Issues in Information Systems
Volume 26, Issue 4, pp. 39-44, 2025

DOI: https://doi.org/10.48009/4_iis_2025_104

Regularization of document-term matrices using singular value decomposition

Ben Kim, *Seattle University*, ***bkim@seattleu.edu***

Abstract

This paper seeks to identify appropriate regularization methods for document-term matrices. Regularization is essential in machine learning for reducing overfitting and improving model generalization. The evaluation uses a document-term matrix generated from a vectorized wine review dataset, with cross-validation performed using random forest and gradient boosting regression algorithms. Using these evaluation scores, we identify the proper regularization methods for document-term matrices. All implementations are written in Python, and the source code is provided to ensure reproducibility. While L1 (lasso) and L2 (ridge or Tikhonov) regularization are widely used, this paper investigates Singular Value Decomposition (SVD) as an alternative approach, particularly suited for high-dimensional and noisy datasets. Three SVD-based regularization techniques are explored for document-term matrices in the context of natural language processing: low-rank approximation, feature orthogonalization, and principal component analysis. Each method is implemented and assessed based on execution time, memory consumption, and predictive performance measured by R^2 scores.

Keywords: egularization, document term matrix, singular value decomposition, low-rank approximation, feature orthogonalization, principal component analysis

Introduction

The paper seeks to identify appropriate regularization methods for document-term matrices. Regularization is a fundamental concept in machine learning, helping models avoid overfitting by discouraging unnecessary complexity. Traditional approaches, such as L1 (lasso) and L2 (ridge) penalties, directly constrain model parameters through modifications to the loss function. However, for high-dimensional, sparse, or noisy datasets, regularization at the data level can also be effective. In text mining, for instance, document-term matrices (DTMs) often exhibit high dimensionality and sparsity, making them suitable candidates for matrix factorization methods that reveal latent structure and reduce noise. In this paper, we seek to identify appropriate regularization methods for document-term matrices.

Singular Value Decomposition (SVD) has long been used in applications such as Latent Semantic Analysis to uncover hidden relationships among terms and documents. Building on this, our study examines how SVD-based techniques can serve as a practical form of data regularization in natural language processing tasks. We focus on three approaches that leverage SVD to address redundancy and improve computational efficiency in DTMs. Using a real-world wine review dataset, we evaluate the impact of these methods on model performance and resource usage, highlighting practical considerations for their implementation in Python.

Literature Review

Deerwester et al. (1990) used SVD to enhance the detection of relevant documents based on query terms. They decomposed a large document-term matrix into approximately 100 orthogonal factors, allowing the original matrix to be approximated through their linear combination. Berry et al. (1995) used SVD to reveal the implicit higher-order structure within document-term matrices. By representing terms and documents using the 200–300 largest singular vectors, they matched these representations to user queries. This retrieval approach, known as Latent Semantic Indexing (LSI), captures important associative relationships between terms and documents that are not apparent when analyzing individual documents alone.

Huang, Shen, and Buja (2013) propose a sparse-smooth regularized SVD that combines sparsity and smoothness constraints to improve interpretability and denoising in high-dimensional data. Their method outperforms standard SVD by selecting relevant features and preserving structural continuity, which is especially useful in applications like image and signal processing. Kawano (2021) discusses sparse regularization with PCA and regression using SVD. The approach aims to obtain principal component loadings that incorporate information from both explanatory variables and a response variable, enhancing the interpretability and predictive performance of the model.

Kumar and Schneider (2017) examine multiple algorithms for low-rank approximation, including Singular Value Decomposition (SVD) and QR decomposition, highlighting the computational challenges associated with high-dimensional data. Hamlomo et al. (2025) present a systematic review of low-rank and local low-rank matrix approximation methods for big data medical imaging. Their work highlights how these techniques address high dimensionality and noise in applications such as MRI reconstruction and image denoising. The review contrasts global low-rank models with newer local approaches that better capture structural details by exploiting spatial redundancies. Overall, the authors emphasize the growing importance of low-rank methods for developing efficient and accurate medical imaging algorithms.

SVD (Singular Value Decomposition)

Singular Value Decomposition is a matrix factorization technique that decomposes any matrix, square or rectangular, into three component matrices. Unlike eigendecomposition, which is limited to square matrices, SVD can be applied to matrices of any shape. The two techniques for regularization in this paper, low-rank approximation and feature orthogonalization, are based on truncated SVD. Given a matrix A, SVD factorizes it as the product of three matrices as in the following:

$A = U \Sigma V^{-1}$

where V^{-1} is the inverse of V. U and V are orthonormal matrices.
Since V is an orthonormal matrix, $V^{-1} = V^T$ where V^T is the transpose of V.

Thus $\underline{A = U \Sigma V^T}$

where
$U \in R^{m \times m}$: orthonormal matrix of left singular vectors
$\Sigma \in R^{m \times n}$: diagonal matrix of singular values
$V \in R^{n \times n}$: orthonormal matrix of right singular vectors

The singular values in $\sum$ are arranged in descending order. Each singular value represents the significance in each corresponding column or row of the original matrix. Left singular vectors (columns of U) span the column space of A. Right singular vectors (columns of V) span the row space A.

