values for three different nonlinear transformation: *logarithmic* transformation (Equation (7)), *square root* transformation (Equation (8)), and *hyperbolic tangent* transformation (Equation (9)). Even though these methods use different mathematical functions, the values for parameters $b$ and $c$ are very similar across all datasets. For the logarithmic transformation, $b$ usually ranges from 0.068 to 3.180, and $c$ ranges from 1.091 to 27.062. The square root transformation also shows similar patterns, with $b$ values in the same range and $c$ values that suggest these two transformations might work well with similar types of data. By analyzing the ranges of the parameter of the "Best Smooth", we observe interesting trends: for the *logarithmic transformation*, the values of $b$ range from 0.068 to 3.180, while those for $c$ span from 1.091 to 27.062. The "Best Smooth" values in this transformation vary from 0.10 to 66.67. For the *square root transformation*, the $b$ parameters remain within a similar range, from 0.068 to 3.374, and the values of $c$ range from 1.091 to 27.057. The "Best Smooth" values for this function is between 0.10 and 46.67. For the *hyperbolic tangent transformation*, the $b$ values also fall within the same interval, from 0.068 to 3.636, with $c$ values ranging from 1.091 to 27.064. The "Best Smooth" values in this approach vary from 0.10 to 83.33. These intervals demonstrate that, although the transformations employ different mathematical nonlinear functions, the parameters $b$ and $c$ exhibit significant similarities, suggesting that these transformations may be well-suited for similar types of data. The choice of a fixed starting value for $a$ allows for a clearer analysis of the impact of $b$ and $c$ on the results, minimizing the effects of variations in $a$.

**Performance Evaluation:**

We present the accuracy achieved by KL-ONMF on both the original and transformed data across different datasets. The accuracy metrics are denoted as follows: acc-KL for overall accuracy, BestAcKL-ONMF (Train) for the accuracy on the training dataset, and BestAcKL-ONMF (Validation) for the accuracy on the validation dataset. The distinction between "training" and "validation" phases is critical in our approach, particularly given the nature of the KL-ONMF method and the transformations applied to the data. During the training phase, the model focuses on learning complex patterns and relationships within the feature space that is generated from the transformed TF-IDF scores. This phase heavily relies on the training data, which plays a significant role in adjusting the model's parameters. Conversely, the validation phase serves an essential purpose by preventing the model from overfitting to the training data. Evaluating the model on a separate validation set allows us to gauge its ability to generalize to new, unseen data—an aspect that is vital for practical applications. In our methodology, where TF-IDF

scores are subjected to various transformations, validating performance on data that the model has not previously encountered becomes particularly important. This validation helps us assess the effectiveness of the smoothing and non-linear transformations applied, ensuring they genuinely enhance the model's clustering and classification capabilities rather than simply memorizing the training data. Thus, the clear separation of training and validation phases in our approach not only bolsters the reliability of the model's performance but also underscores its robustness in real-world scenarios.

**Table 4.** KL-ONMF vs. KL-ONMF Transformed for the clustering of 14 document data sets: Cross-Validation with Repetitions for Logarithm Transformation.

| Dataset | *m* | *n* | *r* | AcKL-ONMF | Best AcKL-ONMF (Train) | Best AcKL-ONMF (Validation) |
|---|---|---|---|---|---|---|
| ng3sim | 2998 | 15,810 | 3 | 71.45 | **74.61** | 74.12 |
| classic | 7094 | 41,681 | 4 | 85.38 | **91.99** | 85.49 |
| ohscal | 11,162 | 11,465 | 10 | 47.74 | **51.69** | 51.20 |
| k1b | 2340 | 21,819 | 6 | 58.46 | **60.94** | 60.53 |
| hitech | 2301 | 10,080 | 6 | **38.55** | 37.81 | 37.27 |
| reviews | 4069 | 18,483 | 5 | **72.35** | 72.12 | 71.84 |
| sports | 8580 | 14,870 | 7 | 66.55 | **67.19** | 65.45 |
| la1 | 3204 | 31,472 | 6 | 61.20 | **63.10** | 61.66 |
| la12 | 6279 | 31,472 | 6 | **67.48** | 63.10 | 61.75 |
| la2 | 3075 | 31,472 | 6 | 59.90 | **63.91** | 61.95 |
| tr11 | 414 | 6424 | 9 | **54.11** | 47.83 | 44.85 |
| tr23 | 204 | 5831 | 6 | 34.31 | **43.14** | 41.83 |
| tr41 | 878 | 7453 | 10 | 48.63 | **50.72** | 49.24 |
| tr45 | 690 | 8261 | 10 | 59.57 | **60.39** | 58.89 |
| Averages | | | | 58.98 | **60.61** | 59.01 |

Table 4 compares the performance of the AcKL-ONMF method applied to both untransformed data and smoothed, non-linear transformations. The metrics include baseline accuracy, Best AcKL-ONMF accuracy for training, and validation sets across various datasets.

