Guided Research

Local Learning Clustering Algorithm for Crash Simulation Data with a Support Vector Machine as Out-of-Sample Extension

Author: Julius Adorf[1,2],
Supervisor: Benjamin Peherstorfer[1] and
Supervisor: Josephine Sullivan[2]

[1]Technische Universität München
[2]Royal Institute of Technology, Stockholm

May 18, 2013

Abstract

This work explores a local-learning based approach for the clustering of more than 6000 finite element nodes of a car crash simulation. The goal is to group those nodes together that show similar bending behavior. Such grouping is required as a part of a larger analysis workflow described in other work. We combine the existing local-learning based clustering algorithm (LLCA) with a SVM-based out-of-sample extension for tractability. We investigate into the expected density measure as a model selection criterion. Finally, we show that an artificial moon-shaped dataset can be separated successfully with our method, and that the the results for the finite element model appear reasonable. The contributions of this work are: an existing algorithms is tested on new datasets, the algorithm is made available as open source, and a sparse-grid variant of LLCA is discussed.

Contents

1 Introduction

Clustering is an unsupervised learning method where data is to be partitioned into separate clusters. In this technical report, we have both an artificial dataset in the shape of two moons and a more realistic dataset that originates from car crash simulations, as described in [1]. In order to cluster these datasets, we use the *local learning based clustering algorithm (LLCA)* [2]. The main objective is to find out how suitable this algorithm is for clustering the particular datasets. The datasets are described in Section 2.

The fundamental idea behind LLCA is the application of supervised learning methods in an unsupervised context. Concretely, LLCA searches for a clustering where a local classifier trained on the cluster

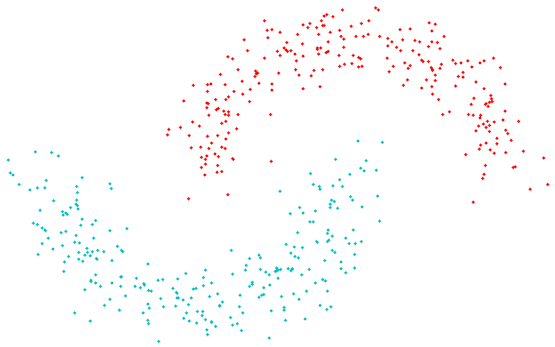

Figure 1: The TWO-MOONS dataset with ground truth labeling. There is no straight line that separates the two moons.

labels of neighbouring data has a low error in predicting the cluster labels [2]. Section 3 discusses the theoretical foundation of LLCA.

Originally, LLCA uses *kernel ridge regression (KRR)* local classification. KRR is a kernelized variant of least-squares regression with a regularization term. LLCA involves training a classifier for each data point. To reduce computational costs, the classifier is trained locally using only the k nearest neighbours. Section 4 discusses LLCA with KRR, and its results on the datasets.

We can obtain a variant of the original LLCA through replacing kernel ridge regression by *classification on sparse grids* [3]. In contrast to the former method, the basis functions of sparse-grid classifiers are not attached to the data points and it is thus no longer necessary to consider the nearest neighbours. Sparse grids have been successfully applied to other problems. It is therefore interesting to see whether the sparse grid methods also work well when plugged into the LLCA framework. Unfortunately, the attempt failed. This is worth reporting, and Section 5 discusses LLCA with sparse grid classifiers.

2 Datasets

2.1 Two moons dataset

The two-moons dataset (TWO-MOONS) is depicted in Figure 1. It is a small, artificial dataset consisting of 500 points, divided into two apparent moon-shaped clusters. The data points have only two features, which makes visual inspection very easy. It is worth noting that there is no line that can separate the two moons. For this dataset: $n = 500$, $d = 2$, $c = 2$.

2.2 Car crash simulation dataset

The second, more realistic dataset (TRUCK-PARTS) originates from a car crash simulation. The dataset has been provided by the Chair of Scientific Computing at Technische Universität München and records the bending behavior of the four longitudinal chassis beams (Figure 2, Figure 3) in the front of a Chevrolet C2500 pick-up truck [1]. The beams are represented by 6674 finite element nodes, and the bending behavior of these beams between two time steps is described by 126 features. Each feature corresponds to the displacement of a node in one of the 126 simulation runs that generated the data.

This work presents a non-linear method that groups the finite element nodes of the truck parts into clusters of nodes with similar bending behavior. This clustering is required in the analysis workflow in [1].

3 Foundations of LLCA

This section summarizes the concepts of LLCA, and introduces the common notation and common theory for both Section 4 and Section 5. A more complete discussion can be found in the original paper [2].

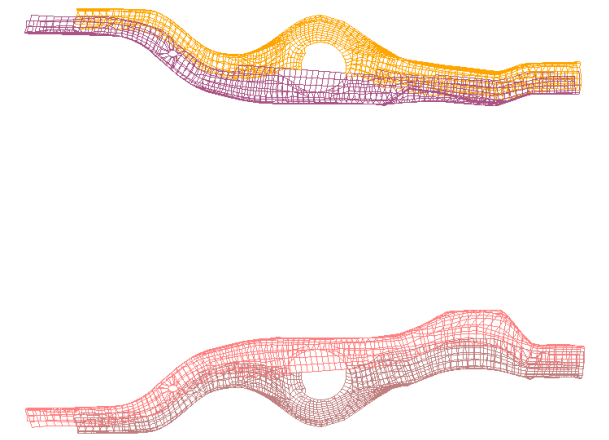

Figure 2: Top view on the finite element models of the four chassis beams of a pick-up truck, displayed as wireframes. Each chassis beam is shown in another color. Different parts of one beam might bend differently whilst parts of different beams might show the same bending behavior.

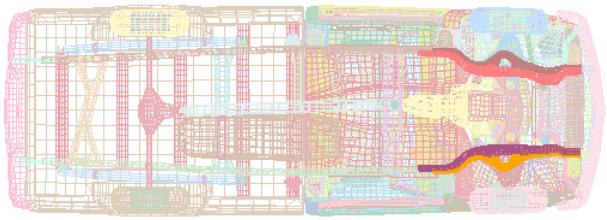

Figure 3: Bottom view of the finite element model of a pick-up truck. The four longitudinal chassis beams shown in the right of the figure need to be partitioned into regions with similar bending behavior in a series of crash simulations.

3.1 Problem formulation

The clustering problem is stated as follows: We have a dataset $X = \left\{\mathbf{x}_i : \mathbf{x}_i \in \mathbb{R}^d\right\}_{i=1\ldots n}$, consisting of n data points. Each data point is described by d features. The objective is to partition the dataset into c clusters.

