**A study of non-linear feature extraction in spike sorting**

Eugen-Richard Ardelean1, Raluca Laura Portase1

1Department of Computer Science, Technical University of Cluj-Napoca, Cluj-Napoca, Romania

**\* Corresponding authors:** [ardeleaneugenrichard@gmail.com](mailto:ardeleaneugenrichard@gmail.com), [placehoder-raluca]

**ORCID Author IDs:**

Eugen-Richard Ardelean: 0000-0002-0098-4228

Raluca Laura Portase: 0000-0002-8985-4728

**Abstract: [**TBWritten]

**Keywords:** clustering, [TBAdded]

# Introduction

## Spike Sorting

Extracellular recordings capture the neural activity as voltage fluctuations from multiple nearby neurons (1), producing a continuous signal. Each individual activity of a single neuron is called a spike and in the case of extracellular recordings the neuron that generated such an activity is unknown. Spike sorting is the process of assigning each detected spike waveform (2) from an extracellular recording to its source neuron, based on the assumption that each neuron produces spikes of consistent shape (3), while different neurons generate distinguishable shapes from each other (2). However, the shape of spikes can be distorted by noise, electrode drift (4) and biological variability which result in scattered clusters instead of pinpoints. Consequently, it is important to find a robust approach to reduce the impact of these phenomena. One approach is to extract a small set of discriminative features that preserve the information that differentiates between the shapes of spikes to improve clustering.

The canonical approach to spike sorting (2) is a pipeline of four sequantial steps: filtering, spike detection and clustering. Filtering is done in a band-pass manner and is applied to the raw recorded signal to isolate the 300 and 3000Hz (5) frequency band where spiking occurs. Spikes are detected typically through a simple amplitude thresholding based on the standard deviation of the signal multiplied by a scalar value (typically, from 3 to 5) to identify the events that could represent the neuronal activity. The choice of this threshold is a compromise between identifying noise segments as spikes and losing spikes of lower amplitude. Feature extraction is utilised to both generate a more auspicious space and to improve efficiency for clustering. The newly generated feature space should be robust as in invariant to small changes in shape such that it provides an easily separable space for clustering while simultaneously it should reduce the dimensionality to improve the execution time of clustering. As the final step of the spike sorting pipeline, the spikes should be grouped to represent all instances of activity of each individual neuron. This final assignment of spikes by source neurons is done by the clustering algorithm.

The spike sorting pipeline has seen many forms over the years. Initially, a supervised manual approach was taken where an expert would classify spikes based on a visual interpretation of similarity in a low dimensional space. This reduced space was generated by simple features (6) (7) such as amplitude, width, the peak-to-trough ratio. Using the peak-to-trough ratio feature was found to be useful in determining the type of neuron as narrow spikes (small peak-to-trough ratio) are representative of inhibitory neurons, while excitatory neurons have wider spikes (8). Since the 1950s, the number of recorded neurons has increased exponentially (9) rendering manual approaches unfeasible and recent developments in recording hardware (4,10) are following this trend. Through empirical analysis, probabilistics models were created that were able to leverage the entire spike waveform (11) allowing for the processing of a low number of electrodes. Later, the high-dimensional space of the spike waveform was projected to lower-dimensional spaces by applying PCA (12) and time-frequency transforms (such as the Wavelet Transform (13)) have started being used to introduce the frequency information in the computation.

The choice of approach is also depedent upon the nature of the analysis, offline or online. Offline spike sorting allows for the use of more complex algorithms as there is no time constraint as the analysis is done after the recording has finished. However, in an online setting, the algorithms employed must have to ability to process the data during the recording, thus requiring low execution times.

A common approach taken lately is template matching (14) on subsets of data, usually this approach substitutes the steps of spike detetion, feature extraction and even clustering in the canonical spike sorting pipeline. Due to it being applied on subsets rather than the whole data, it is an efficient approach from a computational perspective. One such algorithm that employs template matching is M-Sorter (15). M-Sorter is an automated approach to spike detection and classification based on coefficients obtained through the wavelet transform and template matching. This method can be seen as a two-stage process when applied to the filtered signal. It employs correlation of the wavelet coefficients for the detection of the spikes, while through the use of K-Means templates are generated which are used in the assignment of spikes to the neurons that produced them (considered to be that which has the smallest distance).

In this work, we attempt to examine the impact of feature extraction in spike sorting. Although, it is the clustering that outputs the final result and separation of the space into clusters, it is actually the feature extraction which must obtain a separable space for the clustering. Similarly to clustering algorithms, a golden standard (16,17) does not exist for feature extraction algorithms (2,5) either. Their performance depends on the particular set of characteristics of the input data. Here, we employ a number of non-linear feature extraction algorithms in the pursuit to identify the most adequate algorithm for the spike sorting problem.

## Non-linear feature extraction

The classical techniques for dimensinality reduction, such as PCA and MDS, are computationally eficient and perfectly able to find the structure of linear spaces (18). However, they encounter difficulties when non-linear structures are present (19). Non-linear manifold learning algorithms seek to discover a low-dimensional embedding (or a manifold) within the high-dimensional input data. These methods are capable of preserving the intristic geometry (including local neighbourshood and data topology) through their approximation of the underlying manifold, rather than relying on global linear projections such as PCA (20) (21).

In spike sorting, each detected spike waveform can be viewed as a high-dimensional vector. We can consider that the shapes of spikes vary from their ‘true shape’ due to recording artifacts. Therefore, non-linear manifold feature extraction techniques may be able to disentangle these factors by yielding embeddings that are robust to perturbations (22) and offer separability in overlapping clusters (generated by linear techniques). Moreover, modern manifold techniques have been designed to handle large volumes of data by employing sparse neighbourhoods graps and optimisation for scalability (23,24). This makes them a viable candidate for the spike sorting of high-density probes.

## The challenges of spike sorting [TBRewritten]

Spike sorting is inherently complex for several reasons. Brain recordings are inherently subject to the distortion of the spike waveforms due to the reasons specified above, these phenomena affecting spike shape generate clusters that do not have a well-defined separation boundary. This overlap of clusters is a struggle for most clustering algorithms, especially if coupled with data imbalance. Cluster imbalance in neuronal data appears from the variability in firing rate of neurons. Neuronal activity is dynamically modulated by neural circuits, causing individual cells to fire at widely varying rates (25) (26); this variability generates clusters of disparate sizes and yields an intrinsic imbalance in the dataset. Moreover, neuronal activity takes place on a millisecond timescale, thus even relatively short brain recording sessions can produce a vast quantity of data (3). In this context, single-unit activity refers to the spikes of one neuron that can be isolated as a single cluster, whereas spikes from more distant neurons typically appear with lower amplitudes (poor signal-to-noise ratio), and cannot be reliably separated (resulting in a single cluster being identified)—these are generally referred to as multiunit activity (5).

These non-linear manifold feature extraction methods often outperform linear feature spaces (27) and may be able to simultaneously denoise waveforms which can create dense clusters and increase the variability between the spikes of different neurons which can create separable clusters. In this study, we therefore evaluate a suite of representative non-linear feature extractors (e.g., Isomap, LLE, Spectral Embedding, Diffusion Maps, UMAP, TriMap) in comparison with traditional feature extraction methods and other non-linear feature extracton methods, to systematically compare how each manifold embedding influences cluster separability and spike-sorting performance across datasets.

