energies

Article

Manifold Learning in Electric Power System Transient Stability Analysis

Petar Sarajcev * and Dino Lovric

Department of Electric Power Systems, Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, University of Split, R. Boskovica 32, HR-21000 Split, Croatia; dino.lovric@fesb.hr

* Correspondence: petar.sarajcev@fesb.hr

Abstract: This paper examines the use of manifold learning in the context of electric power system transient stability analysis. Since wide-area monitoring systems (WAMSs) introduced a big data paradigm into the power system operation, manifold learning can be seen as a means of condensing these high-dimensional data into an appropriate low-dimensional representation (i.e., embedding) which preserves as much information as possible. In this paper, we consider several embedding methods (principal component analysis (PCA) and its variants, singular value decomposition, isomap and spectral embedding, locally linear embedding (LLE) and its variants, multidimensional scaling (MDS), and others) and apply them to the dataset derived from the IEEE New England 39-bus power system transient simulations. We found that PCA with a radial basis function kernel is well suited to this type of power system data (where features are instances of three-phase phasor values). We also found that the LLE (including its variants) did not produce a good embedding with this particular kind of data. Furthermore, we found that a support vector machine, trained on top of the embedding produced by several different methods was able to detect power system disturbances from WAMS data.

Keywords: electric power system; bulk power system; transient stability analysis; rotor angle stability; wide-area monitoring system; machine learning; manifold learning

check for updates

Citation: Sarajcev, P.; Lovric, D. Manifold Learning in Electric Power System Transient Stability Analysis. *Energies* **2023**, *16*, 7810. https://doi.org/10.3390/en16237810

Academic Editors: Marcin Kłos and Grzegorz Sierpiński

Received: 13 November 2023
Revised: 23 November 2023
Accepted: 25 November 2023
Published: 27 November 2023

1. Introduction

The electric power system features the most complex transport system to date. It is presently undergoing a very rapid and intensive transformation, as it is transitioning toward the renewable and sustainable future. The most important aspect of that transformation is the accelerating change in the mix of generating resources, which reduces the proportion of conventional power plants (with large synchronous generators) while increasing the share of inverter-based resources (IBRs), i.e., wind and solar power plants. This change in the generation mix represents a fundamental shift in the operational characteristics of the power system as such, with potential (significant) reliability implications for the electric grid [1]. Namely, generating resources need to provide on an ongoing basis, among others, voltage control, frequency support, and ramping capability in order to balance and maintain the grid [2]. However, the vast majority of IBRs today do not participate in this support. At the same time, generating resources need to provide system stability during transient events, brought on by faults (i.e., short-circuits), severe weather, and other internal and external disturbances. On this front, new resource mix will significantly (and irreversibly) reduce the power system inertia, thereby exacerbating the power system transient stability (TSA) problems of tomorrow. The response of the power grids with low system inertia to disturbances necessitates new approaches for dealing with the TSA issues [3]. The electric grid is also under increasing strain from the growing share of electric vehicles, which increase in the overall grid load and electricity demands. These combined stresses on the electric power system are without precedent and pose considerable technical challenges, which need to be successfully tackled if the electrification of the transport sector, and transition to the renewable energy, is to be realized.

These aforementioned developments are accompanied by still more innovations in the power system sector, one of which is the introduction of wide-area monitoring (WAM) aspects to the electric grid, based on extensive communication infrastructure and distributed phasor measurement units (PMUs). WAMSs bring a "big data" paradigm into the operational portfolio of the transmission system operators and provide novel avenues for tackling various power system problems, from load flow, state estimation, relay protection, to transient analyses [4,5]. Furthermore, electric power systems are integrating an increasing number of power electronic components, FACTS, AC/DC devices, thereby changing the response characteristics of the system to various disturbances [6]. All these different aspects set new challenges and, at the same time, provide opportunities for novel technical solutions. Namely, some well-known traditional solutions—particularly in the domain of transient stability analysis—have acknowledged weaknesses that may compromise the security of the bulk power system under increased stress emanating from the accelerated pace of its ongoing transformation.

The transient stability analysis of the electric power system is currently the focus of intense research [7], as it is increasingly becoming obvious that traditional solutions are not future-proof and are becoming increasingly maladjusted to rapid changes in modern systems. Neither time-domain simulations, direct methods (equal-area and Lyapunov methods), nor different hybrid schemes are able to deliver solutions that would satisfy the reliability requirements of modern power systems [8]. Hence, a new breed of methods is slowly emerging, based on artificial intelligence and WAMS data troves.

Machine learning (ML) and its subset of deep learning have been thus far applied to the TSA problem, with promising results on benchmark test cases. For example, Tan et al. used a deep imbalanced learning framework for the TSA of a power system [9]. Liu et al. introduced transient stability assessment models based on ensemble learning with network topology [10], distance metric learning [11] and kernel regression. Su et al. proposed a probabilistic stacked denoising autoencoder for a power system transient stability prediction with wind farms [12]. Autoencoder is a special kind of artificial neural network (ANN) that can be trained using an unsupervised learning approach. Tian et al. introduced a transient stability boundary, constructed using the broad learning approach [13]. Ren and Xu recommended a method for the TSA based on generative adversarial networks, where two ANNs contested with each other in an unsupervised manner [14]. Also, Ren et al. employed transfer learning for the transient stability analysis [15]. Transfer learning is the ability to reuse ANNs pretrained on one (general) task and repurpose them (with fine-tuning) for different but related (specialized) tasks. Li et al. suggested an intelligent TSA framework with continual learning ability [16]. An ensemble of decision trees was employed in [17–19] to tackle the TSA problem. Support vector machine (SVM) is a popular ML approach, used by many authors for tackling the TSA problem, e.g., [20–22]. In addition, Bashiri et al. introduced a learning framework for the TSA that relied on a twin convolutional SVM [23].

Inherent in the deep learning approach is the fact that the classical feature engineering step of the ML model building process is absent and replaced by an automatic learning of the features by the deep model itself during training [24]. This removes expert knowledge and instead relies on deep ANN architectures to learn useful features. However, a considerable preprocessing of the data (signals) is still needed for any deep learning model, even without classical feature engineering. Another important aspect of the TSA problem, which any machine learning model needs to tackle, is the fact that data space is multidimensional, often with many thousands of features or more. This poses a challenge to training ML models, and some kind of dimensionality reduction technique or embedding process is typically employed as part of the model training. For example, Chen in [25] applied an indirect principal component analysis (PCA) as a dimensionality reduction technique for a power system transient stability assessment. Bellizio et al. proposed a causality-based feature selection [26]. Gnanavignesh and Shenoy introduced a relaxation-based spatial-domain decomposition method [27]. Arvizu et al. used a diffusion maps approach for the dimensionality reduction in transient simulations [28]. Still, other signal feature extraction and dimension reduction techniques have been proposed, e.g., [29,30].

This paper is concerned with examining the use of manifold learning (as part of the wider ML landscape) in assessing the power system TSA, based on WAMS data and power system simulations. Manifold learning is an approach to the nonlinear dimensionality reduction of the data through its low-dimensional embedding that preserves maximum information. The paper features both supervised and unsupervised learning with different embedding methods in order to demonstrate their feasibility in the domain of power system transient stability analysis with WAMS-PMU data. Namely, our overview of the published peer-reviewed research did not show comprehensive comparisons of different embedding methods for the power system TSA with PMU data. Hence, we believe that this paper fills that need and brings new insights in the domain of applying manifold learning to the power system TSA problem. The following methods are examined [31]: (1) principal component analysis, (2) kernel PCA (kPCA), (3) truncated singular value decomposition (tSVD), (4) isomap embedding, (5) locally linear embedding (LLE) and its variants, (6) spectral embedding, (7) multidimensional scaling (MDS), and (8) *t*-distributed stochastic neighbor embedding (*t*-SNE). These methods are applied on the WAMS-PMU data derived from the IEEE New England 39-bus test-case benchmark power system [32]. The model hyperparameters are optimized using cross-validated metrics and a custom-tailored simulated-annealing stochastic optimization algorithm. When supervised learning is applied, it is based on a support vector machine classifier.

Our research hypothesis is that a low-dimensional embedding, derived from the WAMS-PMU data (features), preserves information and can detect power system TSA events in the supervised and unsupervised settings. However, it is shown that not all embedding methods produce equally good results on the TSA data, and that some are better suited than others. We find that a combination of kPCA with SVM classifier provides good overall results in the domain of supervised learning. The same can be said about the isomap embedding and MDS when used with the SVM. However, unlike kPCA, embeddings produced by isomap and MDS are not divisible into separate clusters, which makes them less favorable. We also find that LLE and its variants cannot produce useful embeddings with these data. Furthermore, when the unsupervised learning is concerned, we find that only kPCA is able to generate separable clusters. Finally, we find that the SVM classifier trained on top of several of the tested embeddings is able to detect power system disturbances from the WAMS-PMU data.

The paper is organized in the following manner. Section 2 first introduces in Section 2.1 a dataset which emanates from the IEEE New England test power system simulations and brings a description of the steps in its processing pipeline. Next, in Section 2.2, it exposes a simulated annealing stochastic optimization algorithm and its application to hyperparameter tuning. Section 2.3 introduces unsupervised and supervised learning for the power system transient stability assessment, with different embedding methods. The results of the models' applications on the IEEE New England dataset are provided in Section 3, which is followed by a discussion in Section 4 and a conclusion is Section 5.