Following [2], the clustering result can be expressed as a *partition matrix* $\mathbf{P} \in \{0,1\}^{n\times c}$, where $\mathbf{P}_{il} = 1$ if and only if the i-th data point belongs to the l-th cluster. In other words, the l-th column of the partition matrix is the *cluster indicator* for the l-th cluster.

We additionally define the *scaled partition matrix* $\mathbf{F} \in \mathbb{R}^{n\times c}$. The l-th column $\mathbf{f}^l$ of $\mathbf{F}$ is a *scaled cluster indicator*, and is formed by dividing the l-th cluster indicators by the square root of the size of the corresponding cluster. We can easily convert back and forth between $\mathbf{F}$ and $\mathbf{P}$. Clustering the data is equivalent to finding a scaled partition matrix. The detailed reason why we work with the scaled partition matrix is given in [2]. Here is an example partition matrix and the corresponding scaled partition matrix:

$$\mathbf{P} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \mathbf{F} = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{3}} \\ 0 & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{3}} \end{pmatrix}$$

3.2 Basic principle of LLCA

LLCA tries to find a clustering where the cluster label of a data point can be well-predicted with a classifier trained on the remaining data and cluster labels.

Formally, given the cluster labels $\mathbf{f}^l$, we construct classifiers $\mathbf{o}_i^l : \mathbb{R}^d \to \mathbb{R}$ such that $\mathbf{o}_i^l(\mathbf{x}_i)$ is the prediction of the scaled cluster label $\mathbf{f}_i^l$. Then we formulate an optimization problem [2], where the distance between the scaled cluster indicators $\mathbf{f}^l$ and classifier outputs $\mathbf{o}^l$ shall be minimized:

$$\min_{\mathbf{f}^1 \ldots \mathbf{f}^c} \sum_{l=1}^{c} \sum_{i=1}^{n} ||\mathbf{f}_i^l - \mathbf{o}_i^l(\mathbf{x}_i)||^2 \quad = \quad \min_{\mathbf{f}^1 \ldots \mathbf{f}^c} \sum_{l=1}^{c} ||\mathbf{f}^l - \mathbf{o}^l||^2 \tag{1}$$

In other words, the cost function for a given clustering is specified by how well local classifiers can predict cluster labels for each data point.

3.3 Solving the optimization problem

This section summarizes how to solve the optimization problem in Equation 1 in order to obtain the actual clustering.

For particular choices of classifiers, there exists a linear mapping $\mathbf{L}$ from the scaled cluster indicators to the classifier outputs, i.e. $\mathbf{o}^l = \mathbf{L}\mathbf{f}^l$. In such a case, following [2], we define a real symmetric matrix $\mathbf{T} \in \mathbb{R}^{n \times n}$:

$$\mathbf{T} = (\mathbf{I}_n - \mathbf{L})^\top (\mathbf{I}_n - \mathbf{L}) \tag{2}$$

The first c eigenvectors of $\mathbf{T}$ present a solution to a relaxed version of the optimization problem in Equation 1 (see also [2]). This means that we do not directly obtain the scaled partition matrix. From the two options mentioned in [2], we use the k-means algorithm in order to compute the final cluster labels from the eigenvectors.

4 LLCA with Kernel Ridge Regression

This section describes the original variant of LLCA, and reports the results obtained with it on both the Two-Moons dataset and the Truck-Parts dataset.

4.1 Local kernel ridge regression

Kernel ridge regression is a regularized, kernelized variant of least-squares linear regression. By using only the k nearest neighbors of a data point when fitting a kernel ridge regression model for that particular data point, we obtain local kernel ridge regression. This introduces two parameters: the number k of nearest neighbors to consider, and the regularization parameter λ, consistent in notation with [2].

The choice of kernel ridge regression leads to a particular linear mapping $\mathbf{L}_{\text{KRR}}$[1], and subsequently to a matrix $\mathbf{T}_{\text{KRR}}$. The matrix entry $(\mathbf{L}_{\text{KRR}})_{ij}$ is given by the weight of the j-th datapoint in the i-th regression model if the j-th datapoint is a neighbor of the i-th datapoint. Otherwise, the matrix entry is zero. For a derivation of kernel ridge regression, please see [2] or [4].

4.2 Radial basis function kernel

In order to perform kernel ridge regression, we need to choose a kernel that controls the influence of a local neighborhood. A Gaussian-shaped kernel is used in this work. This is the same kernel as one of the kernels used in [2]. The shape of the kernel is controlled by parameter σ:

$$\Phi(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{\sigma} ||\mathbf{x}_i - \mathbf{x}_j||^2\right) \tag{3}$$

4.3 SVM as out-of-sample extension

The Truck-Parts dataset is too large to be handled by our implementation of LLCA. One way to address this issue are out-of-sample extensions.

In this report, we train a *support vector machine (SVM)* on the cluster labels produced by LLCA on a sample of the dataset. The sample incorporates $100 \cdot r$ percent of the dataset. The parameter r trades off between feeding more data to LLCA and spending less time in computation. The trained support

[1] The matrix $\mathbf{L}_{\text{KRR}}$ corresponds to matrix $\mathbf{A}$ in [2].

vector machine is used to assign cluster labels to the whole dataset, i.e. both the training sample and the out-of-sample data.

4.4 Implementation

At the time of writing, no readily-available implementation of LLCA existed to the best of our knowledge. Our own implementation of LLCA with kernel ridge regression can be divided into four steps that will be explained in detail in the following:

1. Construct matrix $\mathbf{L}_{\text{KRR}}$ explicitly.
2. Construct matrix $\mathbf{T}_{\text{KRR}}$ explicitly.
3. Compute eigenvectors of $\mathbf{T}_{\text{KRR}}$.
4. Cluster eigenvectors with k-means.
5. Cluster out-of-sample data.

For the first step, we solve n linear systems with k unknowns each, using the `GSL` [5] routines `gsl_linalg_LU_decomp` and `gsl_linalg_LU_solve`. Computing the non-zero entries of $\mathbf{L}_{\text{KRR}}$ is therefore in $\mathrm{O}(nk^3)$. The local classifiers can be trained in parallel. The training of the local classifiers corresponds to filling one of the rows of matrix $\mathbf{L}_{\text{KRR}}$. We use the parallel-for-pragma from `OpenMP` to that end.

In the second step, the matrix $\mathbf{T}_{\text{KRR}}$ is explicitly computed by matrix-matrix multiplication using `gsl_blas_dgemm`. There might be faster ways to do this because $\mathbf{L}_{\text{KRR}}$ is a sparse matrix, but premature optimizing is the root of all evil [6].