The paper is structured as follows. Section 2 reviews traditional feature‐extraction techniques and their performance in spike sorting, outlines the methods proposed for spike sorting, and describes the datasets and evaluation metrics. In Section 3, we assess the methods across multiple metrics and offer a critical interpretation of their performance. Finally, Section 4 examines the limitations of the methods proposed for spike sorting and presents our concluding findings.

# Materials and Methods

## Feature extraction algorithms [TBRewritten]

The most important step of the spike sorting pipeline is the feature extraction where the high-dimensional space of the spike waveform is projected to a usually lower-dimensional space which contains the most informative features. The purpose is thus dual, to preserve as much as possible from the data structure of the original feature space in the reduced space and at the same time to reduce the space as much as possible. There are many criteria by which feature extraction methods may be categorized, such as convexity or linearity (28). Here, we separate the methods used into 3 categories: linear, non-linear, and non-linear manifold feature extraction methods.

### Linear feature extraction methods

The most common algorithm for feature extraction is the Principal Component Analysis (PCA) (20) and it has been thoroughly used in spike sorting (21) (5) as well. Even recently developed spike sorting pipelines employ PCA in their computations (29). PCA identifies orthogonal directions, or eigenvectors, based on maximum variance. PCA projects the original feature space into a new feature space, called principal components, based on the eigenvectors obtained through the eigendecomposition. Essentially, PCA rotates the coordinate system to align with maximum variance. Dimensionality reduction can be achieved by discarding components while preserving data variance, most commonly only the first two or three principal components represent 70-80% of the variance of the original feature space and only these are kept (30) (31). However, variance may not be the best approach for the separability of clusters (2) (5) as the discarded low-variance features may encode more information for separability.

Multidimensional Scaling (MDS) (32) creates low-dimensional representations that attempt to preserve the relationship between data points. Its classical version computes a distance matrix between all points to find coordinates in a lower-dimensional space that best match the original distances by minimizing a stress function. By preserving original distances, it can be considered a linear approach. For the Euclidean distance, MDS produces similar results to PCA.

Independent Component Analysis (ICA) (33) was designed to separate multivariate signals into independent components. Nevertheless, it was shown to be highly performant in the spike sorting domain as well (34) (35). In constrast to PCA which finds uncorrelated components, ICA seeks statistically independent sources by iteratively maximizing non-Gaussianity (using measures like kurtosis or negentropy). Thus, ICA works under the assumption that the signals are linear mixtures of non-Gaussian independent signals. The ICA algorithm effectively unmixes the signals by finding an unmixing matrix that produces the most statistically independent outputs.

### Non-linear feature extraction methods

Kernel PCA (KPCA) (36) is a non-linear extension of PCA through the use of the “kernel trick”. A non-linear kernel is utilized to map the input data into a possibly-higher dimensional feature space followed by PCA. Through the computation and the extraction of the eigenvectors of the kernel matrix (representing the inner product space), KPCA can capture non-linear relationship that PCA misses without the additional computation of coordinates in the higher-dimensional space.

A non-metric version of Multidimensional Scaling (MDS) (32) can preserve the ordering of distances rather than the values themselves. In other words, points that are closer than other in the original space are closer as well in the embeddings obtained. This is ahieved through the transformation of original space using a monotonic function and iteratively minimizing the same stress function. Through the ordering of distances, the non-metric MDS may be able to preserve the structure of data points when the relationship between similarity and distance is non-linear.

Self-Organizing Map (SOM) (37) creates a mapping between the data points and a two-dimensional grid where similar high-dimensional inputs are located nearby to each other. The grid of “neurons” is initialized in the low-dimensional space, followed by a training process where the input data is presented repeatedly to update the neuron (and the neighborhood) that best matches the input. The competitive learning process of SOMs preserves the topology of the input data.

Autoencoders (AE) (38) (39) are a type of neural network that are able to learn embeddings on the input data through an unsupervised approach. They are formed out of two sub-models, an encoder and a decoder. The encoder maps the input data to a latent embedding, while the decoder attempts to reconstruct the input data at the output. Through the optimization of the reconstruction, the autoencoder manages to a relevant low-dimensional representation of the input.

### Non-linear manifold feature extraction methods

Locally Linear Embedding (LLE) (40) preserves the local structure of data points by representing each as a weighted combination of its neighbors. It operates on the assumption that each neighborhood of points lies close on a locally linear patch of the manifold. The three steps of LLE are: identifying the k-nearest neighbors of each points, computing the weights which best reconstruct each point based on its neighbors (by solving linear system of equations) and finding a low-dimensional representation that preserve the reconstruction weights (by solving an eigenvalue problem).

Modified Locally Linear Embedding (MLLE) (41) is an extension of LLE which employs multiple weight vectors for each data point that obtain valid reconstructions. By employing alignment techniques, it allows for the identification of a global embedding that respects the constraints of each set of weight vectors.

Hessian-based Locally Linear Embedding (HLLE) (42) is another extension of LLE which use the Hessian operator to capture the local structure of the data. The Hessian matrix (representing the second derivative of the manifold) is computed for each neighborhood of nearest neighbors. HLLE identifies directions along which the manifold is locally flat by finding the null space of the Hessian. These directions form the basis for the low-dimensional embedding.

Local Tangent Space Alignment (LTSA) (43) is another extension of LLE which aligns local tangent spaces to caputre the global structure of the data. The tangent space is computed (as a linear approximation using principal components) for neighborhoods as the k-nearest neighbors. The embedding is found by aligning these tangent spaces (by solving an eigenvalue problem).

KepplerMapper (44) creates a graph representation of the data by leveraging concepts from topological data analysis to preserve the shape and connectivity of the original data. It projects the data onto a filter function, such as a principal component. The projection if then divided into overlapping interval. These intervals are clustered and it creates a graph where nodes represent clusters and edges represent overlap between clusters.

Isometric Mapping, or Isomap (19), attempts to maintain the geodesic distance between the data. Essentially, it flattens the manifold structure while preserving the geodesic distance. It constructs a graph connecting nodes to nearest neighbors. A distance matrix is computed by computing the shortest paths between pairs of points/nodes. It concludes by applying MDS to obtain the low-dimensional space.

T-distributed Stochastic Neighbor Embedding (t-SNE) (45) manages to create a lower-dimensional space by mapping high-dimensional data to lower dimensions through pairwise probability similarities while preserving both local and global structure. Gaussian distributions are used to compute the conditional probabilities that represent the similarities between the points in the original space. The t-distribution is used to computed the similarity probability distribution of the low-dimensional space. T-SNE minimizes the Kullback-Leibler divergence between input feature space and the reduced feature space by using the two distributions.

Spectral embedding (22) constructs a weighted graph representing the data and uses its Laplacian matrix for dimensionality reduction to preserve local structures. The graph connects each points to its nearby points, computes the graph Laplacian matrix, and finds its eigenvectors corresponding to the smallest non-zero eigenvalues. These eigenvectors form the low-dimensional embedding.

Diffusion Map (46) uses diffusion processes on the manifold to capture the intrinsic structure of the data. A graph is constructed where edges represent the probability of transitioning between points in a random walk. The eigenvectors of the normalized graph Laplacian are computed, which correspond to different time scales of the diffusion process. These eigenvectors are the low-dimensional embedding which were created by preserving diffusion distances.