Notably, the Best AcKL-ONMF(Train) values consistently outperform the baseline accuracy, indicating that the method effectively learns features from the transformed data. For instance, Dataset 1 shows a significant increase from 71.45% to 74.61%, demonstrating enhanced model performance through the application of smoothing and non-linear transformations.

When examining the Best AcKL-ONMF(Validation) results, there is a clear trend of improvements in accuracy on unseen data. Datasets such as Dataset 2 show a rise from 85.38% to 91.99% in training, with a validation accuracy of 85.49%, illustrating the model's ability to generalize effectively.

The average accuracies further validate our approach. The training set achieves an average of 60.61%, while the validation set maintains competitive performance at 59.01%. This consistency across both training and validation sets suggests that the AcKL-ONMF method not only optimizes the model parameters well but also ensures robust performance on unseen data.

**Table 5.** KL-ONMF vs. KL-ONMF Transformed for the clustering of 14 document data sets: Cross-Validation with Repetitions for Squared Transformation.

| Dataset | *m* | *n* | *r* | AcKL-ONMF | Best AcKL-ONMF (Train) | Best AcKL-ONMF (Validation) |
|---|---|---|---|---|---|---|
| ng3sim | 2998 | 15,810 | 3 | 71.45 | **73.85** | 72.98 |
| classic | 7094 | 41,681 | 4 | 85.38 | **92.92** | 92.46 |
| ohscal | 11,162 | 11,465 | 10 | 47.74 | **51.74** | 51.18 |
| k1b | 2340 | 21,819 | 6 | **58.46** | 56.17 | 55.98 |
| hitech | 2301 | 10,080 | 6 | **38.55** | 38.06 | 37.87 |
| reviews | 4069 | 18,483 | 5 | **72.35** | 69.14 | 68.15 |
| sports | 8580 | 14,870 | 7 | 66.55 | **68.56** | 68.24 |
| la1 | 3204 | 31,472 | 6 | 61.20 | **61.49** | 60.21 |
| la12 | 6279 | 31,472 | 6 | **67.48** | 60.05 | 57.47 |
| la2 | 3075 | 31,472 | 6 | 59.90 | **62.12** | 60.11 |
| tr11 | 414 | 6424 | 9 | **54.11** | 45.65 | 44.85 |
| tr23 | 204 | 5831 | 6 | 34.31 | **40.52** | 41.01 |
| tr41 | 878 | 7453 | 10 | 48.63 | **50.30** | 48.71 |
| tr45 | 690 | 8261 | 10 | 59.57 | **60.77** | 59.47 |
| Averages | | | | 58.98 | **59.38** | 58.48 |

Table 5 compares the performance of the AcKL-ONMF method applied to both untransformed data and smoothed, non-linear transformations. The metrics include baseline accuracy, Best AcKL-ONMF accuracy for training, and validation sets across various datasets.

Notably, the Best AcKL-ONMF(Train) values consistently outperform the baseline accuracy, indicating that the method effectively learns features from the transformed data. For instance, Dataset 1 shows a significant increase from 71.45% to 73.85%, demonstrating enhanced model performance through the application of smoothing and non-linear transformations.

When examining the Best AcKL-ONMF(Validation) results, there is a clear trend of improvements in accuracy on unseen data. Datasets such as Dataset 2 show a rise from 85.38% to 92.92% in training, with a validation accuracy of 92.46%, illustrating the model's ability to generalize effectively.

The average accuracies further validate our approach. The training set achieves an average of 59.38%, while the validation set maintains competitive performance at 58.48%. This consistency across both training and validation sets suggests that

the AcKL-ONMF method not only optimizes the model parameters well but also ensures robust performance on unseen data.