In the third step, the eigenproblem solver `gsl_eigen_symmv` is suitable because $\mathbf{T}_{\text{KRR}}$ is symmetric.

In the fourth step, we use the software library `kmlocal` [7] in order to obtain the final clustering from the first c eigenvectors.

In the fifth step, we use a SVM out-of-sample extension that is implemented in C++, using the LIBSVM library [8]. We use C-support vector classification [8] with a radial basis function parametrized by γ as kernel. Note that the parametrization of the radial basis function in LIBSVM is not the same as in Equation 3.

4.5 Hyperparameters

The presented method of LLCA with kernel ridge regression has several hyperparameters that need to be chosen.

1. The number of clusters c.
2. The width σ of the radial basis function kernel used in the kernel ridge regression models.
3. The number k of nearest neighbors considered in the kernel ridge regression models.
4. The regularization parameter λ for the kernel ridge regression models.
5. The out-of-sample proportion r.
6. The parametrization γ of the radial basis function used in the out-of-sample extension.
7. The cost C of violating the constraints in the out-of-sample extension.

Ideally, most of the parameters would be learned from the data without requiring user interaction. In contrast to a supervised context where this can be achieved in a validation step, we use a combination of educated guesses, manual inspection, and cluster measures in order to select good parameter values.

4.6 Cluster measures

The TWO-MOONS dataset has a clustering that is apparent to the human observer. Thus, we can measure the quality of a clustering result with the adjusted Rand index (Section 4.6.1).

In contrast, there is no ground truth available for the TRUCK-PARTS datasets. Clustering these datasets is just one stage of the processing pipeline described in [1]. Ideally, the quality of the clustering result would be determined with help of the overall result at the end of this process. For the TRUCK-PARTS dataset, we restrain ourselves to investigate into two

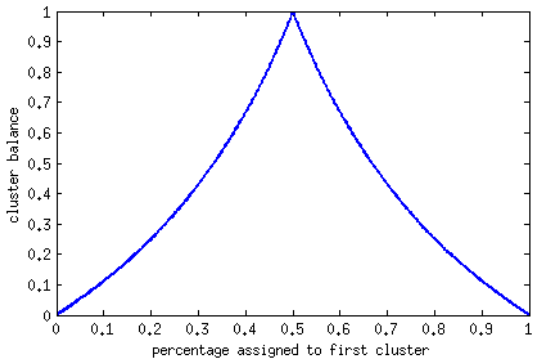

Figure 4: The cluster balance in the case of two clusters. The cluster balance is plotted versus the percentage of data points assigned to the first (or the second, respectively) cluster.

internal cluster measures in order to validate clustering results.

4.6.1 Adjusted Rand index

The *adjusted Rand index* [9] measures similarity between two clusterings. We use this index in order to compare clustering results for the Two-Moons dataset where ground truth is available.

The adjusted Rand index is based on the Rand index, but corrected for chance. The correction for chance means that we measure clustering quality relative to what we can expect in case that the two compared clusterings are picked at random. Equation 4 shows how to obtain the adjusted Rand index R_{adj} from the Rand index R, its expectation, and $R_{\max} = 1$. Large values of R_{adj} indicate that the two clusterings are similar.

$$R_{\text{adj}} = \frac{R - \mathbb{E}[R]}{R_{\max} - \mathbb{E}[R]} \tag{4}$$

We refer the reader to [9] for a detailed formula that describes how to compute both the Rand index and its expectation. Software implementations are readily available because the index is popular.

4.6.2 Cluster balance

The *cluster balance* is defined as the ratio of the sizes of the smallest and the largest clusters (see Figure 4). It is easy to implement. While it does not capture much information about the cluster shapes, it is useful to quickly single out undesired clusterings.

4.6.3 Expected density measure

Besides the cluster balance, there are other internal cluster measures that do not require to compare clusterings against external ground truth. This section describes the *expected density measure* [10].

This internal measure is based on the similarity graph of the data. The vertices V in the similarity graph correspond to the data points, and the edges are weighted with the similarities between the data points. As a similarity measure, we use a radial basis function kernel of the same form as in Equation 3 with σ_{expd} as parameter.

The expected density measure $\bar{\rho}$ is a function of the cluster sizes $|V_i|$, the total number of data points $|V|$, the weights $w(V_i)$ of the subgraphs induced by the clusters, and the total graph weight $w(G)$. The measure is computed according to [10] as follows

$$\bar{\rho} = \sum_{i=1}^{k} \frac{|V_i|}{|V|} \frac{w(G_i)}{|V_i|^{\theta}} \tag{5}$$

$$\theta = \frac{\ln w(G)}{\ln |V|} \tag{6}$$

Unlike the adjusted Rand index, we implemented the expected density measure on our own. The weights of the subgraphs can be computed by iterating over all edges in the subgraph. In order to make this feasible, we modify the measure such that it only considers the similarity graph where each vertex is only connected to the k_{expd} nearest neighbors. More precisely, an edge connects two vertices if either of the vertices belongs to the nearest neighbors of the other. This results in a sparse similarity graph, and reduces

computational complexity as compared to the original version in [10].

During all of the experiments, we set $\sigma_{\text{expd}} = 0.1$ and $k_{\text{expd}} = 25$. The effects of changing these parameters have not been studied in this work.

4.6.4 Other cluster measures

Besides the cluster balance and the expected density measure, there are other well-known internal cluster measures such as Dunn's index [11], the Davies-Bouldin index [12], and the Silhouette index [13]. In the literature, these three cluster measures are reported to work well only if the data contains spherical clusters. Investigating into the utility of these cluster measures is out of the scope of this work. Instead, we concentrate on the expected density measure which is supposedly less susceptible to preferring spherical clusters.

4.7 Results on Two-Moons

Figure 1 shows a hand-picked result where the Two-Moons are perfectly clustered with parameter values $c = 2$, $\sigma = 1$, $k = 10$, $\lambda = 0.1$, $r = 0.5$, $\gamma = 10$, and $C = 1$.

In order to find out how sensitive the method is to the parameter values, we performed a grid search over the parameters σ, k, λ, r, γ, and C. The following levels were chosen:

- $\sigma \in \{0.01,\ 0.1,\ 1,\ 10\}$
- $k \in \{5,\ 10,\ 50\}$
- $\lambda \in \{0.01,\ 0.1,\ 1,\ 10\}$
- $r \in \{0.1,\ 0.5,\ 1.0\}$
- $\gamma \in \{0.01,\ 0.1,\ 1,\ 10\}$
- $C \in \{0.01,\ 0.1,\ 1,\ 10,\ 100\}$

Thus, 2880 experiments were performed in total.