PHATE (47) models diffusion processes through heat kernels to create an embedding that captures the intrinsic structure of the data. A neighbourhood graph is constructed and local affinities are computed. Diffusion is applied to capture multi-scale relationships, potential distances that preserve both local and global structure are computed and embedded into low dimensions using MDS.

UMAP (24) maps the high-dimensional data into low-dimensional embeddings that have a similar topological structure. A weighted graph is constructed with edges representing k-nearest neighbours. The edge weights are assigned using a fuzzy set membership function based on distance, thus creating a fuzzy topological representation. A low-dimensional space is initialized (usually using spectral techniques) and a similar graph is created (with different edge weights). Finally, UMAP optimizes the low-dimensional space to minimize the cross-entropy between the two graphs. Recently, UMAP has been applied to spike sorting with promising results (48,49).

TriMap (23) creates embeddings based on triplet constraints. These triplet constraints compare relative proximities between points. TriMap samples triplets of points *(i, j, k)* where *i* should be closer to *j* than to *k* in the embedding space. It then optimizes an objective function that minimizes violations of these constraints using gradient descent. By focusing on these relative proximity relations rather than absolute distances, TriMap efficiently captures both local and global structure.

## Clustering algorithms

External evaluation metrics require both the predicted cluster labels and the corresponding ground truth labels. As such, we employed K-Means (50) clustering to obtain the labels necessary for external metrics, applying it immediately after feature extraction. K-Means has long been used in clustering, with many adaptations developed over time. It was first utilized for spike sorting in 1988 (51) (52) and has remained the standard method for many years. Even recent spike sorting pipelines (14) (53) either rely on or are inspired by K-Means, and in a recent comparative study of 25 clustering methods, it demonstrated its continued competitive performance by ranking third (51).

K-Means (50) is a centroid-based clustering technique that achieves clustering by dividing the data space into *k* groups and allocating each point to the closest centroid according to Euclidean distance. But it has a number of drawbacks. First of all, it necessitates pre-specifying the number of clusters, which can be difficult for the real-world data. Nevertheless, there are preprocessing methods for finding the *k* parameter. Second, the method is non-deterministic in its most basic version, which means that different outcomes may be obtained from repeated executions. This problem has been resolved by more recent improvements that have increased its consistency. Thirdly, overlapping clusters are hard for K-Means to handle. However, this disadvantage is benefic for our analysis: the more performant feature extraction methods will improve cluster separation which will be shown by a higher K-Means performance.

## Performance metrics

It is worth mentioning that, despite its application in evaluating spike sorting methods (54) (55), accuracy is not a suitable performance metric. The primary issue with accuracy is that spike sorting is an unsupervised task, where ground truth labels are not present. Due to the fact that accuracy requires the ground truth labels to evaluate performance it is rendered impractical. Additionally, as previously stated, neuronal data is inherently unbalanced due to the different firing rates of neurons and it has been widely demonstrated that accuracy is inadequate in measuring performance on imbalanced datasets (56) (57) (58) (59). Nonetheless, the selected metrics allow us to assess cluster separation and structure using internal metrics, and to evaluate the “accuracy” of clustering through external metrics.

Six metrics were used to evaluate the outcomes, the first three are external measures while the latter three serve as internal measures (60). They are: Adjusted Rand Index (ARI), Adjusted Mutual Information (AMI), V-Measure (VM), Calinski-Harabasz Score (CHS), Davies-Bouldin Score (DBS), and Silhouette Score (SS). Since spike sorting ends in clustering, these clustering metrics are suitable for evaluating the quality of feature extraction. External metrics measure how accurately the clustering algorithm recovers known class labels; they reflect how distinctly features separate the data—if features are perfectly separated then typically high external scores are achieved by the clustering algorithms. Internal metrics measure cluster compactness, separation, and shape as independent of ground truth; therefore they are appropriate for judging feature extraction when synthetic datasets provide true labels. Internal metrics describe cluster structure; external metrics measure agreement with true labels.

Using multiple measures instead of one index helps us take different aspects of clustering performance into account. Table 1 gives the intuitive meaning for each metric and its scoring range. A technique that does well across these varied measures suggests a more sound and even approach to feature extraction, lessening the chance of bias in the assessment.

**Table 1** – A short description of each performance evaluation metric, specifying its type and range.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Range [worst, best] |
| ARI | External | Chance‐corrected score based on pairwise comparisons of objects, rewarding when pairs are either consistently grouped together or separated in both predicted and ground‐truth clusters | [-1, 1] |
| AMI | External | Chance‐corrected score based on entropy‐based mutual information between the predicted and actual labels, with an adjustment for the expected value under random assignments | [0, 1] |
| Purity | External | Fraction of correctly assigned points across all clusters by assigning each cluster to the majority true class within it. | [0, 1] |
| DBS | Internal | Average similarity ratio of each cluster with its most similar cluster, where similarity is defined as the sum of within‐cluster scatter relative to between‐cluster separation. | (Inf, 0] |
| CHS | Internal | Ratio of between‐cluster dispersion to within‐cluster dispersion, normalized by the number of clusters and total points. | [0, Inf) |
| SS | Internal | Average across all data points for the normalized difference between its mean intra‐cluster distance and lowest mean inter‐cluster distance | [-1, 1] |

### External metrics

External metrics require the ground truth labels to be compared with the predicted labels. Furthermore, all these metrics are bounded with higher values being more desirable.

ARI (61) (62) (63) (3) extends the Rand Index (RI) metric to account for chance agreements. Essentially, RI (64) (2) computes is score as a pairwise comparison whether both set of labels (predicted and true) are aligned (agreements where both consider two data points in the same cluster or in different clusters) or not (disagreements). The following formulas describe the computation of these metrics:

Here, *ExpectedRI* is theexpected score if clusters were assigned randomly, estimated via a contingency table using permutations, *MaxRI* is 1, the maximum value of the score (61).

AMI (62) (65) (5) extends the Mutual Information (MI) (4) metric by incorporating entropy (*H*) into its computation. AMI also incorporates the normalization component (62) (66) (67) of Normalized Mutual Information. It measures the mutual dependence between two clusters and is described by the following equations:

Here, *U* and *V* are the two clusters, *N* is the total number of data points and *|X|* is the size of a given subset *X*.

Purity (60,68) computes the percentage of samples clustered correctly. This is computed as the ratio between the sum of the maximum intersections between the true and predicted labels for each cluster by the total number of samples. Thus, Purity can be viewed as a measure of how many of the samples of the predicted cluster belong to a single true cluster. The following formulas describe the computation of this metric:

Here, *N* represents the total number of samples in the dataset, *k* is the number of clusters in the set of predicted labels, *Ci* represents the samples of a cluster, *i*,of the predicted set of labels and *L* is the set of true labels.

### Internal metrics

Internal metrics do not require a ground truth to be used. They evaluate the intra-cluster and inter-cluster distances, thus evaluating the morphology of the clusters. Thus, internal metrics are biased toward dense and well-separated clusters. Even correct clusterings in which clusters do not respect this criteria can receive lower scores. For the evaluation of the synthetic datasets, internal metrics were used with the ground truth labels. This results in an evaluation of the capabilities of feature extraction methods to generate clusters (that based on the true labels) are dense and well-separated.