Dataset description for testing

A dataset was obtained from Kaggle (https://www.kaggle.com/datasets/zynicide/wine-reviews) titled "winemag-data-130k-v2.csv," containing 129,971 rows and 14 columns, with a file size of 52.91 megabytes. The dataset includes columns such as 'country', 'description', 'designation', 'points', 'price', 'province', 'region1', 'region2', 'taster_name', 'taster_twitter_handle', 'title', 'variety', and 'winery'. The 'points' column represents the wine scores assigned by Wine Enthusiast magazine on a scale of 1 to 100 (Kaggle, 2022).

The 'price' and 'points' columns contain numerical data, while 'description' consists of text data. The remaining columns are nominal or categorical. The wines originate from 43 countries, including the USA, France, Italy, and Spain. There are 571 grape varieties, such as Pinot Noir, Chardonnay, and Cabernet Sauvignon. Examples of wine descriptions are presented in Table 1.

Table 1. Examples of Wine Descriptions

Description
"Aromas include tropical fruit, broom, brimstone and dried herb. The palate isn't overly expressive, offering unripened apple, citrus and dried sage alongside brisk acidity."
"This is ripe and fruity, a wine that is smooth while still structured. Firm tannins are filled out with juicy red berry fruits and freshened with acidity. It's already drinkable, although it will certainly be better from 2016."
"Tart and snappy, the flavors of lime flesh and rind dominate. Some green pineapple pokes through, with crisp acidity underscoring the flavors. The wine was all stainless-steel fermented."
"Pineapple rind, lemon pith and orange blossom start off the aromas. The palate is a bit more opulent, with notes of honey-drizzled guava and mango giving way to a slightly astringent, semidry finish."

The 'points' column represents the ratings assigned by Wine Enthusiast magazine on a scale of 1 to 100, with scores ranging from 80 to 100. The average and median scores are 88.44 and 88, respectively. The 'price' column ranges from $4 to $3,300, with a mean of $35.36 and a median of $35. The dataset includes 12 wine tasters, including Roger Voss, Michael Schachner, and others. Further details about the dataset are available in Kim (2022).

Reduction of sample size

The original dataset comprises 129,971 samples. Due to computational constraints, we created a more manageable subset by applying filtering criteria, such as selecting wines from countries with at least 1,000 entries and focusing on popular wine varieties. This reduced the dataset to 74,999 samples. To further address computational limitations, we randomly selected 1,000 samples to generate a document-term matrix (DTM) using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer for wine descriptions, as shown in Table 1. The resulting DTM has a shape of 1,000 × 3,856, representing 3,856 unique terms. Subsequently, we applied three data regularization techniques and evaluated two machine learning algorithms—Random Forest and Gradient Boosting. The resulting R^2 scores are presented later in the performance and discussion section.

Dataset Regularization

Low-rank approximation (LRA)

LRA is based on the truncated SVD. This technique approximates a high-dimensional matrix by a matrix of lower rank k, where k is lower than the original matrix's rank. By controlling the number of singular values (via a parameter k), we ensure that only the dataset's most important dimensions (or features) are

retained. This technique can help reduce dimensionality, eliminate noise, and capture the most significant patterns in the data. It may enhance model performance and generalization.
We can compress the dataset by multiplying the top *k* left singular vectors (U_k) by a diagonal matrix containing the top *k* singular values (Σ_k) as in the following:

$$A_k = U_k \Sigma_k V_k^t \text{ where } A_k \in R^{m \times n}$$

The reconstructed matrix A_k retains the same shape as the original matrix but with a reduced rank *k*. This compression technique is useful for applications such as image processing, denoising, and imputing missing values. We can reduce the dimensionality of the matrix A by multiplying the first *k* left singular vectors by the diagonal matrix with the top *k* singular values, as in the following:

$$A_r = U_k \Sigma_k \text{ where } A_r \in R^{m \times k}$$

This process is equivalent to feature orthogonalization using SVD, which will be explained in the next section. LRA was implemented on our dataset using sklearn.decomposition.TruncatedSVD, as shown below:

```
from sklearn.decomposition import TruncatedSVD
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import cross_val_score

k = 100  # Number of singular values
svd = TruncatedSVD(n_components=k, random_state=1234)
X_reduced = svd.fit_transform(X)
print(f'Memory size of X_ reduced is {X_ reduced.nbytes/2**20: .2f} megabytes.')
rfmr = RandomForestRegressor(random_state = 5678)
rfmr_mean_score = np.mean(cross_val_score(rfmr,X_ reduced,y,cv=5))
gbmr = GradientBoostingRegressor(random_state = 5678)
gbmr_mean_score = np.mean(cross_val_score(gbmr,X_ reduced,y,cv=5))
```

