SurvivalLVQ: Interpretable Supervised Clustering and Prediction in Survival Analysis via Learning Vector Quantization

Jasper de Boer[a,b,*], Klest Dedja[a,b], Celine Vens[a,b]

[a] *KU Leuven, Department of Public Health and Primary Care – Campus Kulak, 8500 Kortrijk, Belgium*
[b] *Itec, imec research group at KU Leuven, KU Leuven – Campus Kulak, 8500 Kortrijk, Belgium*

Abstract

Identifying subgroups with similar survival outcomes is a pivotal challenge in survival analysis. Traditional clustering methods often neglect the outcome variable, potentially leading to inaccurate representation of risk profiles. To address this, we present SurvivalLVQ, a novel interpretable method that adapts Learning Vector Quantization (LVQ) to survival analysis. Unlike traditional classification uses of LVQ, SurvivalLVQ groups individuals by survival probabilities and assigns a unique survival curve to each cluster, representing the collective survival behavior within that group. Moreover, it can predict individual survival curves using weighted averages from nearby clusters. When tested on 76 benchmark datasets, it outperformed other clustering methods and showed competitive prediction performance. SurvivalLVQ bridges the gap between clustering techniques and outcome-oriented methods. Its strong clustering performance, coupled with competitive prediction capabilities and with easy to interpret outcomes, make it a promising tool for various applications within survival analysis.

Keywords: Survival analysis, clustering, learning vector quantization (LVQ), Interpretable

*Corresponding author. jasper.deboer@kuleuven.be

1. Introduction

Survival analysis is a branch of statistics that focuses on modeling the duration leading up to specific events, be it death, recurrence of a disease, or mechanical breakdown. A distinguishing feature of survival analysis is its adeptness at handling 'censored' data. This means it can provide unbiased predictions even when only knowing a lower or upper boundary of the actual event time. This approach is highly valuable in numerous fields, including medicine, engineering, and economics, as it allows researchers and practitioners to identify factors that affect the likelihood of an event occurring and to forecast future outcomes.

In many medical applications, a key challenge is to identify subgroups of individuals with similar survival outcomes. For instance, identifying different subgroups of patients with distinct survival probabilities can help physicians tailor treatment plans to each patient's specific needs. Moreover, subgroup identification can also be used to discover novel bio-markers or risk factors that are associated with certain subgroups. This information can then provide valuable insights for developing new treatments and improving patient care [1, 2, 3]. Finally, the explainability of these outcomes, in terms of predictive modeling or clustering, is essential. Being able to clearly understand and communicate the factors that influence survival outcomes and subgroup identification not only can ensure a more effective, personalized treatment, but it also enhances trust among both practitioners and patients, as they can better understand the rationale behind medical decisions [4, 5].

Clustering in survival analysis can serve a critical role in distinguishing different risk categories within groups of individuals. However, traditional clustering techniques do not include the outcome variable in the clustering process, which results in clusters that may not accurately represent groups with differing risk profiles. This omission shows a missed opportunity to leverage crucial data related to survival time in the clustering process.

One algorithm that results in interpretable subgroups and employs outcome information is Learning Vector Quantization (LVQ), which was proposed in the

context of classification tasks [6]. LVQ is a machine learning algorithm that partitions data points into clusters based on their similarity and class. The representatives of each cluster are called prototypes. These prototypes serve as intuitive representative samples from the data, offering users insights into the defining features of each cluster.

An extension of LVQ, Generalized Matrix Learning Vector Quantization (GMLVQ), further enhances interpretability [7]. GMLVQ introduces a parametric distance measure, which not only quantifies differences between data points but also provides insights into feature importance and intercorrelation.

In contrast to many vector quantization algorithms (e.g., K-means), both traditional LVQ and GMLVQ are supervised algorithms and therefore leverage label information. This labeled approach amplifies their interpretability, as clusters can be directly associated with known outcomes. Moreover, LVQ is actively used in a wide range of applications in the biomedical field, for example to detect SARS-CoV-2 types based on their RNA sequences [8], classifying eye states using electroencephalography (EEG) data [9], and in neuroimaging [10].

LVQ has been extended to time series classification [11]. However, to our knowledge LVQ has not been extended to the survival analysis setting yet.