DBS (69–71)(10) is computed as the average similarity of clusters. The similarity is computed using the distance between clusters and their sizes. DBS has an inverse performance interval to the other metrics presented in this work. It is has only a lower bound at 0 and lower values represent a higher performance. The following formulas describe the computation of this metric:

Here, *R* represents the similarity between clusters *i* and *j*, *si* is the mean of all distances between the points of cluster *i* and its centroid, *di,j* is the distance between clusters *i* and *j* given by their centroids, and *max(Ri,j)* is the maximum similarity of clusters *i* and *j*.

CHS (60) (72) (11), or Variance Ratio Criterion, is computed as the ratio between the intra-cluster to inter-cluster dispersion. The dispersion is based on the sum of squared distances. For this metric, a higher value indicates a better result and it has no upper bound. The following formula describes the computation of this metric:

Here, *tr(X)* is the trace of the dispersion matrix (either between *Bk* or within *Wk*), *n* is the dataset size and *k* is the number of clusters.

SS (72) (73) (12) is computed as the ratio between the mean distance between a point and the rest of the points of that cluster and the mean distance between the point and all the points of the nearest cluster. SS has an interval of [-1, 1] where 1 represents well-separated dense clusters, 0 overlapping clusters, and -1 an incorrect clustering. Thus, SS evaluates as correct (and outputs higher scores for) the traditional structure of clusters. The following formula describes the computation of this metric:

Here, *b* is the mean of all distances between a point in cluster *i* and all points of the closest cluster *j*, and *a* is the mean of all distances between a point in cluster *i* and all other points in the same cluster.

## Data

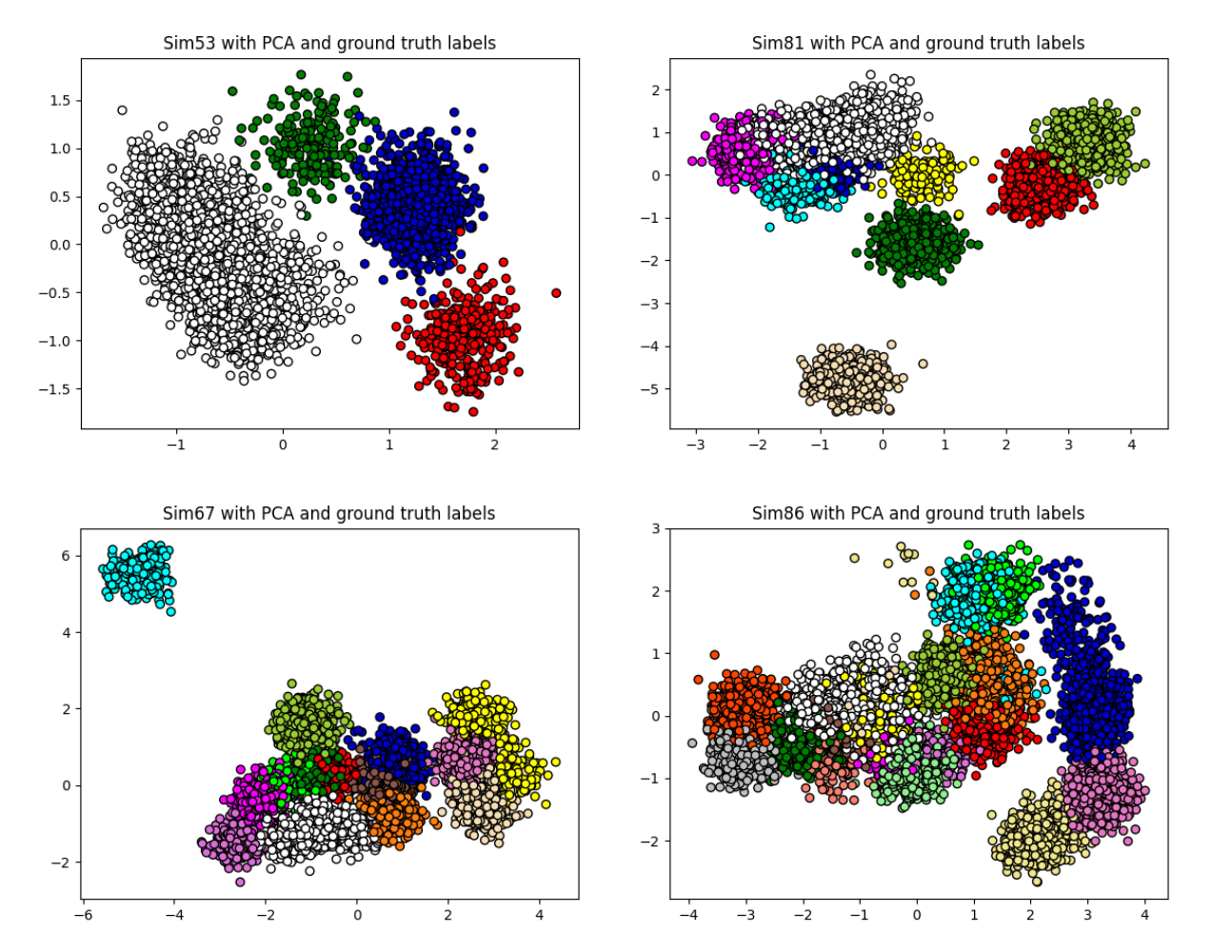
Ninety-five synthetic datasets (16), referred to as simulations (by the authors), were used in the analyses presented in this work. These datasets (16) were created by the Department of Engineering, University of Leicester UK and are publicly available. Each dataset is derived from 594 distinct spike waveforms obtained from real brain recordings of a monkey. The original publication (16) has also investigated the ability of various clustering algorithms on these datasets and it has been found that at best, they were able to identify 10 out of 20 true units.

Initially, the spike waveforms consisted of 316 samples at a 96 kHz sampling frequency. However, the datasets were downsampled to 24 kHz, yielding 79 samples per spike. Each of these datasets consisnt of a varying number of 2 to 20 single unit clusters and a multi-unit cluster. The single-unit clusters lie within 0 and 50μm of the electrode, their amplitude are normally distributed and scaled between 0.9 and 2 to mimic real data, and their firing rate follows Poisson distribution with a mean between 0.1 and 2Hz. The multi-unit cluster introduces complexity into the dataset. It was created through the aggregation of 20 unique neurons (thus, different spike shapes) within 50-140μm of the electrode, their amplitude was fixed to 0.5 with a collective firing rate of 5Hz (while each unique neuron fires at 0.25 Hz under an independent Poisson process). Spikes never overlap in time, it was ensured that spikes have a time separation of at least 0.3ms. Each individual cluster count has 5 independent datasets, meaning that there are 5 datasets with 2 single unit cluster, 5 with 3 single unit clusters and so on. Each of these synthetic datasets carries with it a set of ground truth labels. This allows for the evaluation of performance using external metrics as well.

A detailed comparison of the methods was made. Four representative simulations have been chosen for their variety in cluster count from the 95 datasets allowing for the performance evaluation of feature extraction methods covering a wide range. The selected set of simulations can be viewed in Fig 1, PCA was used to obtain a 2-dimensional representation. A short description of each of these simulations follows:

* Simulation 53 (Sim53 - Fig 1) is composed of 4490 spikes distributed in 3 single-unit clusters and a multi-unit cluster (in total 4). This dataset was used for the visualization of the impact of alignment on feature extraction.
* Simulation 81 (Sim81 - Fig 1) is composed of 7937 spikes distributed in 8 single-unit clusters and a multi-unit cluster (in total 9).
* Simulation 67 (Sim67 - Fig 1) is composed of 11377 spikes distributed in 13 single-unit clusters and a multi-unit cluster (in total 14).
* Simulation 86 (Sim86 - Fig 1) is composed of 13847 spikes distributed in 18 single-unit clusters and a multi-unit cluster (in total 19).