2. Materials and Methods

The materials part introduces the dataset and describes its processing steps: statistical sampling, features engineering, shuffling, splitting, and normalizing. The methods part first describes the simulated annealing algorithm and then briefly introduces embedding models. Next, it provides details on the training process (i.e., supervised and unsupervised learning), including the hyperparameter optimization and model scoring.

2.1. Power System Stability Dataset

The power system stability dataset was produced from the extensive set of time-domain simulations on the IEEE New England 39-bus benchmark test power system, with varying degrees of loading and three different short-circuit types scattered throughout the network. Further details on the benchmark power system, time-domain simulations, and dataset construction can be found in [33]. The full 3.8 GB dataset of PMU signals was deposited on Zenodo with a CC BY license [34]. These raw signals were further processed according to the pipeline graphically depicted in Figure 1. It consisted of following steps: a statistical processing of data cases (from systematic into the stochastic domain), extracting features from the time-domain signals, splitting the data into training and test sets, and scaling them. The processing of data cases by random sampling without replacement followed a probability distribution of short-circuit events found in the actual power systems [33]. The splitting strategy used a stratified shuffle split in order to preserve the class imbalance between stable (majority) and unstable (minority) cases. The scaling standardized (training and test) the datasets by removing the mean and scaling them to a unit variance.

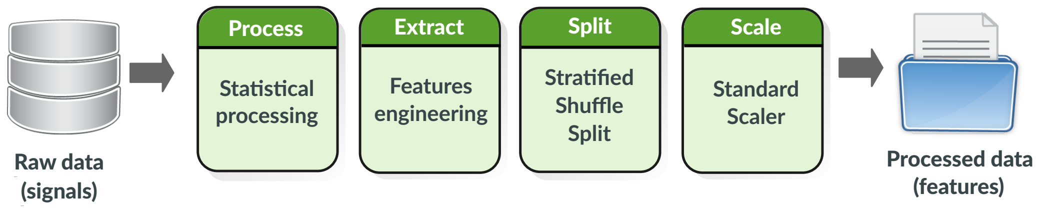


Figure 1. Data processing pipeline transforming the raw time-domain PMU signals into standardized training and test feature sets.

Table 1 presents selected features extracted from the time-domain PMU signals. Each time-domain signal was represented by only two points in time, based on prefault and postfault values of the measured quantity. It needs to be stated here that based on the fault location within the network, a coordination between generator out-of-step protection and distance relay protection of the incident transmission lines should determine the time instants for the feature selection. The labeling of data cases as stable or unstable was carried out by means of the power system transient stability index (TSI) [8,33]. The statistically processed dataset with 350+ features was also deposited on Zenodo with a CC BY license [35].

Table 1. Features derived from the time-domain PMU signals.

Feature Description	Number
Rotor speed	10
Rotor angle deviation	10
Rotor mechanical angle	10
Stator voltage	10
Stator d-component current	10
Stator q-component current	10
Power load angle	2×10 [1]
Generator active power	2×10 [1]
Generator reactive power	2×10 [1]
Bus voltage magnitude	$3 \times 2 \times 39$ [2]

[1] prefault and postfault values. [2] prefault and postfault values in all three phases.

It should be mentioned that we used expert knowledge in selecting these features, and that we had already removed some redundant features (e.g., one of the three phases, all phase angle values, etc.). Notwithstanding that, the number of features was still quite large for this small power system. More importantly, it grew considerably with the increase in number of buses and generators in the system and could easily become difficult to manage and process (i.e., curse of dimensionality). Hence, the main obstacle to the successful training of ML models was controlling and/or reducing this large number of features without significant loss of information.

2.2. Simulated Annealing Algorithm

Simulated annealing is a general-purpose stochastic optimization algorithm that usually consists of four main parts [36]: (1) an objective function that is being optimized, (2) a perturbation method that is used for generating candidate solutions, (3) an acceptance criterion, and (4) a temperature schedule (i.e., cooling). Our improvements on the classical algorithm constituted using the burn-in period, a multidimensional random walk based on Student's *t*-distribution as a perturbation method, and an early stopping criterion. This combination of burn-in with early stopping balanced exploration and exploitation while reducing the possibility of overfitting.

The objective function in our case constructed the model for which the hyperparameters were being optimized $f(\mathbf{x})$ and returned a cross-validated metric by which it was scored.

We employed an *n*-dimensional random walk as a perturbation method, where n is the model's number of hyperparameters x_i, $i = 1, \ldots, n$. For that purpose, we drew uncorrelated random samples from the multivariate statistical distribution in the following manner. During the initial phase of the algorithm's exploration of the search space (i.e., during a select number of initial iterations $i \leq k$), random samples were drawn from the multivariate Student's *t*-distribution as follows, $for\ i = 1, \ldots, k$:

$$\mathbf{z}_i \sim MVN(\mathbf{0}, \mathbf{\Sigma}), \tag{1}$$

$$u_i \sim \chi^2(\nu), \tag{2}$$

$$\mathbf{w}_i = \sqrt{\frac{\nu}{u_i}} \cdot \mathbf{z}_i, \tag{3}$$

with $\mathbf{z}$ being the random 1D vector drawn from the multivariate normal distribution, where $\mathbf{\Sigma} = diag(\sigma_1^2, \ldots, \sigma_n^2)$ is the diagonal matrix of covariances for the hyperparameters (as uncorrelated random variables); u is the random variable from the uniform χ^2 distribution with ν degrees of freedom, and $\mathbf{w}$ is finally the random 1D vector from the multivariate Student's *t*-distribution with ν degrees of freedom. This phase of the algorithm (where $i \leq k$) is called the burn-in period, and it emphasizes the exploration of the search space (by using low values of the ν parameter). After the burn-in period, the samples for the subsequent steps ($i > k$) are drawn directly from the multivariate normal distribution, but (generally) with a lower variance, as follows:

$$\mathbf{w}_i \sim MVN(\mathbf{0}, \mathbf{\Sigma} \cdot f_s);\ for\ i > k, \tag{4}$$

where $0 < f_s \leq 1$ is the factor by which the initial standard deviations (that form the diagonal covariance matrix) are scaled. If there is a statistical correlation between hyperparameters, that information can be readily introduced through the covariance matrix itself.

Hence, for the *i*th iteration, a random walk takes the steps $\mathbf{x_{new}}_i = \mathbf{x_{old}}_i + \mathbf{w}_i$, where $\mathbf{w}$ comes from Equation (3) if $i \leq k$ or from Equation (4) if $i > k$; $\mathbf{x}$ is the 1D vector holding the (either new or old) coordinates of hyperparameter values inside the multidimensional search space. It can be seen that the burn-in phase (first k iterations of the algorithm) take longer steps (from the Student's *t*-distribution which has more mass in the tail than a normal distribution) and thus is able to explore more of the search space. After the burn-in period, the step size is reduced and, in addition, the starting position of vector x at the beginning of iteration $k + 1$ is taken from the best coordinates found during the burn-in period of the previous k iterations. This starts the exploitation phase. Furthermore, bounds can be imposed on hyperparameters within the search space, such that each variable x from the 1D vector $\mathbf{x}$ can have upper and lower bounds, i.e., $x \in [x_l, x_u]$, where x_l is the lower bound and x_u is the upper bound. If, during the random walk, a step takes any variable x outside its imposed bounds, that step is reversed as follows: $x_{new} = x_{old} - abs(w)/2$ if

$x_{new} > x_u$ and $x_{new} = x_{old} + abs(w)/2$ if $x_{new} < x_l$. In other words, if the step brings a variable above the upper bound, it is reversed to the left; alternatively, if the step brings it below the lower bound, it is reversed to the right.

The acceptance criterion is based on the Metropolis algorithm [31], where the difference between each two successive objective function's evaluations is computed as $\Delta E = f(\mathbf{x_{new}}) - f(\mathbf{x_{old}})$. If $\Delta E \leq 0$, the new solution is readily accepted. On the other hand, the case when $\Delta E > 0$ is treated as probabilistic. First, a random number is drawn from theuniform distribution $r \sim \mathcal{U}(0, 1)$. Then, a probability that the new solution is accepted is derived from the *Boltzman* probability distribution as follows:

$$p(\Delta E) = (2\pi T)^{-n/2} \cdot \exp(\frac{-|\Delta E|^2}{2T}), \tag{5}$$

where T is the current temperature. It can be seen that this probability is related to the temperature, and that it decreases as the system state cools down. Now, a new solution is accepted only if $r < p(\Delta E)$; otherwise, it is rejected. As can be seen, this last step is stochastic and allows us to accept inferior solutions at the beginning, when the temperature is still high, which helps to explore the search space. Later, when the temperature is decreased, the acceptance of inferior solutions is far less probable.

The temperature schedule (i.e., cooling) followed the exponential law, where the temperature for the *i*th iteration can be determined as follows:

$$T_i = T_0 \cdot \exp(-i/\kappa), \tag{6}$$

where T_0 is the initial temperature and κ is a constant that determines the cooling time. A total number of iterations is determined directly from the cooling schedule, i.e., the algorithm proceeds until the system has cooled down to a certain predetermined temperature.