4.7.1 Influence of hyperparameters

The parameter σ can be large for the Two-Moons dataset. Even for very large σ, the two moons can be perfectly separated. A theoretical explanation for this behavior is that the radial basis function kernel becomes constant with $\sigma \to \infty$:

$$\lim_{\sigma \to \infty} \Phi(\mathbf{x}_i, \mathbf{x}_j) = 1$$

Since the local regression model for one data point in LLCA considers only its k nearest neighbors, letting $\sigma \to \infty$ will make all nearest neighbors equally important. The number of nearest neighbors turned out to be critical as well: the parameter pair $(\sigma,\ k)$ controls the locality of the kernel ridge regression models.

The parameter r cannot be chosen without regarding the other parameters as well. At least for the Two-Moons dataset, sampling from the data changes the densities of the two clusters, and let the intra-cluster distances come closer to the inter-cluster distances.

A crude, quick and dirty approach to selecting good parameters for the Two-Moons dataset is to marginalize over the distribution of successful experiments. An experiment is considered successful when the adjusted Rand index of the obtained clustering is larger than 0.95. Figure 5 shows the number of successful experiments versus the levels of each parameter. This helps to select parameter levels that are somewhat insensitive to changes in other parameters.

4.7.2 Helpfulness of cluster measures

It is interesting to know whether the internal cluster measures are helpful in choosing the right clustering. To this end, following the approach in [10], we plot the correlation between the internal cluster measures and the adjusted Rand index (Figure 6). In the region where the expected density is larger than 1.33, the adjusted Rand index is consistently larger than 0.95. If we select the clustering with the largest expected density, we achieve an adjusted Rand index of 0.992,

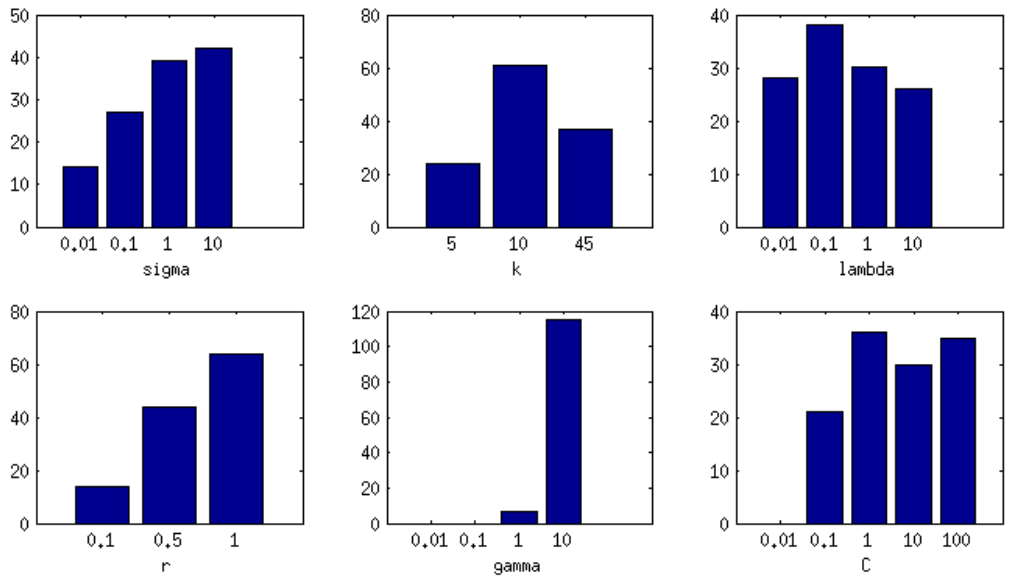

Figure 5: Each plot shows the number of clusterings achieving an ARI greater than 0.95 against the levels of one parameter. This is a crude method to find parameters that are likely not too sensitive to changes in other parameters.

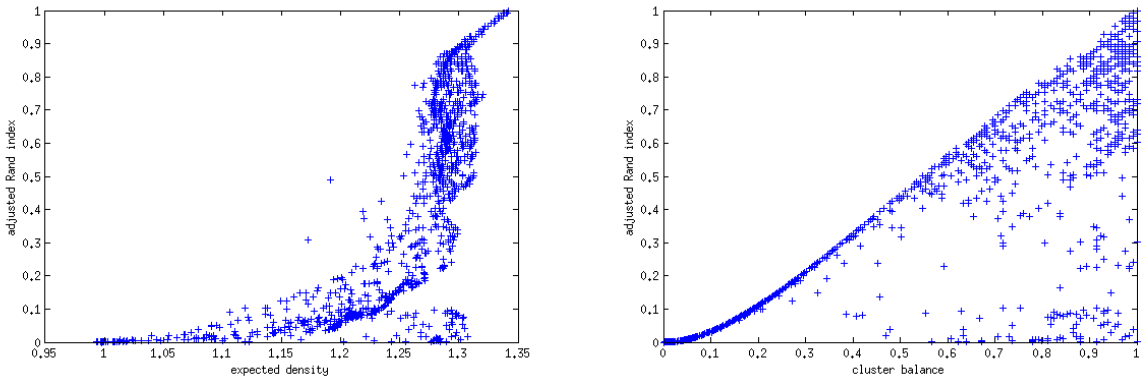

Figure 6: Correlation between the expected density measure (cluster balance, respectively) and the adjusted Rand index in the experiments on the Two-Moons dataset.

which means that only one data point was assigned to the wrong cluster.

The cluster balance however did not prove successful in selecting a good clustering. The adjusted Rand index was close to zero in one of the experiments where the clusters were perfectly balanced (Figure 6). However, since we are looking for a balanced clustering in the case of the Two-Moons dataset, the cluster balance is very helpful to sort out bad clusterings.

4.7.3 Computation time

Altogether, the 2880 experiments took 44 minutes (wall-time) on an Intel i5 Lenovo T510 ThinkPad. This includes time needed for computing various cluster measures and writing experiment results to disk. On average, the LLCA implementation clusters the Two-Moons without employing the out-of-sample extension (Section 4.3) in 1.7 seconds. With the out-of-sample extension and $r = 0.75$, the computation time drops to 0.34 seconds.

4.8 Results on Truck-Parts

The Truck-Parts dataset differs from the Two-Moons dataset in both size and dimensionality. Instead of working with all 126 features, we only take the first seven principal components into account as it is done in [1]. The expected density measures guides the selection of parameters because ground truth is not available.