A general comparison of the methods was also made, where all 95 datasets have been analyzed.

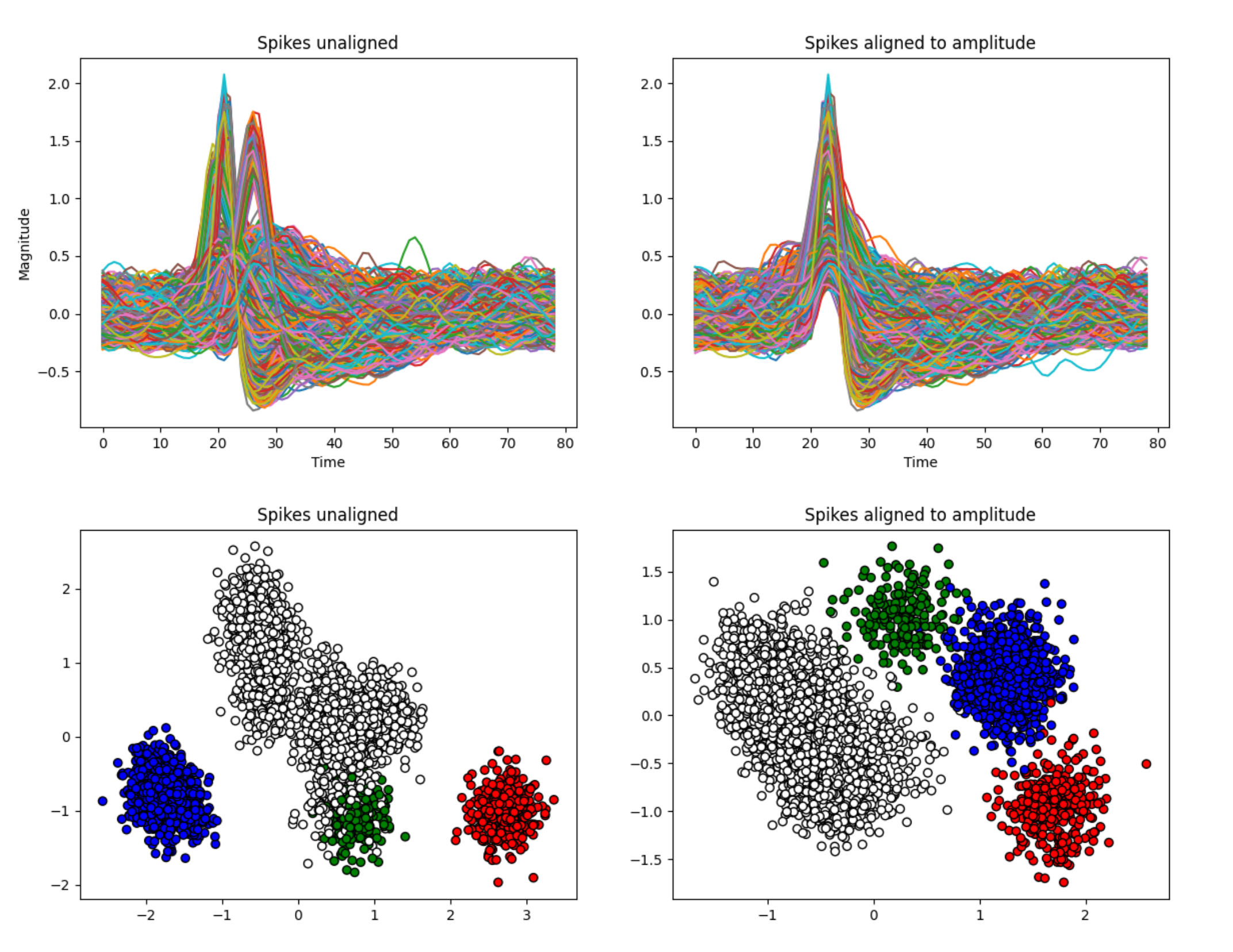


**Fig 1. Synthetic datasets presented with PCA and ground truth labels.** Four different simulations were reduced to a 2-dimensional space using PCA. The colors represent the true clusters indicating that PCA is unable to find a set of features that offer cluster separability.

## Data preprocessing

The spikes obtained from these datasets have went through a preprocessing before applying the feature extraction methods. Alignment of spikes to their amplitude can help in the separation of clusters by feature extraction methods. The result of this process is presented in Fig 2 where the effect on the spikes and on the PCA transformation of the spikes can be viewed. The following formula was applied:

Here, *old\_start* and *start\_start* denote the original and shifted start positions of each spike segment, respectively, *index* represents the target position (amplitude) to which all spikes will be shifted, and *peak* represents the position at which the desired point of reference (typically, the peak) is found. Thus, all spikes have been shifted such that the amplitude, or maximum peak, can be found at a given index. This formula allows for the alignment of any point of reference, such as the minimum peak (74), to any chosen position.



**Fig 4. Impact of alignment on Sim53.** The top row presents the impact of alignment on the spikes themselves, while the bottom row presents the impact of alignment on the PCA features. The white cluster is kept together and the green cluster is now separated from the white cluster.

# Results

## Performance evaluation

### Performance evaluation of synthetic data [TBRewritten]

The 95 synthetic datasets (16) contain varying numbers of clusters and spike shapes providing the complexity required for a comprehensive evaluation of the methods. In Fig 5, we present the results obtained for each metric across all 95 datasets for each method presented. A statistical analysis using t-tests with a Bonferroni correction can be examined in the S4 Fig and a ranking of the methods based on their performance for each metric using Borda rank aggregation (75) in S2 Table.

Sim53

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Algorithm | ARI | AMI | Purity | SS | CHS | DBS |
| PCA | 0.52 | 0.701 | 0.953 | 0.205 | **1635.47** | 1.655 |
| MDS | 0.465 | 0.597 | 0.905 | 0.156 | 1430.06 | 2.052 |
| ICA | 0.521 | 0.705 | 0.951 | 0.199 | 1611.126 | 1.686 |
| KPCA | 0.518 | 0.69 | 0.949 | 0.201 | 1620.848 | 1.691 |
| SOM | 0.362 | 0.503 | 0.837 | 0.025 | 594.599 | 6.263 |
| AE | 0.48 | 0.735 | 0.736 | 0.616 | 39350.58 | 0.373 |
| LLE | **0.999** | **0.998** | **1** | 0.317 | 1496.737 | 1.155 |
| MLLE | **0.999** | **0.995** | **0.999** | 0.317 | 1497.1 | 1.158 |
| Keppler Mapper | 0.522 | 0.701 | 0.955 | 0.193 | 1565.726 | 1.762 |
| Isomap | 0.536 | 0.697 | 0.92 | 0.187 | 1463.637 | 1.687 |
| Spectral embedding | 0.534 | 0.714 | 0.957 | 0.212 | 1632.349 | 1.653 |
| t-SNE | 0.488 | 0.67 | 0.924 | 0.188 | 1372.51 | 2.039 |
| Diffusion Map | 0.959 | 0.931 | 0.957 | **0.338** | 1260.056 | **0.981** |
| PHATE | 0.531 | 0.735 | 0.957 | 0.202 | 1613.265 | 1.659 |
| UMAP | 0.911 | 0.901 | 0.968 | 0.311 | 1383.718 | 1.435 |
| Trimap | 0.508 | 0.716 | 0.929 | 0.203 | 1517.197 | 1.578 |