After each iteration step, the accepted solution is compared to the best solution that has been found thus far and takes its place when having a lower value (i.e., is closer to the optimum). Furthermore, when $|\Delta E| < \epsilon$ holds, where ϵ is an arbitrary small number, the algorithm does not converge. This is monitored and if there is no progress after a certain number of iterations (i.e., waiting period), the algorithm is terminated under the early stopping criterion.

The complete simulated annealing algorithm, described in pseudocode, is depicted in Algorithm 1.

Algorithm 1 Simulated annealing algorithm with burn-in and early stopping.

```
input: k, $\mathbf{x_0}$, $[x_l, x_u]_{i=1,n}$, $\mathbf{\Sigma}$, $f_s$, $T_0$, $\kappa$, $\epsilon$, $\theta$, stop
$\mathbf{x} = \mathbf{x_0}$                                              ▷ initial values
$E = f(\mathbf{x_0})$
$\mathbf{x}_{best} = \mathbf{x}$
$\mathbf{E}_{best} = \mathbf{E}$
i = 0; wait = 0
while $T \geq \theta$ do
    if $i \leq k$ then                                                ▷ burn-in phase
        $\mathbf{z} \sim MVN(\mathbf{0}, \mathbf{\Sigma})$
        $u \sim \chi^2(\nu)$
        $\mathbf{w} = \sqrt{\frac{\nu}{u}} \cdot \mathbf{z}$
    else
        $\mathbf{w} \sim MVN(\mathbf{0}, \mathbf{\Sigma} \cdot f_s)$
    $\mathbf{x_{new}} \leftarrow \mathbf{x} + \mathbf{w}$                                  ▷ random walk
    for $j = 1, n$ do
        if $x_{new} < x_l$ then                                  ▷ below the lower bound
            $x_{new} \leftarrow x + abs(w)/2$
        else if $x_{new} > x_u$ then                             ▷ above the upper bound
            $x_{new} \leftarrow x - abs(w)/2$
    $E_{new} \leftarrow f(\mathbf{x_{new}})$
    $\Delta E = E_{new} - E$
    if $\Delta E \leq 0$ then
        $\mathbf{x} = \mathbf{x_{new}}$
        $E = E_{new}$
    else                                                    ▷ Metropolis acceptance
        $r \sim \mathcal{U}(0,1)$
        $p(\Delta E) = (2\pi T)^{-n/2} \cdot \exp(\frac{-|\Delta E|^2}{2T})$
        if $r < p(\Delta E)$ then
            $\mathbf{x} = \mathbf{x_{new}}$
            $E = E_{new}$
    T $\leftarrow T_0 \cdot \exp(-T/\kappa)$
    if $E < E_{best}$ then
        $\mathbf{x}_{best} = \mathbf{x_{new}}$
        $\mathbf{E}_{best} = \mathbf{E_{new}}$
    if $abs(\Delta E) \leq \epsilon$ then
        wait += 1
    if wait > stop then
        break
    i += 1
```

2.3. Embedding Models

The following embedding methods, as already mentioned, are briefly introduced hereafter: PCA, kPCA, tSVD, isomap embedding, LLE and its variants (modified and Hessian), spectral embedding, MDS, and *t*-SNE. Interested readers are at this point advised to consult [31] for more information and additional mathematical background on some of these different methods.

2.3.1. PCA, kPCA, and tSVD

Principal component analysis is a well-known method for decomposing a multivariate dataset into a set of orthogonal components (i.e., principal components). It employs a linear projection and relies on the SVD algorithm. Kernel PCA is a direct extension of PCA, which introduces a nonlinear dimensionality reduction through the use of kernels (i.e., the kernel trick); see [31] for more information. Different kernels can be applied, but we examined only the radial basis function (RBF) kernel. Truncated SVD is another method closely related to PCA. It uses, as the name suggests, a singular value decomposition of the feature matrix and retains only its largest singular values, thereby reducing the dimensionality of the feature space [31]. If k is the dimensionality of the projection, then a truncated SVD applied to feature matrix X produces a low-rank approximation $X \approx X_k = U_k \Sigma_k V_k^\top$, where Σ_k holds the top k singular values (from the main diagonal), while U_k and V_k hold, respectively, the left- and right-singular vectors.

2.3.2. Isomap Embedding

Isomap embedding can be seen as an extension of the kernel PCA. It seeks a lower-dimensional embedding that maintains geodesic distances between all data points, through the following three-step process [37]: (1) construct a neighborhood graph from distances between data points, (2) compute the shortest paths on the neighborhood graph, and (3) construct a low-dimensional embedding from the partial eigenvalue decomposition. The nearest search is based on the ball-tree algorithm [38]. The embedding is encoded in the eigenvectors corresponding to the largest eigenvalues of the isomap kernel.

2.3.3. Locally Linear Embedding

Locally linear embedding is yet another extension of PCA which seeks a lower-dimensional projection of the data that preserves distances within local neighborhoods. This method can be viewed as an application of a series of local PCAs, which are then globally compared to find the best nonlinear embedding [38]. This method also comprises a tree-step process as follows [39]: (1) assign a neighborhood to each data point based on local distances, (2) compute weights W_{ij} that best linearly reconstruct the data points from their neighbors, and (3) compute the low-dimensional embedding from these weights W_{ij}, by finding the smallest eigenmodes of the sparse symmetric matrix $M_{ij} = \delta_{ij} - W_{ij} - W_{ji} + \sum_k W_{ki} W_{kj}$, where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. Although it relies only on linear algebra, since the data points are reconstructed only from their neighbors, this method can still produce highly nonlinear embeddings. If one were to use multiple weight vectors in each neighborhood (instead of a single one), that would result in the so-called modified LLE method. Furthermore, if the locally linear structure were to be recovered by means of the Hessian-based quadratic form, that would result in the so-called Hessian eigenmapping. This is a variant of LLE also known as Hessian-based LLE; see [40] for more information.

2.3.4. Spectral Embedding

Spectral embedding with Laplacian eigenmaps is yet another nonlinear embedding method which is closely related to the locally linear embedding discussed above. It preserves local information and implicitly emphasizes natural clusters in the data. This method constructs a low-dimensional representation of the data using a spectral decomposition of the graph Laplacian. It also consists of three individual steps, as follows [41]: (1) construct the adjacency graph from nearest neighbors, (2) weight the graph edges by forming the matrix W, where $W_{ij} = 1$ if vertices i and j are connected by an edge and $W_{ij} = 0$ otherwise, and (3) compute eigenvalues and eigenvectors for the generalized eigenvalue problem of the form $Lf = \lambda D f$, where D is a diagonal weight matrix $D_{ii} = \sum_j W_{ji}$ and L is the Laplacian matrix $L = D - W$.

2.3.5. Multidimensional Scaling

The central concept in multidimensional scaling is the dissimilarity data δ_{ij} expressed through the distances in geometric spaces. The MDS algorithm minimizes the so-called *stress* function for the configuration of points $x_1, \ldots, x_n$ in t-dimensional space ($t < n$), which is described by the following root-mean-square residual:

$$S = \sqrt{\frac{\sum (d_{ij} - \hat{d}_{ij})^2}{\sum d_{ij}}}, \tag{7}$$

where d_{ij} are distances between points $x_1, \ldots, x_n$, and $\hat{d}_{ij}$ are those values that minimize S, subject to the constraint that $\hat{d}_{ij}$ have the same rank order as the dissimilarity measures δ_{ij}. The stress function is seen as a measure of how well the low-dimensional representation matches the data. The minimization of the multiparameter stress function can be tackled by means of the steepest decent or some other suitable optimization algorithm; see [42] for more information.

2.3.6. *t*-Distributed Stochastic Neighbor Embedding

The *t*-SNE method is based on converting the affinities of data points into probabilities, where the affinities in the original space are represented by Gaussian joint probabilities and the affinities in the embedded space are represented by the Student's *t*-distributions. The *t*-SNE method then minimizes the Kullback–Leibler (KL) divergence between a joint probability distribution, P, in the high-dimensional space and a joint probability distribution, Q, in the low-dimensional space [43]:

$$KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}, \tag{8}$$

with $p_{ij} = p_{ji}$, $q_{ij} = q_{ji}$, $\forall\, i, j$, and $p_{ii} = q_{ii} = 0$. The pairwise similarities, for the low-dimensional map q_{ij}, are given by:

$$q_{ij} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq l} \exp\left(-\|y_k - y_l\|^2\right)}, \tag{9}$$

and for a high-dimensional map, it is stated that $p_{ij} = (p_{i|j} + p_{j|i})/2n$, where $p_{i|j}$ is the conditional probability [43]

$$p_{i|j} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma^2\right)}{\sum_{k \neq l} \exp\left(-\|x_k - x_l\|^2 / 2\sigma^2\right)}, \tag{10}$$

scaled by the variance σ of the Gaussian that is centered on the data point x.

The KL divergence is minimized by means of the steepest descent algorithm [43]. The *t*-SNE focuses on the local structure of the data and tends to extract clustered local groups more than the other previously mentioned algorithms. This ability to group samples based on the local structure might be beneficial in some circumstances.