Similarly to the Two-Moons dataset, we ran a grid search in order to extract knowledge about the influence of the parameters c, σ, k, λ, r, γ, and C. The following levels were chosen:

- $c \in \{2,\ 4,\ 8\}$
- $\sigma \in \{0.01,\ 0.1,\ 1,\ 10\}$
- $k \in \{10,\ 20,\ 30\}$
- $\lambda \in \{0.01,\ 0.1,\ 1,\ 10\}$
- $r \in \{0.1,\ 0.2,\ 0.3\}$
- $\gamma \in \{0.01,\ 0.1,\ 1,\ 10\}$

	c	σ	k	λ	r	γ	C
top	2	0.1	10	0.01	0.1	10	10
middle	4	1	20	0.01	0.2	1	100
bottom	8	10	20	0.1	0.3	1	100

Table 1: The parameters chosen for the three selected experiments in Figure 7.

- $C \in \{0.01,\ 0.1,\ 1,\ 10,\ 100\}$

In total, this makes 8640 experiments. It is not necessary to run these many experiments in order to obtain a reasonable clustering. Yet, it is better to have more comprehensive data when exploring the effect of parameters.

Figure 7 shows the results of three experiments in three rows and two columns. The parameter values are given in Table 1. The diagrams in the left column show the clustering of the data points projected on pairs of principal components. The caption of Figure 8 explains more detailed how to interpret these diagrams. The right column of Figure 7 shows the clustered chassis beams for two (top), four (middle), and eight (bottom) clusters. The colors do not carry any meaning beyond cluster indication.

4.8.1 Computation time

In total, the 8640 experiments took more than ten days on the same notebook as used in Section 4.7.3 to finish (fortunately, the experiments need no surveillance). Clustering the Truck-Parts dataset with the parameters as in Table 1 takes less than 45 seconds on average. If only the fraction $r = 0.1$ of the dataset is used to train the out-of-sample extension, then the clustering requires around four seconds.

4.9 Potential improvements

The computation time can be potentially reduced. The matrix $\mathbf{L}_{\text{KRR}}$ is a sparse matrix with $n \cdot k$ nonzero entries, a fact that might be exploited in im-

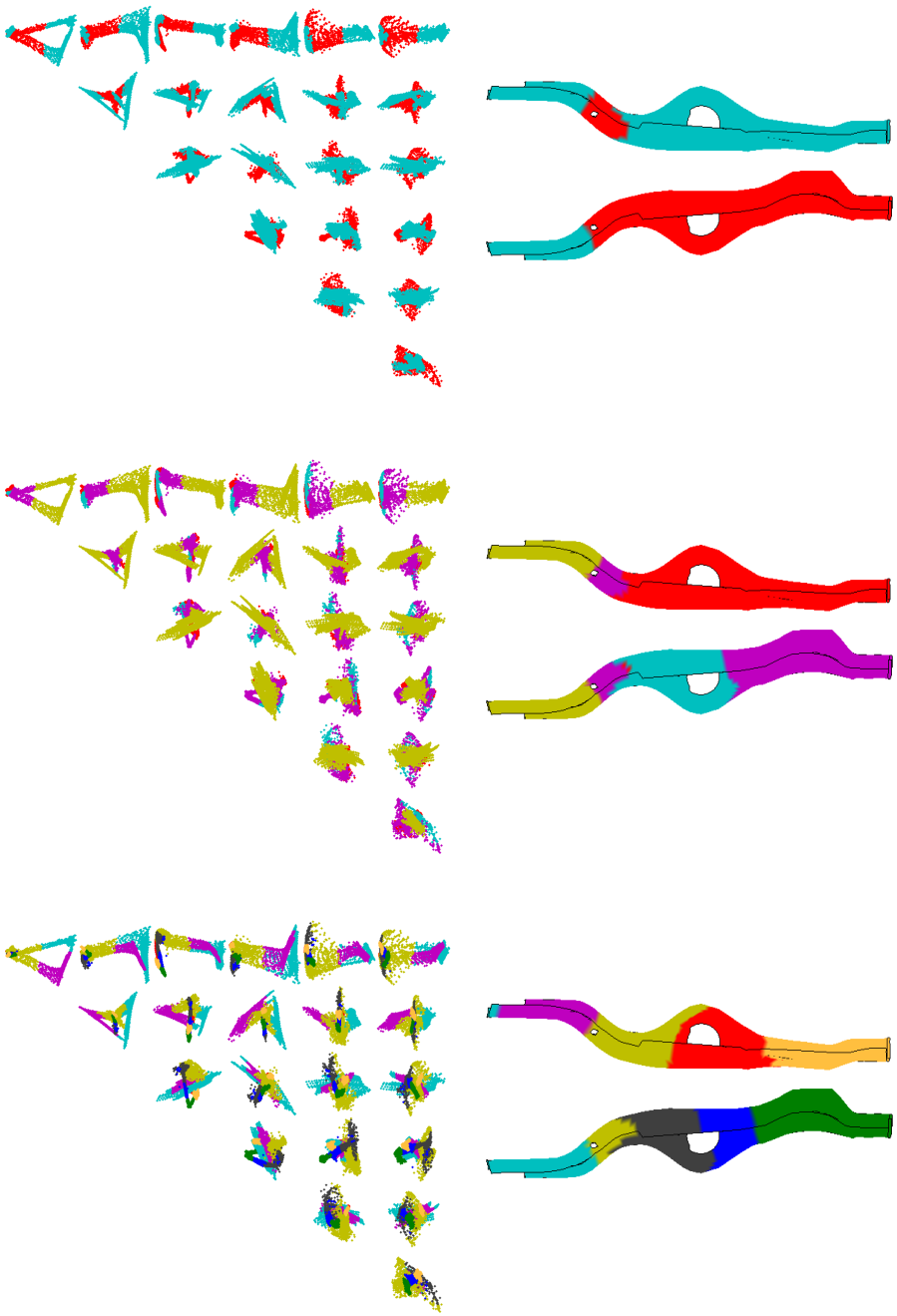

Figure 7: Computed cluster labels. The left column shows pairwise projections of the principal components of the TRUCK-PARTS dataset. The right column shows the corresponding parts of the truck. Each row corresponds to a different clustering with two clusters (top), four clusters (middle), and eight clusters (bottom).