Sim81

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Algorithm | ARI | AMI | Purity | SS | CHS | DBS |
| PCA | 0.528 | 0.744 | 0.795 | 0.23 | 5407.828 | 1.841 |
| MDS | 0.619 | 0.774 | 0.813 | 0.207 | 5327.273 | 6.976 |
| ICA | 0.462 | 0.711 | 0.804 | 0.197 | 5317.578 | 2.099 |
| KPCA | 0.562 | 0.767 | 0.844 | 0.226 | 5310.696 | 1.946 |
| SOM | 0.481 | 0.697 | 0.83 | 0.193 | 3954.929 | 1.861 |
| AE | 0.569 | 0.809 | 0.81 | 0.419 | 15502.94 | 0.982 |
| LLE | 0.515 | 0.737 | 0.784 | 0.216 | 4384.025 | 2.421 |
| MLLE | 0.513 | 0.786 | 0.756 | 0.274 | 3752.723 | **1.093** |
| Keppler Mapper | 0.466 | 0.682 | 0.772 | 0.181 | 4936.57 | 2.86 |
| Isomap | 0.647 | 0.819 | 0.903 | 0.266 | 5759.539 | 1.423 |
| Spectral embedding | 0.609 | 0.789 | 0.806 | 0.255 | 4688.755 | 2.259 |
| t-SNE | 0.721 | 0.885 | 0.948 | 0.28 | 5822.569 | 1.431 |
| Diffusion Map | **0.802** | **0.88** | **0.867** | **0.294** | 4973.175 | 1.205 |
| PHATE | 0.725 | 0.9 | 0.933 | 0.287 | 6008.121 | 1.314 |
| UMAP | 0.739 | 0.912 | 0.938 | 0.287 | 5996.497 | 1.3 |
| Trimap | 0.733 | 0.909 | 0.937 | 0.289 | **6024.442** | 1.298 |

Sim67

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Algorithm | ARI | AMI | Purity | SS | CHS | DBS |
| PCA | 0.485 | 0.72 | 0.779 | 0.169 | 3338.865 | 1.875 |
| MDS | 0.554 | 0.774 | 0.815 | 0.206 | 3701.141 | 2.314 |
| ICA | 0.519 | 0.727 | 0.802 | 0.174 | 3528.657 | 1.787 |
| KPCA | 0.371 | 0.622 | 0.666 | 0.088 | 2722.653 | 2.735 |
| SOM | 0.409 | 0.653 | 0.694 | 0.069 | 1617.331 | 2.874 |
| AE | 0.726 | 0.894 | 0.894 | 0.453 | 22183.83 | 0.819 |
| LLE | 0.47 | 0.701 | 0.742 | 0.121 | 3108.751 | 2.523 |
| MLLE | 0.664 | 0.79 | 0.731 | 0.219 | 2649.706 | 1.348 |
| Keppler Mapper | 0.634 | 0.774 | 0.846 | 0.178 | 4071.876 | 2.526 |
| Isomap | 0.57 | 0.783 | 0.816 | 0.22 | 4224.48 | 1.711 |
| Spectral embedding | 0.573 | 0.768 | 0.786 | 0.147 | 2571.183 | 2.382 |
| t-SNE | 0.678 | 0.88 | 0.893 | 0.243 | 4582.876 | 1.538 |
| Diffusion Map | 0.163 | 0.522 | 0.499 | **0.352** | 1267.865 | 1.542 |
| PHATE | **0.757** | **0.9** | **0.925** | 0.264 | 4610.405 | 1.437 |
| UMAP | **0.758** | **0.916** | **0.92** | 0.272 | 4792.607 | **1.314** |
| Trimap | **0.759** | **0.916** | **0.92** | 0.275 | 4804.884 | **1.311** |

Sim86

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Algorithm | ARI | AMI | Purity | SS | CHS | DBS |
| PCA | 0.466 | 0.713 | 0.739 | 0.125 | 2859.273 | 2.733 |
| MDS | 0.583 | 0.783 | 0.832 | 0.144 | 3378.853 | 2.623 |
| ICA | 0.431 | 0.693 | 0.716 | 0.109 | 2760.434 | 2.939 |
| KPCA | 0.467 | 0.705 | 0.727 | 0.105 | 2821.59 | 3.132 |
| SOM | 0.483 | 0.74 | 0.729 | 0.091 | 2546.982 | 2.325 |
| AE | 0.962 | 0.92 | 0.92 | 0.705 | 38335.08 | 0.473 |
| LLE | 0.469 | 0.746 | 0.732 | 0.115 | 2760.308 | 2.822 |
| MLLE | 0.602 | 0.826 | 0.714 | 0.186 | 2323.771 | 1.891 |
| Keppler Mapper | 0.625 | 0.836 | 0.856 | 0.232 | 4125.113 | 1.863 |
| Isomap | 0.588 | 0.788 | 0.793 | 0.166 | 3518.915 | 2.597 |
| Spectral embedding | 0.5 | 0.718 | 0.694 | 0.11 | 2622.293 | 2.804 |
| t-SNE | 0.781 | 0.914 | 0.954 | 0.214 | 4286.09 | 1.871 |
| Diffusion Map | 0.317 | 0.652 | 0.475 | **0.286** | 1434.835 | **0.897** |
| PHATE | 0.707 | 0.907 | 0.905 | 0.25 | 4089.876 | 1.632 |
| UMAP | **0.781** | **0.921** | **0.939** | 0.24 | **4461.659** | 1.567 |
| Trimap | 0.683 | 0.902 | 0.902 | 0.254 | 4429.627 | 1.603 |

# Discussion

Evidently, other clustering algorithms may be used and may even obtain better results. However, we have chosen to analyze feature extraction algorithms and the separability of the embeddings spaces that they offer. As such, a clustering algorithm such as K-Means which separates any spaces in a linear manner is a perfect candidate to explore the separability of any feature extraction method. Moreover, as K-Means is one of the fastest algorithms and is commonly used in spike sorting, including some of the more recently developed spike sorters (14,76).

# Bibliography

1. Carter M, Shieh J. Chapter 4 - Electrophysiology. In: Carter M, Shieh J, editors. Guide to Research Techniques in Neuroscience (Second Edition) [Internet]. San Diego: Academic Press; 2015 [cited 2022 Aug 2]. p. 89–115. Available from: https://www.sciencedirect.com/science/article/pii/B9780128005118000046

2. Quiroga RQ. Spike sorting. Scholarpedia. 2007 Dec 21;2(12):3583.

3. Bear M, Connors B, Paradiso M. Neuroscience: Exploring the brain: Fourth edition. 2015. 1 p.

4. Steinmetz NA, Aydin C, Lebedeva A, Okun M, Pachitariu M, Bauza M, et al. Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. Science. 2021 Apr 16;372(6539):eabf4588.

5. Rey HG, Pedreira C, Quian Quiroga R. Past, present and future of spike sorting techniques. Brain Res Bull. 2015 Oct 1;119:106–17.