2.4. Model Training with Supervised and Unsupervised Learning

Producing a low-dimensional embedding from the high-dimensional data entails training the associated model by means of unsupervised learning. However, some models (such as the PCA) can be trained using supervised learning as well; see [31] for more information. In addition, a classifier can be trained, using supervised learning, on top of the embedding produced by any of the above-introduced (dimensionality reduction) models. This essentially establishes a pipeline which consists of the embedding model (as a transformer) and a classifier. We used a support vector machine (SVM) with an RBF kernel as a classifier. The entire process of model training (both supervised and unsupervised) is graphically represented in Figure 2.

As can be seen from Figure 2, the unsupervised learning loop uses only data features (matrix X), while the supervised learning loops additionally use targets (vector y) that are fed to the classifier. In addition, supervised learning can optimize the entire pipeline (outer loop), or only the classifier (inner loop). The hyperparameter optimization (both of the embedding method and of the SVM classifier) was carried out using the same simulated annealing algorithm described in Section 2.2. Only the objective function was different, as is explained next.

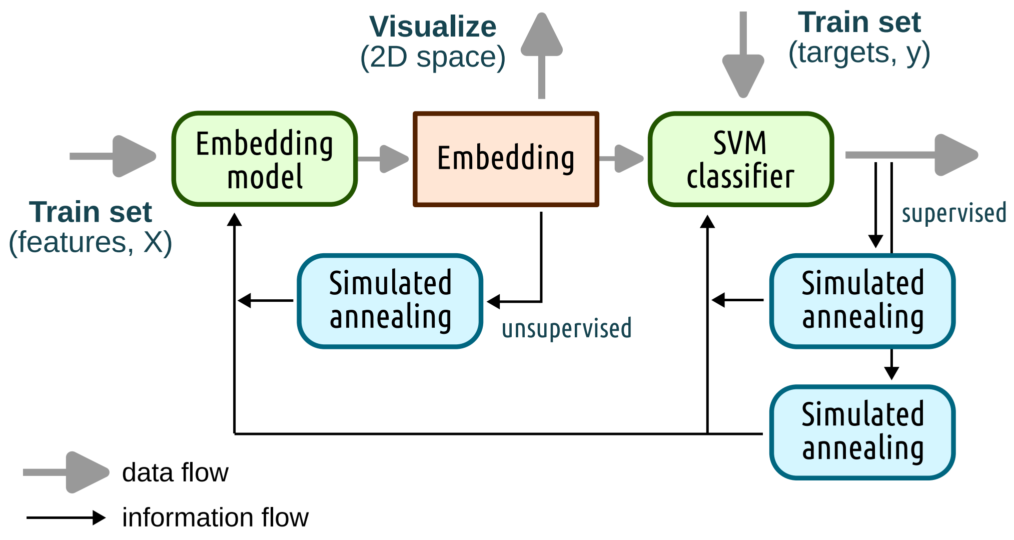


Figure 2. Training models using supervised and unsupervised learning.

2.4.1. Unsupervised Learning

Unsupervised learning used only the features of the dataset and no targets when optimizing the hyperparameters of the embedding models. We demonstrated this approach using two different embedding models: (a) kPCA and (b) *t*-SNE.

(a) The kPCA model has only one hyperparameter, the γ value of the underlying RBF kernel. The objective function here is the mean squared error (MSE). Namely, kPCA first constructs a low-dimensional embedding and then reconstructs the original high-dimensional data by the inverse transform. The MSE error between the original data and their reconstruction was minimized using the simulated annealing algorithm described in Section 2.2.

(b) The *t*-SNE model has one important hyperparameter, a perplexity (P) parameter [38]. The objective function here is based on the Kullback–Leibler divergence and can be described by the following relation [44]:

$$S(P) = 2 \cdot KL + \log(n) \cdot \frac{P}{n}, \tag{11}$$

where KL is the Kullback–Leibler divergence, P is the perplexity, and n is the number of samples. This function was minimized using the simulated annealing algorithm described in Section 2.2.

Unsupervised learning can be extended a step further by applying clustering on top of the low-dimensional embedding; see for example [45]. We used K-means clustering for separating the low-dimensional space of samples into stable and unstable cases [38]. Since we knew a priori that there are only two classes (stable and unstable), we assigned the K-means clustering with (hopefully) finding these clusters withing the embedding space.

2.4.2. Supervised Learning

Supervised learning used an SVM binary classifier with a radial basis function (RBF) kernel and class-weight balancing for classifying cases into stable and unstable ones. An SVM constructs a decision function of the form:

$$f(x) = \sum_{i \in SV} y_i \alpha_i \cdot K(x_i, x) + b, \tag{12}$$

where the summation of sample pairs (x_i, y_i) is over the space of the support vectors (SV), while $K(\cdot)$ is the kernel:

$$K(x, x') = \exp(-\gamma \|x - x'\|^2). \tag{13}$$

Unknown coefficients α_i and b are to be determined by solving the following optimization problem [31]:

$$\begin{aligned} &\min_{w,b,\zeta} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \zeta_i, \\ &\text{s.t.} \begin{cases} y_i(w^T\phi(x_i) + b) \geq 1 - \zeta_i, \\ \zeta_i \geq 0, \quad i = 1, \dots, n, \end{cases} \end{aligned} \tag{14}$$

where C is the penalty that acts as an inverse regularization parameter, while ζ_i is a slack variable, and n is again the number of samples. The penalty (C) and the RBF kernel coefficient (γ) are two crucial hyperparameters of the SVM that need to be optimized.

The supervised loop can be applied to the SVM classifier itself only or to the entire pipeline (see Figure 2). In case that it is applied to the classifier itself, there are two hyperparameters that need to be optimized. These are penalty $C \in (0.1 - 1000)$ and $\gamma \in (0.0001 - 1)$. In this case, the classifier "sits" on top of the embedding. The objective function here was the negative value of the cross-validated F_1-metric as a classification score. On the other hand, if the supervised loop is applied to the entire pipeline, it optimizes the hyperparameters of the embedding model in addition to those of the SVM, and it does that at the same time. We demonstrated this approach on a pipeline which consisted of kPCA as the embedding model (with an RBF kernel) and an SVM classifier (also with its own RBF kernel). In this case there were a total of three hyperparameters (one for kPCA and two for the SVM). Kernel PCA has its own γ parameter for the RBF kernel, that is independent of and unrelated to that of the SVM classifier. All three hyperparameters were optimized (by means of the simulated annealing) at the same time. The objective function was again the negative value of the cross-validated F_1-metric.

It is important to note that the hyperparameter search space during optimization (both supervised and unsupervised) is based on the common logarithm space (i.e., decadic logarithm). Namely, since the parameters can span large values, for example, $C \in (0.1 - 1000)$, in log-space, this transforms to $(10^{-1} - 10^3)$, which means searching within the interval $(-1, 3)$. Since this is a much narrower band (when seen in log-space), it significantly improves the convergence speed.

2.4.3. Scoring Models

The SVM binary classifier, trained with supervised learning on top of the embedding, was scored on the test set. The scoring metric is reported as the mean value with a standard deviation ($\mu \pm \sigma$), obtained from a 3-fold cross validation. The process of scoring the model is graphically visualized in Figure 3. It can be seen that the embedding model (as a transformer) and the SVM classifier constituted a pipeline, which was fed the test dataset along with the optimal hyperparameters obtained from the training phase. The classifier produced predictions, which were compared to the ground-truth values (i.e., test-set targets) and a certain metric was reported as the score (i.e., measure of test accuracy). Since the dataset was class-imbalanced, we used the F_1-score as the harmonic mean of the classifier's precision and recall, i.e., $F_1 = 2TP/(2TP + FP + FN)$, where TP is a true positive, FP is a false positive, and FN is a false negative outcome (from the confusion matrix of the classifier).

It can be seen that both the objective function and the scoring method were based on the same metric, which was not necessary but streamlined the comparison. This scoring process enabled us to compare different embedding models in terms of quality. Namely, the SVM classifier had a higher score if the embedding on which it was trained preserved more of the informational content.

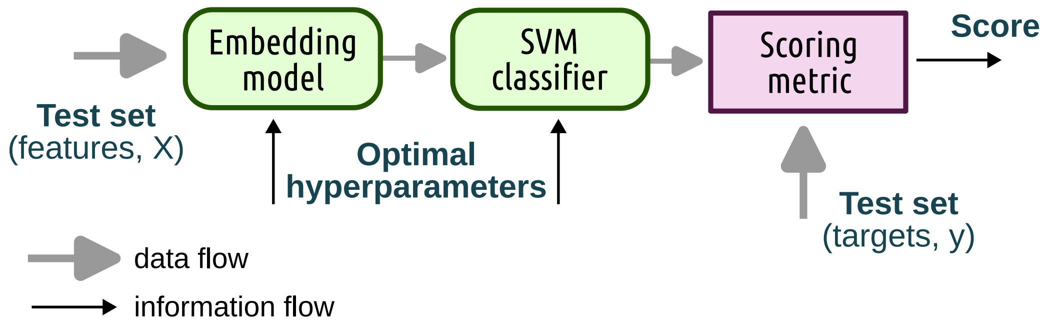


Figure 3. Scoring models on the test set.