Feature orthogonalization
This technique converts a set of features into orthogonal ones, effectively eliminating multicollinearity and redundancy in large datasets. Unlike PCA, which transforms the data into linear combinations of principal components, feature orthogonalization preserves the original feature space, keeping each feature distinct and interpretable. Additionally, data centering is not required for feature orthogonalization. Full orthogonalization does not truncate any features. However, in this paper, we remove the less significant features (those associated with low singular values) to improve computational efficiency. Using SVD, orthogonalized features can be computed by multiplying the first *k* left singular vectors by the diagonal matrix with the top *k* singular values, as in the following:

$$A_o = U_k \Sigma_k \text{ where } A_o \in R^{m \times k}$$

This can implement this mathematical equation in two ways: first, by manually decomposing the matrix using SVD, and second, by utilizing the *TruncatedSVD* function from sklearn.*decomposition*, as demonstrated below.

```
# First way
U, S, VT = np.linalg.svd(X, full_matrices=False)
```

```
# Construct orthogonalized features
X_ortho = U @ np.diag(S)[:,:100]
print(f'Memory size of X_ortho is {X_ortho.nbytes/2**20:.2f} megabytes.')

# Alternative way
from sklearn.decomposition import TruncatedSVD
k = 100
svd = TruncatedSVD(n_components=k)
A_reduced = svd.fit_transform(X)  # Shape: (500, 100)
```

PCA (Principal Component Analysis)

As noted earlier, this paper examines regularization methods in the context of document-term matrices (DTMs). PCA requires data standardization, but since DTMs are typically sparse—composed largely of zeros—standardization transforms many of these zeros into non-zero values. This increases memory usage and computational complexity while reducing interpretability. In a DTM, zeros signify the absence of specific terms in a document; altering them distorts the semantic structure and compromises the integrity of the data.

Another limitation of PCA is that its principal components are generated as linear combinations of the original features, making the transformed components unintuitive and difficult to interpret. In contrast, feature orthogonalization preserves the original meanings of the features, allowing the transformed data to be directly understood. While PCA is widely used for dimensionality reduction and regularization, we do not believe that PCA is an appropriate regularization technique for document-term matrices.

Performances and Discussion

As discussed in the previous section, $U_k\Sigma_k$ represents the dataset where each orthogonalized feature (a column of U_k) is scaled by its corresponding singular value in Σ_k. This effectively reweights the features based on their importance, capturing the structure of the original dataset in a reduced-dimensional space. This dataset is a common basis for both low-rank approximation and feature orthogonalization when SVD is used for both techniques.

Table 2. Performances for each dataset

Predictor Features Points ~ Description	Size of dataset in megabytes	Algorithms	R-Squared	Time in seconds
Original dataset with 3744 features	29.42	Gradient Boosting	0.30	7.35
		Random Forest	0.30	49.42
$U_k\Sigma_k$ for 100 features using feature orthogonalization	0.76	Gradient Boosting	0.29	14.81
		Random Forest	0.25	25.57

To assess the contribution of the wine description alone, we executed the machine learning models using only the vectorized text data('description' in the dataset), excluding the other 12 features provided in the dataset (Kaggle, 2022). The number of features was reduced from 3,856 to 100, as shown in Table 2 — a 97% reduction. Consequently, the dataset size decreased from 29.42 megabytes to 0.76 megabytes. Despite the reduction, the R^2 scores declined only slightly, from 0.30 to 0.29 for gradient boosting and from 0.30 to 0.25 for random forest. The execution times decreased significantly, from 49.42 to 25.57 seconds for random forest.

Conclusion

Document-term matrices typically contain an extensive number of features in a sparse matrix, making the application of machine learning algorithms computationally intensive. This complexity can render execution infeasible in environments with limited computing resources. In this paper, we explore the use of feature orthogonalization to reduce the number of features while maintaining model performance. However, given the variability in the structure and content of document-term matrices, the effectiveness of this technique may not be universally applicable.

References

Berry, M. W., Dumais, S. T., & O'Brien, G. W. (1995). Using linear algebra for intelligent information retrieval. *SIAM review*, *37*(4), 573-595.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*(6), 391–407.

Hamlomo, S., Atemkeng, M., Brima, Y., Nunhokee, C., & Baxter, J. (2025). A systematic review of low-rank and local low-rank matrix approximation in big data medical imaging. *Neural Computing and Applications*, 1-56.

Hong, Z., & Lian, H. (2013). Sparse-smooth regularized singular value decomposition. *Journal of Multivariate Analysis*, *117*, 163-174.

Huang, J. Z., Shen, H., & Buja, A. (2013). Sparse-smooth regularized singular value decomposition. *Journal of Multivariate Analysis, 122*, 1–17. https://doi.org/10.1016/j.jmva.2013.06.007

Kaggle. (2022, May 21). *Wine Reviews*. https://www.kaggle.com/datasets/zynicide/wine-reviews

Kawano, S. (2021). Sparse principal component regression via singular value decomposition approach. *Advances in Data Analysis and Classification, 15*(4), 795–823. https://doi.org/10.1007/s11634-020-00435-2

Kim, B. (2022). Prediction of wine ratings using natural language processing. *Issues in Information Systems, 23*(1).

Kumar, N. K., & Schneider, J. (2017). Literature survey on low rank approximation of matrices. *Linear and Multilinear Algebra*, *65*(11), 2212-2244.