6. Meister M, Pine J, Baylor DA. Multi-neuronal signals from the retina: acquisition and analysis. J Neurosci Methods. 1994 Jan 1;51(1):95–106.

7. Chung JE, Magland JF, Barnett AH, Tolosa VM, Tooker AC, Lee KY, et al. A fully automated approach to spike sorting. Neuron. 2017 Sep 13;95(6):1381-1394.e6.

8. Ebbesen CL, Reifenstein ET, Tang Q, Burgalossi A, Ray S, Schreiber S, et al. Cell Type-Specific Differences in Spike Timing and Spike Shape in the Rat Parasubiculum and Superficial Medial Entorhinal Cortex. Cell Rep. 2016 Jul 14;16(4):1005–15.

9. Stevenson IH, Kording KP. How advances in neural recording affect data analysis. Nat Neurosci. 2011 Feb;14(2):139–42.

10. Jun JJ, Steinmetz NA, Siegle JH, Denman DJ, Bauza M, Barbarits B, et al. Fully integrated silicon probes for high-density recording of neural activity. Nature. 2017 Nov;551(7679):232–6.

11. Pouzat C, Mazor O, Laurent G. Using noise signature to optimize spike-sorting and to assess neuronal classification quality. J Neurosci Methods. 2002 Dec 31;122(1):43–57.

12. Litke AM, Bezayiff N, Chichilnisky EJ, Cunningham W, Dabrowski W, Grillo AA, et al. What does the eye tell the brain?: Development of a system for the large-scale recording of retinal output activity. IEEE Trans Nucl Sci. 2004 Aug;51(4):1434–40.

13. Hulata E, Segev R, Ben-Jacob E. A method for spike sorting and detection based on wavelet packets and Shannon’s mutual information. J Neurosci Methods. 2002 May 30;117(1):1–12.

14. Pachitariu M, Steinmetz N, Kadir S, Carandini M, D HK. Kilosort: realtime spike-sorting for extracellular electrophysiology with hundreds of channels [Internet]. bioRxiv; 2016 [cited 2022 Jul 18]. p. 061481. Available from: https://www.biorxiv.org/content/10.1101/061481v1

15. Yuan Y, Yang C, Si J. The M-Sorter: An automatic and robust spike detection and classification system. J Neurosci Methods. 2012 Sep 30;210(2):281–90.

16. Pedreira C, Martinez J, Ison MJ, Quian Quiroga R. How many neurons can we see with current spike sorting algorithms? J Neurosci Methods. 2012 Oct 15;211(1):58–65.

17. Estivill-Castro V. Why so many clustering algorithms: a position paper. SIGKDD Explor Newsl. 2002 Jun 1;4(1):65–75.

18. Multivariate Analysis [Internet]. 1979 [cited 2025 May 4]. Available from: https://shop.elsevier.com/books/multivariate-analysis/mardia/978-0-08-057047-1

19. Tenenbaum JB, de Silva V, Langford JC. A global geometric framework for nonlinear dimensionality reduction. Science. 2000 Dec 22;290(5500):2319–23.

20. Mishra S, Sarkar U, Taraphder S, Datta S, Swain D, Saikhom R, et al. Principal Component Analysis. Int J Livest Res. 2017 Jan 1;1.

21. Adamos DA, Kosmidis EK, Theophilidis G. Performance evaluation of PCA-based spike sorting algorithms. Comput Methods Programs Biomed. 2008 Sep 1;91(3):232–44.

22. Belkin M, Niyogi P. Laplacian Eigenmaps for dimensionality reduction and data representation. Neural Comput. 2003 Jun 1;15(6):1373–96.

23. Amid E, Warmuth MK. TriMap: Large-scale Dimensionality Reduction Using Triplets [Internet]. arXiv; 2022 [cited 2025 May 2]. Available from: http://arxiv.org/abs/1910.00204

24. McInnes L, Healy J, Melville J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction [Internet]. arXiv; 2020 [cited 2025 May 2]. Available from: http://arxiv.org/abs/1802.03426

25. Buzsáki G. Rhythms of the Brain [Internet]. New York: Oxford University Press; 2006 [cited 2021 Dec 8]. 464 p. Available from: https://oxford.universitypressscholarship.com/10.1093/acprof:oso/9780195301069.001.0001/acprof-9780195301069

26. Lewicki MS. A review of methods for spike sorting: the detection and classification of neural action potentials. Netw Bristol Engl. 1998 Nov;9(4):R53-78.

27. Meilă M, Zhang H. Manifold learning: what, how, and why [Internet]. arXiv; 2023 [cited 2025 May 4]. Available from: http://arxiv.org/abs/2311.03757

28. Dimensionality reduction: a comparative review | BibSonomy [Internet]. [cited 2022 Aug 11]. Available from: https://www.bibsonomy.org/bibtex/2ed03568f0e9bca9cdaf6b25304e55940/peter.ralph

29. Toosi R, Akhaee MA, Dehaqani MRA. An automatic spike sorting algorithm based on adaptive spike detection and a mixture of skew-t distributions. Sci Rep. 2021 Jul 6;11(1):13925.

30. Glaser EM, Marks WB. ON-LINE SEPARATION OF INTERLEAVED NEURONAL PULSE SEQUENCES. In: Enslein K, editor. Data Acquisition and Processing in Biology and Medicine [Internet]. Pergamon; 1968 [cited 2022 Aug 11]. p. 137–56. Available from: https://www.sciencedirect.com/science/article/pii/B9780080035437500124

31. Abeles M, Goldstein MH. Multispike train analysis. Proc IEEE. 1977 May;65(5):762–73.

32. Borg I, Groenen PJF, editors. Constructing MDS Representations. In: Modern Multidimensional Scaling: Theory and Applications [Internet]. New York, NY: Springer; 2005 [cited 2025 May 2]. p. 19–35. Available from: https://doi.org/10.1007/0-387-28981-X\_2

33. Hyvärinen A. Independent component analysis: recent advances. Philos Transact A Math Phys Eng Sci. 2013 Feb 13;371(1984):20110534.

34. Tiganj Z, Mboup M. Neural spike sorting using iterative ICA and a deflation-based approach. J Neural Eng. 2012 Dec;9(6):066002.

35. Lopes MV, Aguiar E, Santana E, Santana E, Barros AK. ICA feature extraction for spike sorting of single-channel records. In: 2013 ISSNIP Biosignals and Biorobotics Conference: Biosignals and Robotics for Better and Safer Living (BRC). 2013. p. 1–5.

36. Schölkopf B, Smola A, Müller KR. Kernel principal component analysis. In: Gerstner W, Germond A, Hasler M, Nicoud JD, editors. Artificial Neural Networks — ICANN’97. Berlin, Heidelberg: Springer; 1997. p. 583–8.

37. Kohonen T. Self-organized formation of topologically correct feature maps. Biol Cybern. 1982 Jan 1;43(1):59–69.

38. Pinaya W, Vieira S, Garcia-Dias R, Mechelli A. Autoencoders. In 2019. p. 193–208.

39. Baldi P. Autoencoders, Unsupervised Learning, and Deep Architectures. In: Proceedings of ICML Workshop on Unsupervised and Transfer Learning [Internet]. JMLR Workshop and Conference Proceedings; 2012 [cited 2021 Dec 22]. p. 37–49. Available from: https://proceedings.mlr.press/v27/baldi12a.html

40. Roweis ST, Saul LK. Nonlinear Dimensionality Reduction by Locally Linear Embedding. Science. 2000 Dec 22;290(5500):2323–6.