3. Results

The previously discussed embedding models were applied to the IEEE New England 39-bus power system dataset (Section 2.1). This section presents the results and other findings which were derived from that application. In order to make the task more challenging for this small power system, each embedding model was required to transform the original high-dimensional space (with 350+ features) into a two-dimensional (2D) space, while preserving as much information as possible. Each embedding model was individually trained using the unsupervised learning approach (Section 2.4). A good embedding should feature only two (distinct) separable clusters, one (larger) for the stable and another (smaller) for the unstable cases. In order to further compare the quality of embeddings, the SVM classifier was individually trained (using the supervised training approach) on top of each of the embeddings. A classifier with a higher score indicated that the embedding model was able to preserve more of the information content and hence produced a better low-dimensional representation.

First, in order to demonstrate the convergence of our modified simulated annealing algorithm for hyperparameter tuning, as an example, Figure 4 presents the algorithm's convergence when optimizing the hyperparameters of the kPCA model using the unsupervised learning approach. It can be seen that the algorithm attained convergence before reaching 200 iterations.

The models produce embedding in a 2D space, which could be easily visualized and further inspected. Hence, Figure 5 demonstrates an example of a "good" embedding (from kPCA on the left side) and a "bad" embedding (from spectral embedding on the right side). It can be clearly seen that kPCA produced two distinct clusters, while spectral embedding was not able to provide a meaningful separation between stable and unstable cases. The color in Figure 5 was superimposed on the data points from the labels (after the fact) in order to visually emphasize stable and unstable cases (this information, however, was not available during the unsupervised training of these models).

Moreover, we found that from all the different embedding models that we tested on this dataset, only kPCA produced exactly two separable clusters. Furthermore, as can be seen from Figure 5 (left side), these clusters (roughly) coincided with stable and unstable classes. Other embedding models, such as the isomap embedding and MDS, preserved information that could be valuable to the classifier which could be trained on top of them but did not produce separable clusters (we found this to be the case with PCA and tSVD as well), or produced too many meaningless clusters (which was the case with *t*-SNE). For example, Figure 6 presents the embedding produced by *t*-SNE from unsupervised learning (unlabeled on the left and labeled on the right), where the color was added after the fact for visual emphasis. At the same time, Figure 7 presents the embedding produced by MDS, without labels (on the left) and with the addition of labels after the fact (on the right). It is clear from Figure 7 that this low-dimensional embedding produced a single (somewhat loose) cluster of data points; similar (but more dense) single clusters resulted from applying PCA and tSVD as well. However, if labels were present in the data, then these embeddings would still preserve enough information for training the classifier on top of them (with supervised learning or even semisupervised learning).

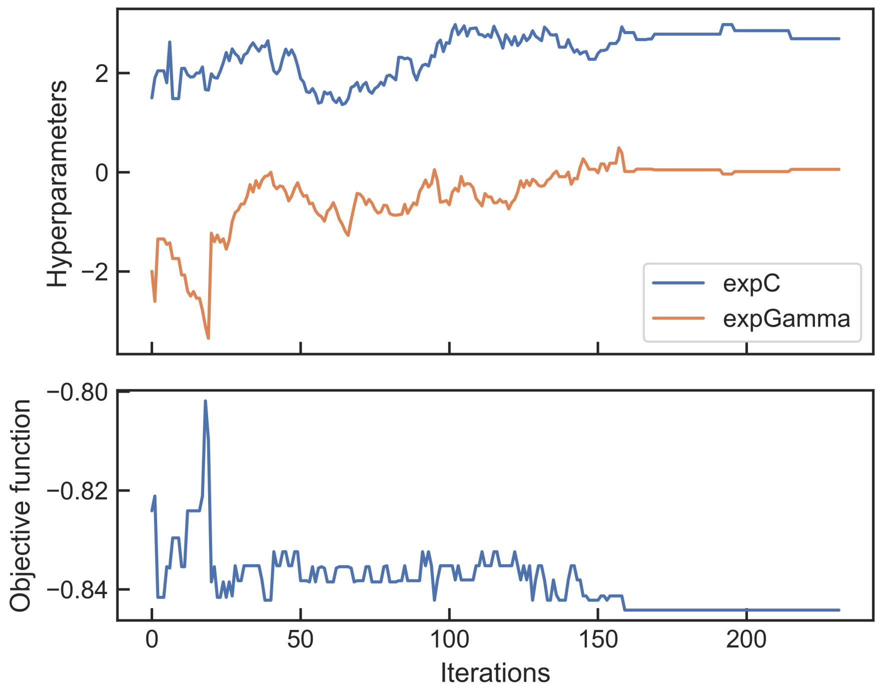


Figure 4. Convergence of the simulated annealing algorithm for optimizing hyperparameters of the kPCA model using the unsupervised learning approach.

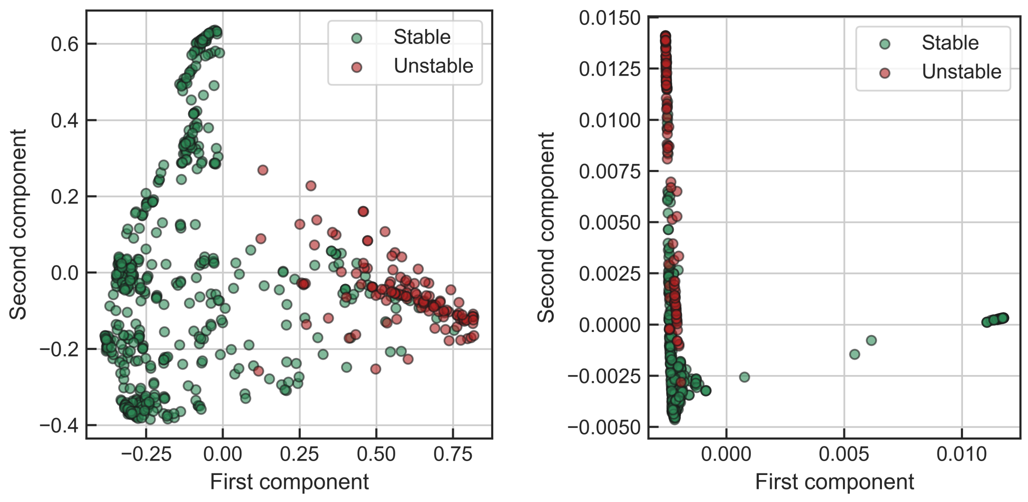


Figure 5. Embedding produced by kPCA (**left**) and spectral embedding (**right**), from the unsupervised learning approach (color added after the fact for visual emphasis).

In addition, when the kPCA model was trained together with the SVM classifier using the pipeline approach (i.e., using the outer optimization loop from Figure 2), we found that this prioritized the classifier's accuracy (as would be expected), and that the resulting embedding occasionally did not feature two clearly distinct clusters. An example of this outcome is depicted in Figure 8, where we show labeled data on the left and unlabeled data on the right. Of course, labels must be known a priori in order to apply this (supervised learning) approach.

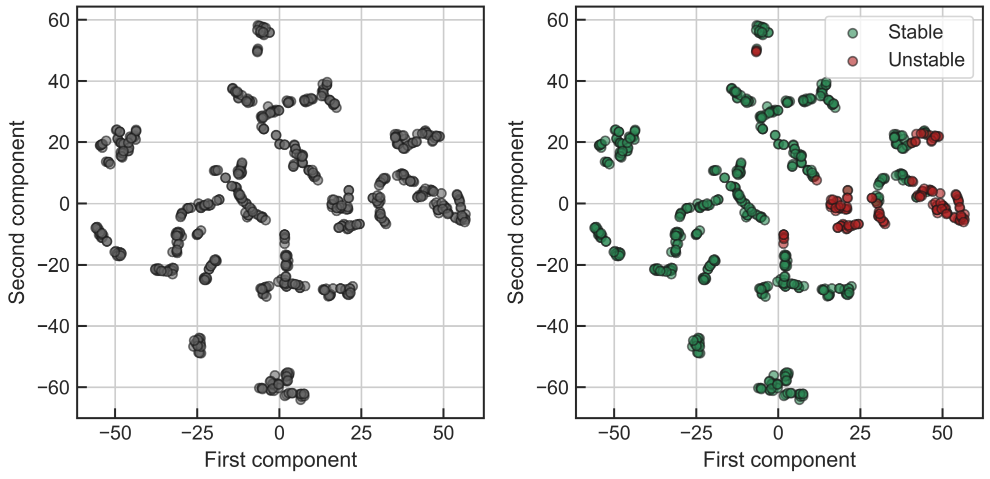


Figure 6. Embedding produced by *t*-SNE from the unsupervised learning approach without labels (**left**) and with the addition of labels after the fact (**right**).