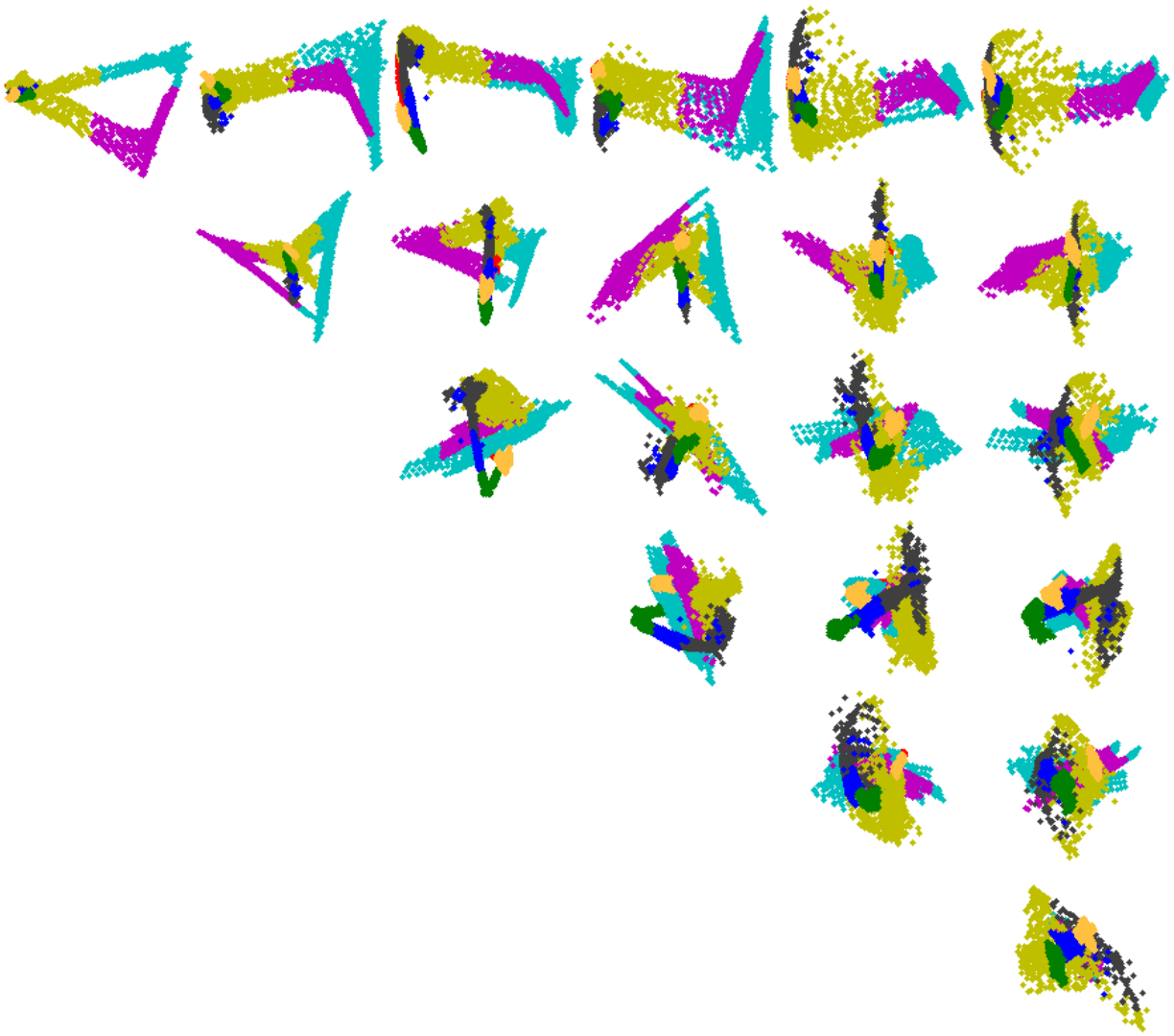

Figure 8: A clustering of the TRUCK-PARTS dataset. Projections of the high-dimensional data are arranged in a plot matrix where the (i, j)-th plot corresponds to the projection of the data onto the planes defined by the i-th and the j-th principal components. In order to avoid clutter, only the right upper triangle of the plot matrix is shown. Note that clusters are not well-separated in any two dimensions alone. The clusters therefore partially occlude each other in the drawing, which makes the plots more difficult to interpret. Still, some of the plots reveal symmetrical structure in both the data and the cluster labels.

plementations. For example, $\mathbf{L}_{\mathrm{KRR}}$ might be stored using a sparse matrix representation, thereby reducing the storage requirements from $\mathrm{O}(n^2)$ to $\mathrm{O}(nk)$. Additionally, computing $\mathbf{T}_{\mathrm{KRR}}$ explicitly should be possible in $\mathrm{O}(n^2 + nk^2)$ with a suitable sparse matrix representation that allows to skip over zeros in both rows and columns. Furthermore, we only need the first few eigenvectors of $\mathbf{T}_{\mathrm{KRR}}$.

5 LLCA with Sparse Grid Classifiers

This section describes the sparse-grid variant of LLCA, and reports the results obtained with it on the two moons dataset. First, we give a short introduction to sparse grids. Then, we describe how to use sparse grids for classification in LLCA. Finally, we report the results, focusing on the question why the method did not work as expected.

5.1 Sparse grids

Sparse grids are a means of discretizing functions. They have been proven useful in a number of applications where it is necessary to deal with high-dimensional data. An in-depth discussion is found in [14], but here we motivate the use of sparse grids and introduce what is required for LLCA on sparse grids.

Full grids suffer from the curse of dimensionality: the number of grid points in a full grid grows exponentially with the number of dimensions. For example, if we have 10 grid points along each dimension of a 5-dimensional space, then we already have 100,000 grid points in the full grid. Figure 9 shows examples of a full grid, a regular sparse grid, and an adaptively refined sparse grid.

Sparse grids address this issue by making a selection of grid points that is optimal with respect to a certain function class with bounded mixed second derivatives in each direction. The number of grid points in a sparse grid does not grow in the same order as with full grids.

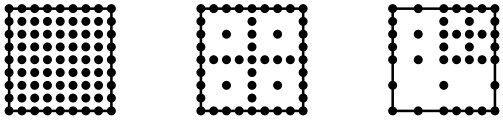

Figure 9: A full grid (left), a regular sparse grid (middle), and an adaptively refined sparse grid (right).

Functions on a sparse grid are represented with a hierarchical basis of basis functions with non-local support. A sparse grid function f is a linear combination of of the hierarchical basis functions $\varphi = (\varphi_1, \ldots, \varphi_m)$, and the hierarchical coefficients $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)$:

$$\mathrm{f}(x) = \sum_{i=1}^{N} \alpha_i \underset{i}{\varphi}(x) \tag{7}$$

In order to improve the accuracy of the function representation, we can use regular sparse grids of a higher *grid level*. This provides us with a means of controlling the model complexity.

5.2 Principle of LLCA on sparse grids

LLCA on sparse grids builds upon the principles explained in Section 3. The choice of sparse grid classifiers results in a particular linear mapping $\mathbf{L}_{\mathrm{SG}}$, and thus a matrix $\mathbf{T}_{\mathrm{SG}}$ as compared to LLCA with kernel ridge regression. Apart from this, the rest of the algorithm works the same. The method is described as follows: first, we define a sparse grid, then we derive how to learn classifiers for the individual data points, and finally we derive the system matrix $\mathbf{T}_{\mathrm{SG}}$.