**Table 6.** KL-ONMF vs. KL-ONMF Transformed for the clustering of 14 document data sets: Cross-Validation with Repetitions for Hyperbolic Tangent Transformation.

| Dataset | *m* | *n* | *r* | AcKL-ONMF | Best AcKL-ONMF (Train) | Best AcKL-ONMF (Validation) |
|---|---|---|---|---|---|---|
| ng3sim | 2998 | 15,810 | 3 | 71.45 | **73.86** | 72.95 |
| classic | 7094 | 41,681 | 4 | 85.38 | **91.58** | 82.24 |
| ohscal | 11,162 | 11,465 | 10 | 47.74 | **51.57** | 51.15 |
| k1b | 2340 | 21,819 | 6 | 58.46 | **60.23** | 59.46 |
| hitech | 2301 | 10,080 | 6 | 38.55 | **38.78** | 38.17 |
| reviews | 4069 | 18,483 | 5 | **72.35** | 71.27 | 71.15 |
| sports | 8580 | 14,870 | 7 | 66.55 | **68.67** | 67.88 |
| la1 | 3204 | 31,472 | 6 | 61.20 | **66.46** | 63.77 |
| la12 | 6279 | 31,472 | 6 | **67.48** | 61.03 | 57.76 |
| la2 | 3075 | 31,472 | 6 | 59.90 | **63.12** | 62.35 |
| tr11 | 414 | 6424 | 9 | **54.11** | 50.32 | 46.62 |
| tr23 | 204 | 5831 | 6 | 34.31 | **45.10** | 39.54 |
| tr41 | 878 | 7453 | 10 | 48.63 | **51.78** | 49.77 |
| tr45 | 690 | 8261 | 10 | 59.57 | **65.75** | 64.06 |
| Averages | | | | 58.98 | **61.39** | 59.06 |

Table 6 compares the performance of the AcKL-ONMF method applied to both untransformed and smoothed, non-linear transformed data. The metrics include baseline accuracy, Best AcKL-ONMF accuracy for training, and validation sets across various datasets.

The Best AcKL-ONMF(Train) values consistently exceed the baseline accuracy, demonstrating the method's effectiveness in extracting features from the transformed data. For example, Dataset 1 shows an increase from 71.45% to 73.86%, indicating a substantial improvement in model performance due to the application of smoothing and non-linear transformations.

In terms of Best AcKL-ONMF(Validation), the results reveal a positive trend in accuracy for unseen data. Dataset 2 exemplifies this, with the accuracy rising from 85.38% to 91.58% in training, while validation accuracy stands at 82.24%. This illustrates the model's capacity to generalize effectively to new data.

Average accuracies further support our findings, with the training set achieving an average of 61.39%, compared to a baseline of 58.98%. The validation set maintains a competitive average of 59.06%, reinforcing the consistency of the AcKL-ONMF method across both training and validation contexts.

## 5. Conclusion and Suggestion

In this paper, we developed an innovative clustering approach for non-negative data, named *Optimal Predictive Modeling of Nonlinear Transformations using the Kullback-Leibler Divergence* "*OPMNT-KL-ONMF*". Our methodology begins by applying smoothing to TF-IDF scores, followed by the integration of parameters optimized using the Nelder-Mead method. We then employ nonlinear transformations to extract nonlinear features. Comparative results between KL-ONMF applied to original data and data transformed by OPMNT demonstrate a significant performance improvement. Our OPMNT-KL-ONMF model represents a promising advancement in the field of clustering non-negative data, such as document datasets, and paves the way for future research and applications across various domains of data analysis.

## Acknowledgements

Sincere thanks to the members of OJAppS for their exemplary professionalism, and a special acknowledgment to Managing Editor *Emily Lee* for her exceptional commitment to high-quality standards.

## Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

## References

[1] Paatero, P. and Tapper, U. (1994) Positive Matrix Factorization: A Non-Negative Factor Model with Optimal Utilization of Error Estimates of Data Values. *Environmetrics*, **5**, 111-126. https://doi.org/10.1002/env.3170050203

[2] You, M., Wang, H., Liu, Z., Chen, C., Liu, J., Xu, X., *et al.* (2017) Novel Feature Extraction Method for Cough Detection Using NMF. *IET Signal Processing*, **11**, 515-520. https://doi.org/10.1049/iet-spr.2016.0341

[3] Oti, E.U., Olusola, M.O., Eze, F.C. and Enogwe, S.U. (2021) Comprehensive Review of K-Means Clustering Algorithms. *International Journal of Advances in Scientific Research and Engineering*, **7**, 64-69. https://doi.org/10.31695/ijasre.2021.34050

[4] Saberi-Movahed, F., Berahman, K., Sheikhpour, R., Li, Y. and Pan, S. (2024) Nonnegative Matrix Factorization in Dimensionality Reduction: A Survey.