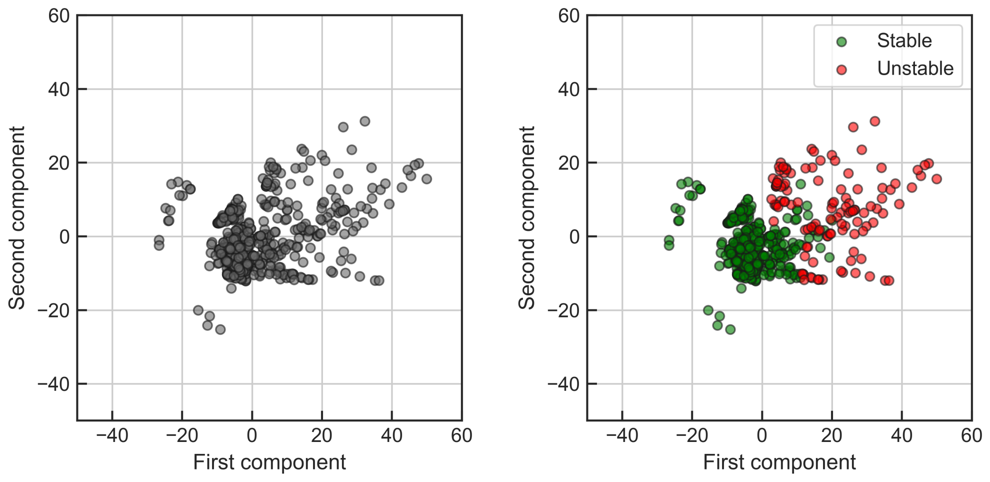
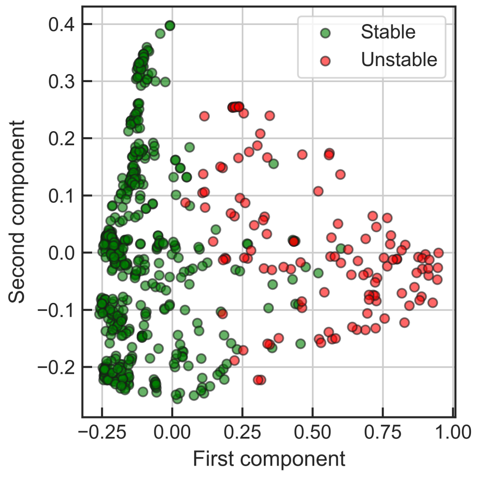


Figure 7. Embedding produced by MDS, from the unsupervised learning approach, without labels (**left**) and with the addition of labels after the fact (**right**).

Furthermore, interestingly, we found that locally linear embedding (LLE), including all of the variants that we tested (i.e., modified and Hessian), produced a 2D embedding that had very low informational content. Hence, classifiers trained on top of these embeddings had low accuracy, as shown later. Moreover, we found that the embedding produced by LLE (and its variants) was very sensitive to the number of neighbors used for producing it. In order to demonstrate these effects, Figure 9 provides the embedding computed by LLE with 10 (left) and 20 (right) neighbors. This high sensitivity was not found with other embedding methods and makes LLE less useful for this particular application.



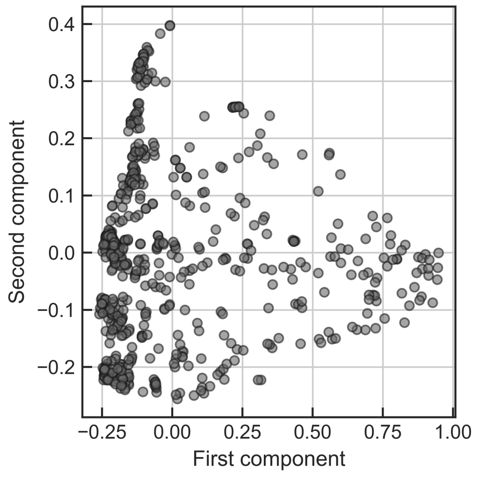
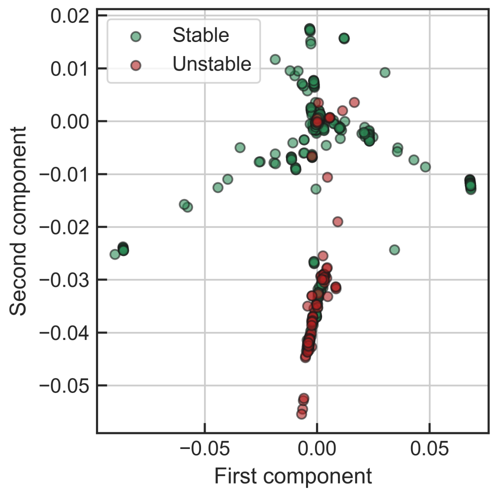


Figure 8. Example of the embedding produced by the pipeline composed of kPCA and SVM with supervised training; labeled data (**left**) and unlabeled data (**right**).



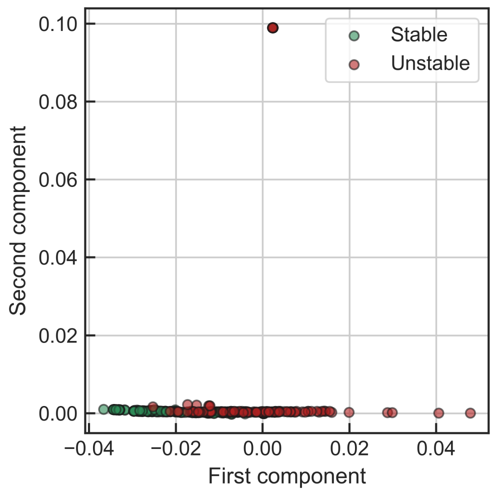


Figure 9. Embedding produced by locally linear embedding with 10 (**left**) and 20 (**right**) neighbors.

Finally, an SVM classifier was independently trained by means of supervised learning (i.e., using the inner optimization loop from Figure 2) on top of each embedding, and its classification metric (i.e., the threefold cross-validated F_1-score) is reported, along with its optimal hyperparameters (C, γ), in Table 2. The last column shows the number of iterations of the simulating annealing algorithm, along with a Y/N indicator that reports if early stopping was triggered. It should be mentioned that the SVM classifier was not calibrated.

In a fully unsupervised learning approach, one would take the unlabeled data and train the embedding model (which hopefully would produce only two clusters), which could then be discovered by the appropriate clustering algorithm (as another unsupervised learning method). We demonstrate this approach here on the embedding produced by the kPCA method trained using the unsupervised learning; see Figure 4 (left). The K-means clustering was applied on top of that embedding. The produced clusters are graphically presented in Figure 10 (left), where the color indicates the cluster membership of each data point. If this is compared to Figure 4 (left), it can be seen that many of the points from the clusters also belong to the actual classes, which means that a completely unsupervised learning approach (kPCA plus K-means) is able to detect power system disturbances from the WAMS-PMU data. Furthermore, Figure 10 (on the right) presents a decision boundary from the SVM classifier trained on top that embedding with a supervised learning approach.

An overlap between cluster and classifier boundaries can be discerned from Figure 10 by comparing the left- and right-hand sides, corroborating the fact that clustering on top of the kPCA embedding can separate power system TSA cases and identify unstable ones.

Table 2. The SVM classifier's parameters and scores after being applied to each of the embeddings.

Embedding Model	SVM (C, γ)		F_1-Score $(\mu \pm \sigma)$	No. Iters.
PCA	1.93	0.0064	0.8521 ± 0.0121	75 (Y)
kPCA	98.5	0.2610	0.8456 ± 0.0223	167 (Y)
tSVD	0.43	0.0118	0.8517 ± 0.0058	43 (Y)
t-SNE	4.51	0.0496	0.9316 ± 0.0301	232 (N)
MDS	14.95	0.0443	0.9281 ± 0.0133	67 (Y)
kPCA (pipeline)	398.1	6.5463	0.9114 ± 0.0226	232 (N)
LLE (default)	67.1	8.8491	0.7307 ± 0.0301	64 (Y)
LLE (modified)	147.7	1.2313	0.7352 ± 0.0297	39 (Y)
Hessian eigen.	212.6	0.0184	0.8213 ± 0.0406	37 (Y)
Spectral embed.	405.5	0.0733	0.7541 ± 0.0346	99 (Y)
Isomap embed.	5.14	0.0036	0.8984 ± 0.0391	65 (Y)

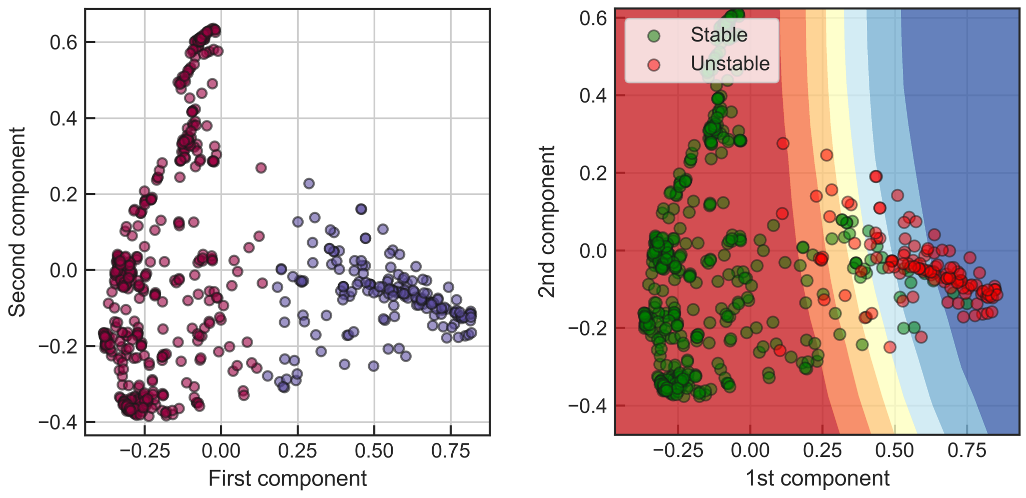


Figure 10. K-means clusters (**left**) and SVM decision boundary (**right**) on top of the embedding produced by kPCA with unsupervised learning.