Let m denote the number of grid points of a regular sparse grid in the d-dimensional feature space. This sparse grid provides the basis for all the classifiers that are to be learned as part of the algorithm. With this grid, we associate a matrix $\mathbf{B} \in \mathbb{R}^{n \times m}$, which we refer to as the *grid evaluation matrix*:

$$\mathbf{B} = \begin{pmatrix} \varphi_1(\mathbf{x}_1) & \cdots & \varphi_m(\mathbf{x}_1) \\ \cdots & \cdots & \cdots \\ \varphi_1(\mathbf{x}_n) & \cdots & \varphi_m(\mathbf{x}_n) \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1^T \\ \cdots \\ \mathbf{b}_n^T \end{pmatrix} \quad (8)$$

The classifier $\mathbf{o}_i^l$ for the l-th cluster and the i-th data point is defined as a sparse grid function with hierarchical coefficients $\widetilde{\boldsymbol{\alpha}}_i^l \in \mathbb{R}^m$:

$$\mathbf{o}_i^l(\mathbf{x}_i) = \sum_{j=0}^{m} \varphi_j(\mathbf{x}_i)\, \widetilde{\boldsymbol{\alpha}}_{ij}^l = \widetilde{\boldsymbol{\alpha}}_i^{l\top} \mathbf{b}_i \quad (9)$$

The hierarchical coefficients $\widetilde{\boldsymbol{\alpha}}_i^l$ need to be learned from the scaled cluster indicator $\mathbf{f}^l$. The coefficients can be found by solving the following linear system of equations [3]:

$$(\lambda \mathbf{C} + \mathbf{B}^\top \mathbf{B}) \widetilde{\boldsymbol{\alpha}}_i^l = \mathbf{B}^\top \mathbf{f}^l \quad (10)$$

Here, $\lambda \mathbf{C} \in \mathbb{R}^{m \times m}$ serves as a regularization term. We restrict $\mathbf{C}$ to be symmetric. In the original paper about data mining on sparse grids, $\mathbf{C}$ corresponds to a discrete Laplacian [3], but we choose $\mathbf{C} = \mathbf{I}_m$ to be the identity matrix. Equation 10 can be solved for the sought-after hierarchical coefficients:

$$\widetilde{\boldsymbol{\alpha}}_i^l = (\lambda \mathbf{C} + \mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top \mathbf{f}^l \quad (11)$$

Let us define another classification problem. The reason becomes clear in Equation 13, whereas the interpretation remains an open question.

$$\left(\lambda \mathbf{C} + \mathbf{B}^\top \mathbf{B}\right) \boldsymbol{\alpha}_i = \mathbf{b}_i \quad (12)$$

Plugging Equation 11 into Equation 9, while making use of Equation 12, we take the first step towards the sought-after linear mapping $Ł_{\mathrm{SG}}$:

$$\begin{aligned} \mathbf{o}_i^l(\mathbf{x}_i) &= \widetilde{\boldsymbol{\alpha}}_i^{l\top} \mathbf{b}_i && (13) \\ &= \left[(\lambda \mathbf{C} + \mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top \mathbf{f}^l\right]^\top \mathbf{b}_i && (14) \\ &= \mathbf{f}^{l\top} \mathbf{B} (\lambda \mathbf{C} + \mathbf{B}^\top \mathbf{B})^{-\top} \mathbf{b}_i && (15) \\ &= \mathbf{f}^{l\top} \mathbf{B} \boldsymbol{\alpha}_i && (16) \\ &= \boldsymbol{\alpha}_i^\top \mathbf{B}^\top \mathbf{f}^l && (17) \end{aligned}$$

Now we can stack the sparse grid coefficients $\boldsymbol{\alpha}_i$ obtained from Equation 12 vertically in order to form matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$:

$$\mathbf{A} = \begin{pmatrix} \boldsymbol{\alpha}_1^\top \\ \vdots \\ \boldsymbol{\alpha}_n^\top \end{pmatrix} \quad (18)$$

Then Equation 13 can be compactly rewritten for all data points

$$\mathbf{o}^l = \mathbf{A} \mathbf{B}^\top \mathbf{f}^l \quad (19)$$

Hence, the linear mapping is $\mathbf{L}_{\mathrm{SG}} = \mathbf{A}\mathbf{B}^\top$. This yields the system matrix

$$\mathbf{T}_{\mathrm{SG}} = (\mathbf{I}_n - \mathbf{A}\mathbf{B}^\top)^\top (\mathbf{I}_n - \mathbf{A}\mathbf{B}^\top) \quad (20)$$

5.3 Troubleshooting

The LLCA variant on sparse grid was expected to correctly cluster the Two-Moons dataset. Unfortunately, all attempts failed. However, it turns out that there is a theoretical explanation why these attempts failed. The explanation is given in Section 5.4.

5.4 Regularization parameter λ

Experiments revealed that the parameter λ, which was intended to control the regularization term, does not have any effect at all on the eigenvectors of the system matrix $\mathbf{T}_{\mathrm{SG}}$. A theoretical explanation follows.

We can compactly rewrite Equation 12 for all data points.

$$\left(\lambda \mathbf{C} + \mathbf{B}^\top \mathbf{B}\right) \mathbf{A}^\top = \mathbf{B}^\top \tag{21}$$

Then, Equation 21 can be solved for matrix $\mathbf{A}$.

$$\mathbf{A} = \mathbf{B} \left(\lambda \mathbf{C} + \mathbf{B}^\top \mathbf{B}\right)^{-1} \tag{22}$$

It is apparent that matrix $\mathbf{A}\mathbf{B}^\top$ is symmetric.

$$\mathbf{A}\mathbf{B}^\top = \mathbf{B} \left(\lambda \mathbf{C} + \mathbf{B}^\top \mathbf{B}\right)^{-1} \mathbf{B}^\top \tag{23}$$

Therefore, the system matrix can be rewritten as

$$\mathbf{T}_{\mathrm{SG}} = (\mathbf{I}_n - \mathbf{A}\mathbf{B}^\top)^2 \tag{24}$$

It is now clear that matrices $\mathbf{T}_{\mathrm{SG}}$ and $\mathbf{A}\mathbf{B}^\top$ share eigenvectors. The next step in the analysis is to find the eigenvectors of $\mathbf{A}\mathbf{B}^\top$. To this end, we decompose matrix $\mathbf{B}$ by singular value decomposition into $\mathbf{U} \in \mathbb{R}^{n \times n}$, $\mathbf{D} \in \mathbb{R}^{n \times m}$, and $\mathbf{V} \in \mathbb{R}^{m \times m}$.