In this article, we present SurvivalLVQ, a novel adaptation of the LVQ framework tailored specifically for survival analysis. Drawing from the advanced features of the GMLVQ model, our method harnesses the clustering capability of LVQ and groups datapoints by their survival probabilities. Furthermore, SurvivalLVQ assigns a distinct survival curve to each prototype, representing the collective survival behavior within that group. In addition, the method can also be used to predict individualized survival curves by computing a weighted average of the survival curves from clusters adjacent to the instance being predicted.

We have rigorously tested our method across a comprehensive collection of 76 benchmark datasets. We compared SurvivalLVQ's performance with other prevalent clustering and prediction survival analysis techniques.

In Section 2, we refer to the related literature of our research, providing a context within which our contributions can be understood. Following this,

Section 3 introduces Learning Vector Quantization (LVQ) and its various adaptations. Next, in Section 4 we describe how we modified LVQ to be compatible with the survival analysis setting and created survivalLVQ. We present the experimental setup in Section 5 and present the results in Section 6. In Section 7, we illustrate the practical application of SurvivalLVQ through a detailed example analysis. Section 8 concludes by discussing the implications of our findings and potential future research directions in this area.

2. Related work

In the body of literature dedicated to survival analysis, the primary emphasis has been on methods that estimate point risk scores or that model individualized survival distributions. In contrast, clustering based on risk-profiles has received less attention, but a few methods have previously been proposed. These include feature-based clustering methods such as K-means and hierarchical clustering [12, 13, 14], and a two-step approach followed by Bair et al. [15] and Gaynor et al. [16]. More specifically, Bair et al. applied univariate Cox proportional hazards models to establish the correlation between each covariate and the target outcome, and assigned a Cox score to each covariate. Through a cross-validation process, an optimal threshold was identified for the set of covariates to be incorporated into the K-means algorithm. Only the covariates that surpassed this threshold and demonstrated significant survival disparities among the clusters (verified through a log-rank test) were considered for the K-means algorithm.

Gaynor et al. [16] subsequently refined this methodology by introducing supervised sparse clustering. Extending upon Bair's technique, Gaynor et al. also calculated Cox scores for the covariates, but allocated weights to these covariates instead, based on their respective Cox scores derived from a multivariate model. The weighted covariates were then used in the K-means clustering.

It should be noted that the previously mentioned methods do not incorporate survival outcomes during the clustering stage, and therefore the resulting

clusters do not necessarily align with survival outcomes.

Beyond the clustering frameworks discussed, researchers have explored diverse methodologies. Luo et al. [17] present a thorough review that encapsulates the cutting-edge advancements in this field. Below we will provide a concise summary.

Xia et al.[18] proposed an outcome driven attention-based multi-task model for patient classification and clustering in acute coronary syndrome using K-means on latent features. Mouli et al.[19] presented DeepCLife, optimizing survival function differences between clusters through deep learning. Chapfuwa et al. [20] proposed a Bayesian method that enhances prediction and risk profiling by clustering data in a latent space. Manduchi et al.[21] developed a deep generative model to reveal intricate distributions underlying explanatory variables and survival outcomes.

Nevertheless, while employing deep learning and latent spaces can significantly enhance the power of these techniques, they often come at the cost of interpretability. This becomes especially problematic in the field of medicine, where the ability to explain and understand model outcomes is often important.

3. Learning Vector Quantization (LVQ)

In this section, we provide an introduction to the LVQ algorithm family, specifically the Generalized Learning Vector Quantization (GLVQ), as well as the Generalized matrix Learning Vector Quantization (GMLVQ) algorithms, as they serve as the foundation for the method proposed in this article. Please note that our discussion of LVQ based algorithms is not exhaustive. For a more comprehensive overview of available algorithms, we recommend referring to the review by Nova et al. [22].

LVQ is a class of supervised machine learning algorithms, primarily used for classification tasks. The first version of LVQ was introduced by Kohonen [6]. LVQ is based on the concept of prototype vectors, which are representatives of each class in the dataset. During training the main goal is to update the LVQ

prototypes to minimize classification errors.

LVQ stands out for its interpretability. The prototypes provide an easy-to-understand representation of each class. These prototypes provide insights into the underlying data structure and can facilitate the identification of distinguishing features. Its appeal extends to ease of implementation, customizable prototype numbers for optimal complexity and performance balance, and effective multi-class problem handling.