4. Discussion

We compared our proposed simulated annealing approach to a hyperparameter optimization with a random search approach [38] and found that it produced better results with the same scoring method, while (generally) using fewer iterations. We further found that the random search was less consistent since it exhibited a larger deviation of the cross-validated scores. This is not completely unexpected, since a random search simply draws samples from (in our case exponential) distributions and has no overarching guiding principle in finding the objective function's optimum value. On the other hand, our simulated annealing balances exploration with exploitation through the use of burn-in and cooling schedule and avoids overfitting by means of the early stopping. Also, early stopping enables the simulated annealing to stop searching before all iterations have been exhausted, which is not possible with a random search [38]. However, the random search algorithm, unlike simulated annealing, can be run in parallel on multicore architectures, which helps reduce its execution time.

As far as different embedding methods are concerned, we found that the isomap embedding and MDS provided good bases for supervised classification (which can be seen from Table 2) but, unfortunately, could not produce separable clusters. Hence, they are not well suited for a fully unsupervised learning approach with this kind of data. The same can be said about the PCA and tSVD methods, which also could not produce separable clusters. Furthermore, although an SVM trained on top of the *t*-SNE embedding had a high score (as evident from Table 2), it suffered from a similar deficiency, which emanated from the fact that that embedding contained a lot of small and artificial clusters (see Figure 6). Hence, the *t*-SNE method could not be combined with clustering in a fully unsupervised learning approach, in contrast to what was proposed for some other datasets, e.g., see [45]. This finding is characteristic for the studied dataset and may be due to its peculiar information content (time instants of phase voltages, currents, active and reactive powers, etc.). We also found that only kPCA, when applied to this dataset, produced an embedding that could be utilized in a fully unsupervised manner, as evidenced from Figure 10 (left side).

It is important to mention that there are several sources of randomness that need to be considered here: in the dataset itself (statistical sampling), in the way it was split into training and test sets, in the embedding process (finding nearest neighbors, etc.), and in the simulated annealing (which rests on a random walk). All of these aspects contribute to the variability in the classifier scores between different runs. However, we found that there was a persistence where some embedding methods, when combined with the SVM, would reliably (and consistently) produce better results. These were the PCA, kPCA, and MDS methods (even their 2D embeddings were consistent between runs, despite different data splits). This is a reassuring finding. Also, the tSVD often produced embedding (almost) identical to that of the PCA, which is not surprising considering their similar backgrounds. On the other hand, we found that LLE (and its variants) was quite brittle, sensitive to inputs, and produced embeddings with a low informational content. Finally, although *t*-SNE often produced embeddings that happened to be fertile ground for the SVM training, it was not consistent between runs and could not produce meaningful (nor separable) clusters.

Also, it can be stated that the classifier scores improved across the board when more than two dimensions were retained for the low-dimensional embedding, which was something to be expected. For example, PCA with only seven principal components was able to explain 90% of the variance in the dataset. Classifiers trained on embeddings with more than two dimensions generally produce F_1-scores above the 0.9 level. However, if the goal is to use unsupervised (or semisupervised) learning, then two- and three-dimensional embeddings are particularly appealing, since results of applied clustering can be easily visualized, inspected, and explained (i.e., retaining the properties of the white-box model). We showed that kPCA with K-means clustering produced satisfactory results on the IEEE New England 39-bus dataset. We also found that contrary to the findings on other datasets [45], *t*-SNE was not able to produce meaningful clusters. Neither could the other embedding models that we tested.

In addition, we also tested an LLE variant that implemented a local tangent space alignment algorithm [38]. This approach proved no better than any other LLE variant. Furthermore, it can be mentioned in passing that we also examined a stacked autoencoder for embedding this same dataset and found that it could not consistently produce separable clusters [33]. However, we did not carry out exhaustive tests of different autoencoder architectures and hence, our conclusions are limited by these facts. Finally, there are still other embedding methods, but due to the limited space, they could not be tested here.

5. Conclusions

This paper introduced manifold learning to the electric power system transient stability analysis, based on the dataset derived from extensive transient simulations of the IEEE New England 39-bus benchmark power system. A majority of published papers that applied machine learning approaches to the power system TSA problem were based on supervised learning, with very few exceptions. This paper, however, explored supervised and unsupervised learning for the power system TSA problem. It compared and contrasted different embedding methods. The main contributions of this paper to the state of the art can be summarized through the following findings: (1) kPCA is able to produce a two-dimensional embedding with clearly separable clusters of stable and unstable cases; (2) K-means clustering can be linked with kPCA for a fully unsupervised learning approach; (3) several embedding methods (e.g., multidimensional scaling, isomap embedding) produce a low-dimensional embedding which preserves information without finding clearly separable clusters; (4) an SVM classifier can attain a high accuracy when trained on top of these low-dimensional embeddings; (5) *t*-SNE embedding could not produce meaningful clusters; and (6) simulated annealing can be used for hyperparameter optimization and provides better results than a random search approach.

It should be mentioned that some of these conclusions may be restricted to the examined dataset, although some findings reported here (e.g., the SVM classifier performance) have been corroborated by other researchers. Certainly, further research is needed to independently verify and extrapolate these findings on other power system datasets. Furthermore, additional research is needed to stress-test these findings in an environment that is not a benchmark test power system. This would include introducing various levels of noise into the dataset, along with missing values, measurement errors, etc., that would corrupt the data in different ways. Also, the resilience of different embeddings to the data drift could be examined. These are some future research directions that we can foresee at the present time.

Author Contributions: Conceptualization, P.S. and D.L.; methodology, P.S.; software, P.S. and D.L.; validation, P.S. and D.L.; investigation, D.L.; resources, P.S. and D.L.; data curation, P.S.; writing—original draft preparation, P.S.; writing—review and editing, D.L.; visualization, P.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The IEEE New England 39-bust test power system transient dataset with 350+ features has been deposited on Zenodo under the CC BY license, accessed on 1 October 2023 (www.zenodo.org/record/7350829), https://doi.org/10.5281/zenodo.7350828.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IBR	Inverter-based resource
TSA	Transient stability analysis
TSI	Transient stability index
ML	Machine learning
KL	Kullback–Leibler divergence
MVN	Multivariate normal distribution
ANN	Artificial neural network
PMU	Phasor measurement unit
WAMS	Wide-area monitoring system
PCA	Principal component analysis
kPCA	Kernel principal component analysis
tSVD	Truncated singular value decomposition
LLE	Locally linear embedding
MDS	Multidimensional scaling
t-SNE	t-Distributed stochastic neighbor embedding
SVM	Support vector machine
RBF	Radial basis-function kernel
MSE	Mean squared error
FACTS	Flexible AC transmission system