41. Zhang Z, Wang J. MLLE: Modified Locally Linear Embedding Using Multiple Weights. In: Advances in Neural Information Processing Systems [Internet]. MIT Press; 2006 [cited 2025 May 2]. Available from: https://proceedings.neurips.cc/paper/2006/hash/fb2606a5068901da92473666256e6e5b-Abstract.html

42. Donoho DL, Grimes C. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. Proc Natl Acad Sci. 2003 May 13;100(10):5591–6.

43. Zhang Z, Zha H. Principal Manifolds and Nonlinear Dimension Reduction via Local Tangent Space Alignment [Internet]. arXiv; 2002 [cited 2025 May 2]. Available from: http://arxiv.org/abs/cs/0212008

44. Singh G, Memoli F, Carlsson G. Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition [Internet]. The Eurographics Association; 2007 [cited 2025 May 2]. Available from: https://doi.org/10.2312/SPBG/SPBG07/091-100

45. Zhou H, Wang F, Tao P. t-Distributed Stochastic Neighbor Embedding Method with the Least Information Loss for Macromolecular Simulations. J Chem Theory Comput. 2018 Nov 13;14(11):5499–510.

46. Variable bandwidth diffusion kernels. Appl Comput Harmon Anal. 2016 Jan 1;40(1):68–96.

47. Moon KR, van Dijk D, Wang Z, Gigante S, Burkhardt DB, Chen WS, et al. Visualizing structure and transitions in high-dimensional biological data. Nat Biotechnol. 2019 Dec;37(12):1482–92.

48. Lee K, Carr N, Perliss A, Chandrasekaran C. WaveMAP for identifying putative cell types from *in vivo* electrophysiology. STAR Protoc. 2023 Jun 16;4(2):102320.

49. Lee EK, Balasubramanian H, Tsolias A, Anakwe SU, Medalla M, Shenoy KV, et al. Non-linear dimensionality reduction on extracellular waveforms reveals cell type diversity in premotor cortex. eLife. 10:e67490.

50. MacQueen J. Some methods for classification and analysis of multivariate observations. Proc Fifth Berkeley Symp Math Stat Probab Vol 1 Stat. 1967 Jan 1;5.1:281–98.

51. Veerabhadrappa R, Ul Hassan M, Zhang J, Bhatti A. Compatibility Evaluation of Clustering Algorithms for Contemporary Extracellular Neural Spike Sorting. Front Syst Neurosci [Internet]. 2020 [cited 2022 Jul 18];14. Available from: https://www.frontiersin.org/articles/10.3389/fnsys.2020.00034

52. Salganicoff M, Sarna M, Sax L, Gerstein GL. Unsupervised waveform classification for multi-neuron recordings: a real-time, software-based system. I. Algorithms and implementation. J Neurosci Methods. 1988 Oct;25(3):181–7.

53. Caro-Martín CR, Delgado-García JM, Gruart A, Sánchez-Campusano R. Spike sorting based on shape, phase, and distribution features, and K-TOPS clustering with validity and error indices. Sci Rep. 2018 Dec 12;8(1):17796.

54. Eom J, Park IY, Kim S, Jang H, Park S, Huh Y, et al. Deep-learned spike representations and sorting via an ensemble of auto-encoders. Neural Netw. 2021 Feb 1;134:131–42.

55. Radmanesh M, Rezaei AA, Jalili M, Hashemi A, Goudarzi MM. Online spike sorting via deep contractive autoencoder. Neural Netw [Internet]. 2022 Aug 5 [cited 2022 Aug 11]; Available from: https://www.sciencedirect.com/science/article/pii/S089360802200301X

56. Wegier W, Ksieniewicz P. Application of Imbalanced Data Classification Quality Metrics as Weighting Methods of the Ensemble Data Stream Classification Algorithms. Entropy Basel Switz. 2020 Jul 31;22(8):E849.

57. Sun Y, Wong AKC, Kamel MS. Classification of imbalanced data: a review. Int J Pattern Recognit Artif Intell. 2009 Jun;23(04):687–719.

58. Joshi MV, Kumar V, Agarwal RC. Evaluating boosting algorithms to classify rare classes: comparison and improvements. In: Proceedings 2001 IEEE International Conference on Data Mining. 2001. p. 257–64.

59. Weiss GM. Mining with rarity: a unifying framework. ACM SIGKDD Explor Newsl. 2004 Jun;6(1):7–19.

60. Rendón E, Abundez I, Arizmendi A, Quiroz EM. Internal versus External cluster validation indexes. 2011;5(1):8.

61. Hubert L, Arabie P. Comparing partitions. J Classif. 1985 Dec 1;2(1):193–218.

62. Vinh NX, Epps J, Bailey J. Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. :18.

63. Steinley D. Properties of the Hubert-Arable Adjusted Rand Index. Psychol Methods. 2004;9(3):386–96.

64. Fowlkes EB, Mallows CL. A Method for Comparing Two Hierarchical Clusterings. J Am Stat Assoc. 1983;78(383):553–69.

65. Strehl A, Ghosh J. Cluster Ensembles --- A Knowledge Reuse Framework for Combining Multiple Partitions. J Mach Learn Res. 2002;3(Dec):583–617.

66. Lazarenko D, Bonald T. Pairwise Adjusted Mutual Information. 2021.

67. Vinh N, Epps J, Bailey J. Information theoretic measures for clusterings comparison: Is a correction for chance necessary? ICML. 2009. 135 p.

68. Manning CD, Raghavan P, Schütze H. Introduction to Information Retrieval. Illustrated edition. New York: Cambridge University Press; 2008. 506 p.

69. Halkidi M, Batistakis Y, Vazirgiannis M. On Clustering Validation Techniques. J Intell Inf Syst. 2001 Dec 1;17(2):107–45.

70. Caliński T, JA H. A Dendrite Method for Cluster Analysis. Commun Stat - Theory Methods. 1974 Jan 1;3:1–27.

71. Davies DL, Bouldin DW. A Cluster Separation Measure. IEEE Trans Pattern Anal Mach Intell. 1979 Apr;PAMI-1(2):224–7.

72. Rosenberg A, Hirschberg J. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In 2007. p. 410–20.

73. Rousseeuw PJ. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math. 1987 Nov 1;20:53–65.

74. Dipalo M, Amin H, Lovato L, Moia F, Caprettini V, Messina G, et al. Intracellular and Extracellular Recording of Spontaneous Action Potentials in Mammalian Neurons and Cardiac Cells with 3D Plasmonic Nanoelectrodes. Nano Lett. 2017 May 23;17.

75. Dwork C, Kumar R, Naor M, Sivakumar D. Rank aggregation methods for the Web. In: Proceedings of the 10th international conference on World Wide Web [Internet]. New York, NY, USA: Association for Computing Machinery; 2001 [cited 2022 Dec 6]. p. 613–22. (WWW ’01). Available from: https://doi.org/10.1145/371920.372165

76. Pachitariu M, Sridhar S, Pennington J, Stringer C. Spike sorting with Kilosort4. Nat Methods. 2024 May;21(5):914–21.