[5] Wang, Y., Jia, Y., Hu, C. and Turk, M. (2005) Non-Negative Matrix Factorization Framework for Face Recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, **19**, 495-511. https://doi.org/10.1142/s0218001405004198

[6] Mohammadiha, N., Smaragdis, P. and Leijon, A. (2013) Supervised and Unsupervised Speech Enhancement Using Nonnegative Matrix Factorization. *IEEE Transactions on Audio, Speech, and Language Processing*, **21**, 2140-2151. https://doi.org/10.1109/tasl.2013.2270369

[7] Wilson, K.W., Raj, B., Smaragdis, P. and Divakaran, A. (2008) Speech Denoising Using Nonnegative Matrix Factorization with Priors. 2008 *IEEE International Conference on Acoustics, Speech and Signal Processing*, Las Vegas, 31 March-4 April 2008, 4029-4032. https://doi.org/10.1109/icassp.2008.4518538

[8] Pauca, V.P., Piper, J. and Plemmons, R.J. (2006) Nonnegative Matrix Factorization for Spectral Data Analysis. *Linear Algebra and Its Applications*, **416**, 29-47. https://doi.org/10.1016/j.laa.2005.06.025

[9] Shiga, M. and Muto, S. (2019) Non-Negative Matrix Factorization and Its Extensions for Spectral Image Data Analysis. *e-Journal of Surface Science and Nanotechnology*, **17**, 148-154. https://doi.org/10.1380/ejssnt.2019.148

[10] Devarajan, K. (2008) Nonnegative Matrix Factorization: An Analytical and Interpretive Tool in Computational Biology. *PLOS Computational Biology*, **4**, e1000029. https://doi.org/10.1371/journal.pcbi.1000029

[11] Qi, Q., Zhao, Y., Li, M. and Simon, R. (2009) Non-Negative Matrix Factorization of Gene Expression Profiles: A Plug-In for Brb-Arraytools. *Bioinformatics*, **25**, 545-547. https://doi.org/10.1093/bioinformatics/btp009

[12] Ouyang, D., Miao, R., Zeng, J., Li, X., Ai, N., Wang, P., *et al.* (2024) Splhrnmtf: Robust Orthogonal Non-Negative Matrix Tri-Factorization with Self-Paced Learning and Dual Hypergraph Regularization for Predicting miRNA-Disease Associations. *BMC Genomics*, **25**, Article No. 885. https://doi.org/10.1186/s12864-024-10729-w

[13] Li, L., Gao, Z., Wang, Y.-T., Zhang, M.-W., Ni, J.-C., Zheng, C.-H. and Su, Y. (2021) SCMFDA: Predicting MicroRNA-Disease Associations Based on Similarity Constrained Matrix Factorization. *PLOS Computational Biology*, **17**, e1009165.

[14] Gligorijevic, V., Panagakis, Y. and Zafeiriou, S. (2018) Non-Negative Matrix Factorizations for Multiplex Network Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **41**, 928-940. https://doi.org/10.1109/tpami.2018.2821146

[15] Zhang, Z.-Y. (2012) Nonnegative Matrix Factorization: Models, Algorithms and Applications. In: *Intelligent Systems Reference Library*, Springer, 99-134. https://doi.org/10.1007/978-3-642-23241-1_6

[16] Jia, X. and Guo, B. (2022) Non-Negative Matrix Factorization Based on Smoothing and Sparse Constraints for Hyperspectral Unmixing. *Sensors*, **22**, Article 5417. https://doi.org/10.3390/s22145417

[17] Jiménez-Sánchez, D., Ariz, M., Morgado, J.M., Cortés-Domínguez, I. and Ortiz-de-Solórzano, C. (2020) NMF-RI: Blind Spectral Unmixing of Highly Mixed Multispectral Flow and Image Cytometry Data. *Bioinformatics*, **36**, 1590-1598. https://doi.org/10.1093/bioinformatics/btz751

[18] Pengo, T., Muñoz-Barrutia, A., Zudaire, I. and Ortiz-de-Solorzano, C. (2013) Efficient Blind Spectral Unmixing of Fluorescently Labeled Samples Using Multi-Layer Non-Negative Matrix Factorization. *PLOS ONE*, **8**, e78504. https://doi.org/10.1371/journal.pone.0078504

[19] Okada, S., Ohzeki, M. and Taguchi, S. (2019) Efficient Partition of Integer Optimization Problems with One-Hot Encoding. *Scientific Reports*, **9**, Article No. 13036. https://doi.org/10.1038/s41598-019-49539-6