The first versions of LVQ faced a few limitations, including the possibility of slow convergence and inconsistent performance. Many authors have proposed adaptations to LVQ to overcome these limitations, but also to further improve performance or interpretability. We highlight further advantages arising from specific adaptations of the LVQ family below.

3.1. Generalized Learning Vector Quantization (GLVQ)

Generalized Learning Vector Quantization (GLVQ), introduced by Sato and Yamada [23], is an extension of the classical LVQ algorithm. It employs a cost function-based method to refine prototype locations, tackling classical LVQ's potential issues like slow convergence and instability. By evaluating the distances between an input sample and the nearest correct and incorrect class prototypes, GLVQ directs prototypes more effectively, enhancing classification accuracy and ensuring more stable and rapid convergence than the heuristic-based updates of classical LVQ.

Since GLVQ forms the basis for GMLVQ and therefore also for our newly proposed method, we describe this method in more detail. Let us assume we have a dataset $\mathcal{D} = \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \ldots, (\boldsymbol{x_N}, y_N)\}$, where $\boldsymbol{x_i} \in \mathbb{R}^p$ represents a feature vector, and $y_i \in 1, 2, \ldots, C$ represents the corresponding class label for $i = 1, 2, \ldots, N$. There are C distinct classes in the dataset. For each class c, there are M_c prototypes, and the total number of prototypes is $M = \sum_{c=1}^{C} M_c$.

The classification process in GLVQ takes a simple nearest-prototype approach. A new data point $\boldsymbol{x} \in \mathbb{R}^p$ is mapped to the class label $c(\boldsymbol{x}) = c(\boldsymbol{w_i})$ of the closest prototype, i.e., the prototype w_i for which $d(\boldsymbol{w_i}, \boldsymbol{x}) \leq d(\boldsymbol{w_j}, \boldsymbol{x})$ holds

for every $j \neq i$. In the case of ties, an arbitrary choice is made. Distance function d can be any valid distance metric but often the squared Euclidean distance metric is selected: $d(\boldsymbol{x}, \boldsymbol{w}) = (\boldsymbol{x} - \boldsymbol{w})^T(\boldsymbol{x} - \boldsymbol{w})$. In the distance calculation between a prototype and a datapoint, it is possible to exclude certain features from the calculation if they contain missing values. This partial distance calculation allows LVQ to effectively handle missing data [24, 25].

During training a cost function is minimized to find the optimal positions of class prototypes in the feature space. Gradient descent is typically employed for optimization and the cost function is defined as follows:

$$E_{GLVQ} = \sum_{i=1}^{N} \Phi(\mu_i) \quad \text{where} \quad \mu_i = \frac{d^+(\boldsymbol{x_i}) - d^-(\boldsymbol{x_i})}{d^+(\boldsymbol{x_i}) + d^-(\boldsymbol{x_i})} \tag{1}$$

where Φ is a monotonically increasing function, e.g. the logistic function or the identity function. More specifically, $d^+(x_i)$ denotes the distance to the nearest data point from the same class, whereas $d^-(x_i)$ represents the distance to the closest data point from a different class. As such, it is worth noting that a negative value of the numerator in μ_i indicates accurate classification of the data point, while a positive numerator indicates inaccurate classification. The denominator ensures that $-1 < \mu_i < 1$ and the magnitude of μ_i reflects the confidence level in the prediction: values near -1 denote high confidence in correct predictions, and values near 1 indicate confidence in incorrect predictions.

A crucial step in the learning process of LVQ algorithms (including GLVQ) is a proper initialization of the prototype locations. When the LVQ system has only one prototype per class, it is common to assign the initial locations of the prototypes to the class centroids resulting from a K-means partitioning. In cases where the system has multiple prototypes per class, locations are typically derived using k-means clustering for each class.

An advantage of GLVQ is that it can work with any differentiable distance measure and customized cost function. This allows for more flexibility in adapting the algorithm to various types of data and problem domains. One such adaptation is Generalized Matrix Learning Vector Quantization (GMLVQ).