References

1. NERC. *Reliability Guideline: Performance, Modeling, and Simulations of BPS-Connected Battery Energy Storage Systems and Hybrid Power Plants*; North American Electric Reliability Corporation: Atlanta, GA, USA, 2021.
2. Eto, J.H.; Undrill, J.; Roberts, C.; Mackin, P.; Ellis, J. *Frequency Control Requirements for Reliable Interconnection Frequency Response*; Lawrence Berkeley National Laboratory: Berkeley CA, USA, 2018. Available online: https://eta-publications.lbl.gov/sites/default/files/frequency_control_requirements_lbnl-2001103.pdf (accessed on 1 October 2023).
3. Zhang, Y.; Liu, W.; Wang, F.; Zhang, Y.; Li, Y. Reactive Power Control Method for Enhancing the Transient Stability Total Transfer Capability of Transmission Lines for a System with Large-Scale Renewable Energy Sources. *Energies* **2020**, *13*, 3154. [CrossRef]
4. Phadke, A.G.; Bi, T. Phasor measurement units, WAMS, and their applications in protection and control of power systems. *J. Mod. Power Syst. Clean Energy* **2018**, *6*, 619–629. [CrossRef]
5. Oladeji, I.; Zamora, R.; Lie, T.T. An Online Security Prediction and Control Framework for Modern Power Grids. *Energies* **2021**, *14*, 6639. [CrossRef]
6. Li, Y.; Xia, Y.; Ma, J.; Peng, Y. Transient Stability Analysis for a Multi-VSC Parallel System Based on the CFND Method. *Energies* **2023**, *16*, 7243. [CrossRef]
7. Zhang, S.; Zhu, Z.; Li, Y. A Critical Review of Data-Driven Transient Stability Assessment of Power Systems: Principles, Prospects and Challenges. *Energies* **2021**, *14*, 7238. [CrossRef]
8. Alimi, O.A.; Ouahada, K.; Abu-Mahfouz, A.M. A Review of Machine Learning Approaches to Power System Security and Stability. *IEEE Access* **2020**, *8*, 113512–113531. [CrossRef]
9. Tan, B.; Yang, J.; Tang, Y.; Jiang, S.; Xie, P.; Yuan, W. A Deep Imbalanced Learning Framework for Transient Stability Assessment of Power System. *IEEE Access* **2019**, *7*, 81759–81769. [CrossRef]
10. Liu, X.; Zhang, X.; Chen, L.; Xu, F.; Feng, C. Data-driven Transient Stability Assessment Model Considering Network Topology Changes via Mahalanobis Kernel Regression and Ensemble Learning. *J. Mod. Power Syst. Clean Energy* **2020**, *8*, 1080–1091. [CrossRef]
11. Liu, X.; Min, Y.; Chen, L.; Zhang, X.; Feng, C. Data-driven Transient Stability Assessment Based on Kernel Regression and Distance Metric Learning. *J. Mod. Power Syst. Clean Energy* **2021**, *9*, 27–36. [CrossRef]
12. Su, T.; Liu, Y.; Zhao, J.; Liu, J. Probabilistic Stacked Denoising Autoencoder for Power System Transient Stability Prediction with Wind Farms. *IEEE Trans. Power Syst.* **2021**, *36*, 3786–3789. [CrossRef]
13. Tian, Y.; Wang, K.; Oluic, M.; Ghandhari, M.; Xu, J.; Li, G. Construction of Multi-State Transient Stability Boundary Based on Broad Learning. *IEEE Trans. Power Syst.* **2021**, *36*, 2906–2917. [CrossRef]
14. Ren, C.; Xu, Y. A Fully Data-Driven Method Based on Generative Adversarial Networks for Power System Dynamic Security Assessment with Missing Data. *IEEE Trans. Power Syst.* **2019**, *34*, 5044–5052. [CrossRef]
15. Ren, J.; Chen, J.; Li, B.; Zhao, M.; Shi, H.; You, H. A Method for Power System Transient Stability Assessment Based on Transfer Learning. In Proceedings of the 2020 IEEE Power Energy Society General Meeting (PESGM), Montreal, QUE, Canada, 2–6 August 2020; pp. 1–5. [CrossRef]
16. Li, X.; Yang, Z.; Guo, P.; Cheng, J. An Intelligent Transient Stability Assessment Framework with Continual Learning Ability. *IEEE Trans. Ind. Inform.* **2021**, *17*, 8131–8141. [CrossRef]
17. Mukherjee, R.; De, A. Development of an Ensemble Decision Tree-Based Power System Dynamic Security State Predictor. *IEEE Syst. J.* **2020**, *14*, 3836–3843. [CrossRef]
18. Behdadnia, T.; Yaslan, Y.; Genc, I. A new method of decision tree based transient stability assessment using hybrid simulation for real-time PMU measurements. *IET Gener. Transm. Distrib.* **2021**, *15*, 678–693. [CrossRef]
19. Mi, D.; Wang, T.; Gao, M.; Li, C.; Wang, Z. Online transient stability margin prediction of power systems with wind farms using ensemble regression trees. *Int. Trans. Electr. Energy Syst.* **2021**, *31*, e13057. [CrossRef]
20. Shao, D.; Xu, Y.; Ma, S.; Jing, Y.; Sun, W.; Zhou, T.; Yang, J. Transient Stability Assessment Method for Power System Based on SVM with Adaptive Parameters Adjustment. In Proceedings of the 2021 IEEE 4th International Electrical and Energy Conference, CIEEC 2021, Wuhan, China, 28–30 May 2021. [CrossRef]
21. Zhou, Y.; Wu, J.; Yu, Z.; Ji, L.; Hao, L. A Hierarchical Method for Transient Stability Prediction of Power Systems Using the Confidence of a SVM-Based Ensemble Classifier. *Energies* **2016**, *9*, 778. [CrossRef]
22. Hu, W.; Lu, Z.; Wu, S.; Zhang, W.; Dong, Y.; Yu, R.; Liu, B. Real-time transient stability assessment in power system based on improved SVM. *J. Mod. Power Syst. Clean Energy* **2019**, *7*, 26–37. [CrossRef]
23. Bashiri Mosavi, A.; Amiri, A.; Hosseini, H. A Learning Framework for Size and Type Independent Transient Stability Prediction of Power System Using Twin Convolutional Support Vector Machine. *IEEE Access* **2018**, *6*, 69937–69947. [CrossRef]
24. Zhu, Q.; Chen, J.; Zhu, L.; Shi, D.; Bai, X.; Duan, X.; Liu, Y. A Deep End-to-End Model for Transient Stability Assessment with PMU Data. *IEEE Access* **2018**, *6*, 65474–65487. [CrossRef]
25. Chen, K. Indirect PCA Dimensionality Reduction Based Machine Learning Algorithms for Power System Transient Stability Assessment. In Proceedings of the 2019 IEEE Innovative Smart Grid Technologies—Asia (ISGT Asia), Chengdu, China, 21–24 May 2019; pp. 4175–4179. [CrossRef]
26. Bellizio, F.; Cremer, J.; Sun, M.; Strbac, G. A causality based feature selection approach for data-driven dynamic security assessment. *Electr. Power Syst. Res.* **2021**, *201*, 107537. [CrossRef]

27. Gnanavignesh, R.; Shenoy, U. An improved relaxation based spatial domain decomposition method for parallel transient stability simulation. *Int. J. Electr. Power Energy Syst.* **2022**, *135*, 107561. [CrossRef]
28. Arvizu, C.M.C.; Messina, A.R. Dimensionality Reduction in Transient Simulations: A Diffusion Maps Approach. *IEEE Trans. Power Deliv.* **2016**, *31*, 2379–2389. [CrossRef]
29. Biswal, M.; Hao, Y.; Chen, P.; Brahma, S.; Cao, H.; De Leon, P. Signal features for classification of power system disturbances using PMU data. In Proceedings of the 19th Power Systems Computation Conference, PSCC 2016, Genoa, Italy, 19–23 June 2016. [CrossRef]
30. Hang, F.; Huang, S.; Chen, Y.; Mei, S. Power system transient stability assessment based on dimension reduction and cost-sensitive ensemble learning. In Proceedings of the 2017 IEEE Conference on Energy Internet and Energy System Integration, EI2 2017-Proceedings, Beijing, China, 26–28 November 2017; pp. 1–6. [CrossRef]
31. Murphy, K.P. *Machine Learning: A Probabilistic Perspective*, 4th ed.; MIT Press: Cambridge, MA, USA, 2013.
32. Moeini, A.; Kamwa, I.; Brunelle, P.; Sybille, G. Open data IEEE test systems implemented in SimPowerSystems for education and research in power grid dynamics and control. In Proceedings of the Universities Power Engineering Conference, Stoke-on-Trent, UK, 1–4 September 2015; pp. 3–8. [CrossRef]
33. Sarajcev, P.; Kunac, A.; Petrovic, G.; Despalatovic, M. Power system transient stability assessment using stacked autoencoder and voting ensemble. *Energies* **2021**, *14*, 3148. [CrossRef]
34. Sarajcev, P.; Kunac, A.; Petrovic, G.; Despalatovic, M. Power System Transient Stability Assessment Simulations Dataset—IEEE New England 39-Bus Test Case. 2021. Available online: https://zenodo.org/records/4521886 (accessed on 1 October 2023). [CrossRef]
35. Sarajcev, P.; Kunac, A.; Petrovic, G.; Despalatovic, M. IEEE New England 39-Bus Test Case: Dataset for the Transient Stability Assessment. 2021. Available online: https://zenodo.org/records/7350829 (accessed on 1 October 2023). [CrossRef]
36. Kirkpatrick, S.; Gelatt, C.D., Jr.; Vecchi, M.P. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671–680. [CrossRef] [PubMed]
37. Tenenbaum, J.B.; de Silva, V.; Langford, J.C. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science* **2000**, *290*, 2319–2323. [CrossRef] [PubMed]
38. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
39. Roweis, S.T.; Saul, L.K. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science* **2000**, *290*, 2323–2326. [CrossRef]
40. Donoho, D.L.; Grimes, C. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proc. Natl. Acad. Sci. USA* **2003**, *100*, 5591–5596. [CrossRef]
41. Belkin, M.; Niyogi, P. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Comput.* **2003**, *15*, 1373–1396. [CrossRef]
42. Kruskal, J.B. Nonmetric multidimensional scaling: A numerical method. *Psychometrika* **1964**, *29*, 115–129. [CrossRef]
43. van der Maaten, L.; Hinton, G. Visualizing Data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
44. Cao, Y.; Wang, L. Automatic Selection of t-SNE Perplexity. *arXiv* **2017**, arXiv:1708.03229.
45. Liu, L.P.; Wang, Q.; Dong, M.N.; Zhang, Z.Y.; Li, Y.; Wang, Z.Z.; Wang, S.Q. Application of K-Means ++ Algorithm Based on t-SNE Dimension Reduction in Transformer District Clustering. In Proceedings of the 2020 Asia Energy and Electrical Engineering Symposium (AEEES), Chengdu, China, 28–31 May 2020; pp. 74–78. [CrossRef]