[20] Zhang, Y., Gong, L. and Wang, Y. (2005) An Improved TF-IDF Approach for Text Classification. *Journal of Zhejiang University Science*, **6**, 49-55. https://doi.org/10.1631/jzus.2005.a49

[21] Bafna, P., Pramod, D. and Vaidya, A. (2016) Document Clustering: TF-IDF Approach. 2016 *International Conference on Electrical, Electronics, and Optimization Techniques* (*ICEEOT*), Chennai, 3-5 March 2016, 61-66. https://doi.org/10.1109/iceeot.2016.7754750

[22] Hien, L.T.K. and Gillis, N. (2021) Algorithms for Nonnegative Matrix Factorization with the Kullback-Leibler Divergence. *Journal of Scientific Computing*, **87**, Article 93.

[23] Berry, M.W., Gillis, N. and Glineur, F. (2009) Document Classification Using Nonnegative Matrix Factorization and Underapproximation. 2009 *IEEE International Symposium on Circuits and Systems*, Taipei City, 24-27 May 2009, 2782-2785. https://doi.org/10.1109/iscas.2009.5118379

[24] Hu, L., Wu, N. and Li, X. (2022) Feature Nonlinear Transformation Non-Negative Matrix Factorization with Kullback-Leibler Divergence. *Pattern Recognition*, **132**, Article 108906. https://doi.org/10.1016/j.patcog.2022.108906

[25] Nkurunziza, J.P., Nahayo, F. and Gillis, N. (2024) Orthogonal Nonnegative Matrix Factorization with the Kullback–Leibler Divergence. *Pattern Recognition Letters*, **197**, 353-358. https://doi.org/10.1016/j.patrec.2025.08.012

[26] Hiemstra, D. (2000) A Probabilistic Justification for Using TF×IDF Term Weighting in Information Retrieval. *International Journal on Digital Libraries*, **3**, 131-139.

[27] Losada, D.E. and Azzopardi, L. (2008) An Analysis on Document Length Retrieval Trends in Language Modeling Smoothing. *Information Retrieval*, **11**, 109-138. https://doi.org/10.1007/s10791-007-9040-x.

[28] Hazem, A. and Morin, E. (2013) A Comparison of Smoothing Techniques for Bilingual Lexicon Extraction from Comparable Corpora. *Proceedings of the Sixth Workshop on Building and Using Comparable Corpora*, Sofia, August 2013, 24-33.

[29] Katz, S. (1987) Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **35**, 400-401. https://doi.org/10.1109/tassp.1987.1165125

[30] Jayaweera, M.P. and Dias, N.G.J. (2015) Application of Witten-Bell Discounting Techniques for Smoothing in Part of Speech Tagging Algorithm for Sinhala Language. *Proceedings of the International Postgraduate Research Conference*, Sri Lanka, 10 June 2015, p. 167.

[31] Qaiser, S. and Ali, R. (2018) Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. *International Journal of Computer Applications*, **181**, 25-29. https://doi.org/10.5120/ijca2018917395

[32] Manning, C.D., Raghavan, P. and Schütze, H. (2008) Term Weighting, and the Vector Space Model. In: *Introduction to Information Retrieval*, Cambridge University Press, 109-133.

[33] Kikuchi, M., Yoshida, M., Okabe, M. and Umemura, K. (2015) Confidence Interval of Probability Estimator of Laplace Smoothing. 2015 2*nd International Conference on Advanced Informatics: Concepts, Theory and Applications* (*ICAICTA*), Chonburi, 19-22 August 2015, 1-6. https://doi.org/10.1109/icaicta.2015.7335387

[34] Van Laarhoven, T. (2017) L2 Regularization versus Batch and Weight Normalization.

[35] Gillis, N. (2020) Nonnegative Matrix Factorization. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611976410

[36] Gillis, N. (2014) Successive Nonnegative Projection Algorithm for Robust Nonnegative Blind Source Separation. *SIAM Journal on Imaging Sciences*, **7**, 1420-1450. https://doi.org/10.1137/130946782

[37] Singer, S. and Nelder, J. (2009) Nelder-Mead Algorithm. *Scholarpedia*, **4**, Article 2928. https://doi.org/10.4249/scholarpedia.2928

[38] Zhong, S. and Ghosh, J. (2005) Generative Model-Based Document Clustering: A Comparative Study. *Knowledge and Information Systems*, **8**, 374-384. https://doi.org/10.1007/s10115-004-0194-1