$$\mathbf{B} = \mathbf{U}\mathbf{D}\mathbf{V}^\top \tag{25}$$

Then, we expand and transform Equation 23 until we can read off the spectral decomposition of matrix $\mathbf{A}\mathbf{B}^\top$.

$$\begin{aligned}
\mathbf{A}\mathbf{B}^\top &= \mathbf{B} \left(\lambda \mathbf{I}_m + \mathbf{B}^\top \mathbf{B}\right)^{-1} \mathbf{B}^\top \\
&= \mathbf{U}\mathbf{D}\mathbf{V}^\top \left(\lambda \mathbf{I}_m + \mathbf{V}\mathbf{D}^\top \mathbf{D}\mathbf{V}^\top\right)^{-1} \mathbf{V}\mathbf{D}^\top \mathbf{U}^\top \\
&= \mathbf{U}\mathbf{D} \left(\lambda \mathbf{I}_m + \mathbf{D}^\top \mathbf{D}\right)^{-1} \mathbf{D}^\top \mathbf{U}^\top \qquad (26) \\
&= \sum_{i=1}^{m} \mathbf{u}_i \frac{d_i^2}{d_i^2 + \lambda^2} \mathbf{u}_i^\top \qquad (27)
\end{aligned}$$

Let $\mathbf{\Sigma}_\lambda \in \mathbb{R}^{n \times n}$ denote the diagonal matrix of eigenvalues of matrix $\mathbf{T}_{\mathrm{SG}}$. Since matrices $\mathbf{T}_{\mathrm{SG}}$ and $\mathbf{A}\mathbf{B}^\top$ share eigenvectors, we have

$$\mathbf{T}_{\mathrm{SG}} = \mathbf{U}\mathbf{\Sigma}_\lambda \mathbf{U}^\top \tag{28}$$

This effectively means that the eigenvectors of matrix $\mathbf{T}_{\mathrm{SG}}$ are independent of the parameter λ. Only the eigenvalues of matrix $\mathbf{T}_{\mathrm{SG}}$ depend on parameter λ. Even more, the eigenvectors of matrix $\mathbf{T}_{\mathrm{SG}}$ are equal to the left-singular vectors of matrix $\mathbf{B}$.

6 Conclusion

The results of the local learning based clustering algorithm are encouraging. The non-linearly separable Two-Moons dataset with 500 data points is successfully clustered in less than two seconds. By employing a SVM-based out-of-sample extension, the time can be reduced to less than one second. Drawing a conclusion for the Truck-Parts dataset with over 6000 data points requires more prudence because ground truth is not available. Yet, the clusters found in the chassis beams look reasonable by visual inspection. It is pleasant to see the expected density measure work considerably well for both datasets.

The contributions of this work comprise several aspects. First, LLCA has been applied in a new context to both artificial data and more realistic data from car crash simulations. Second, LLCA has been extended with an out-of-sample extension. The latter is based on a support vector machine. Hence, an unsupervised learning technique is combined with a supervised learning algorithm in order to produce a non-linear clustering algorithm that is expected to cope with several thousand data points. Third, the code for this combined algorithm is publicly available as open source. Fourth, a sparse-grid variant of LLCA is derived, which unfortunately does not work. However, a problem with this variant is revealed, which might either prevent or guide future work in that direction.

There is much room for future work and many questions remain open. A natural continuation of this work is to cluster the whole pick-up truck, not only the four chassis beams. The crucial question here is whether the out-of-sample extension alone can handle the more than 66,000 data points of the whole truck model. Potential improvements of the implementation are described in Section 4.9. Reducing the

number of hyperparameters by automatic model selection would improve usability. The expected density measure needs more attention. It might be interesting to see how the presented clustering framework performs when plugged into the analysis workflow in [1]. It would be helpful to see how the clustering method performs in comparison to other approaches on datasets with ground truth beyond those used in [2]. Regarding the sparse grid variant, it might be worth investigating whether the problem is connected to the choice of the identity matrix in the regularization term.

In summary, future work comprises further evaluation and comparison of the presented clustering scheme to existing methods.

7 Acknowledgments

I thank Benjamin Peherstorfer from the Chair of Scientific Computing at Technische Universität München for his supervision and his excellent support during the whole course of this work, and the chair for providing the Truck-Parts dataset. I thank Josephine Sullivan from the Computer Vision and Active Perception Lab at Royal Institute of Technology, Stockholm for her supervision and especially her key contributions to Section 5.4. I thank both supervisors for their cooperation across borders and across institutions.

References

[1] B. Bohn, J. Garcke, R. Iza-Teran, A. Paprotny, B. Peherstorfer, U. Schepsmeier, and C.-A. Thole, "Analysis of Car Crash Simulation Data with Nonlinear Machine Learning Methods," in *International Conference on Computational Science*, 2013.

[2] M. Wu and B. Schölkopf, "A Local Learning Approach for Clustering," in *Advances in Neural Information Processing Systems*, 2006.

[3] J. Garcke, M. Griebel, and M. Thess, "Data Mining with Sparse Grids," *Computing*, vol. 67, pp. 225–253, Oct. 2001.

[4] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, P. Alken, M. Booth, and F. Rossi, *GNU Scientific Library Reference Manual*. Network Theory Ltd., 3 ed., 2009.

[6] D. E. Knuth, "Structured Programming with go to Statements," *ACM Computing Surveys*, vol. 6, pp. 261–301, Dec. 1974.

[7] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "Efficient Algorithms for K-Means Clustering." `http://www.cs.umd.edu/~mount/Projects/KMeans`.

[8] C.-c. Chang and C.-j. Lin, "LIBSVM : A Library for Support Vector Machines," tech. rep., 2012.

[9] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, no. 1, pp. 193–218, 1985.

[10] B. Stein and S. Meyer, "On Cluster Validity and the Information Need of Users," in *3rd International Conference on Artificial Intelligence and Applications* (M. Hanza, ed.), pp. 216–221, ACTA Press, 2003.

[11] J. C. Dunn, "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters," *Journal of Cybernetics*, no. March 2013, pp. 23–57, 1973.

[12] D. L. Davies and D. W. Bouldin, "A Cluster Separation Measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, pp. 224–227, Apr. 1979.

[13] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, Nov. 1987.

[14] H.-J. Bungartz and M. Griebel, "Sparse grids," *Acta Numerica*, vol. 13, pp. 147–269, 2004.