3.2. Generalized Matrix Learning Vector Quantization (GMLVQ)

Generalized Matrix Learning Vector Quantization (GMLVQ) was introduced by Schneider et al. [7] and extends GLVQ by changing the distance metric to an adaptive metric. GMLVQ incorporates a full matrix of relevance factors in the distance metric, accounting for the correlations between features and their importance for classification. The distance metric is adapted during the training phase. This generalized distance metric is shown in Equation 2.

$$d^{\Lambda}(\boldsymbol{w}, \boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{w})^T \Lambda (\boldsymbol{x} - \boldsymbol{w}) \tag{2}$$

Here, $\Lambda \in \mathbb{R}^{p \times p}$ where p is the dimensionality of the feature space. The Λ matrix represents the learned distance metric and can capture feature importance and account for correlations between features. Λ has to be symmetric and positive-definite to define a valid distance metric. This is ensured by substituting $\Lambda = \Omega^T \Omega$, where typically $\Omega \in \mathbb{R}^{p \times p}$ and computing Ω instead.

The training of GMLVQ uses the same cost function as GLVQ (Equation 1) but substitutes the distance metric with the new distance function. The learning rate for Λ can be selected independently from the learning rate used for adjusting the prototype locations. Typically, it is set to be an order of magnitude smaller to ensure a more gradual pace of metric learning in comparison to the updates made to prototype locations [7].

The incorporation of feature importance and correlations commonly leads to improved classification performance. Furthermore, importance and correlations are made visible by the Λ matrix. The diagonal elements of the matrix represent the importance of each feature in the classification process whereas the off-diagonal elements capture the correlations or dependencies between pairs of features. This ability to account for feature correlations enables GMLVQ to perform better than GLVQ in cases where features are not independent.

During training initialization, the Λ matrix is typically set to the identity matrix that assigns equal weights of the features at the start of the training process. Additionally, in order to prevent degeneration during training and

to ease interpretation, the constraint $\sum_i \Lambda_{ii} = 1$ is enforced by normalizing the matrix after every update during training. Consequently, each feature's relevance lies between 0 (no importance) and 1 (maximal importance).

The approach however, becomes computationally impractical for high- dimensional data, as the number of Λ parameters grows quadratically with the number of the features. This challenge can be addressed by employing standard preprocessing methods, like implementing Principal Component Analysis (PCA) to the data instances. Another option is to restrict the rank of the matrix Λ. Since $\Lambda = \Omega^T\Omega$, we can restrict Ω to be a $\in \mathbb{R}^{m\times p}$, where $m \ll p$, resulting in positive-semidefinite matrices with a restricted rank [7].

4. SurvivalLVQ

In this section we introduce our new algorithm named SurvivalLVQ, an adaptation of the GMLVQ algorithm to the survival analysis setting. This adaptation is achieved by replacing the fixed class label of each prototype with a survival function, which is learned during the training phase. A detailed description of the training and inference phases is provided in this section.

Given a survival dataset $\mathcal{D}_S = \{(\boldsymbol{x_1}, \delta_1, T_1), (\boldsymbol{x_2}, \delta_2, T_2), \ldots, ((\boldsymbol{x_N}, \delta_N, T_N)\}$, where $\boldsymbol{x_i} \in \mathbb{R}^p$ represents a feature vector, $\delta_i \in \{0, 1\}$ denotes the censoring status (1 if true event was observed, 0 otherwise) and T_i the event or censoring time; we need to employ appropriate statistical tools to analyse this type of data. A fundamental tool for computing survival over time is the Kaplan-Meier estimator [26]. This non-parametric method is capable of handling censored data and offers an unbiased estimate of the survival function for the underlying population, even in the presence of censoring. For each time point t_i, let d_i be the number of events that have happened at time t_i, and let n_i be the number of instances that have not yet experienced any event, then the Kaplan-Meier estimator is defined as:

$$K(t) = \prod_{i:\ t_i \leq t} \left(1 - \frac{d_i}{n_i}\right) \tag{3}$$

In the context of SurvivalLVQ, the M prototypes extracted by the LVQ approach now correspond to the Kaplan-Meier survival estimate, referred to as $\boldsymbol{c}$. This vector corresponds to a survival function of the instances represented (i.e. instances within the same cluster), and has T components, where T denotes the number of discrete time steps.

The predicted survival function of a new datapoint, $\hat{S}(\boldsymbol{x})$, is parameterized by the two closest prototypes, j and k, as shown in Equation 4.

$$\hat{S}(\boldsymbol{x}) = \frac{d_k^\Lambda}{d_j^\Lambda + d_k^\Lambda}\boldsymbol{c_j} + \frac{d_j^\Lambda}{d_j^\Lambda + d_k^\Lambda}\boldsymbol{c_k} \tag{4}$$

Here, d_j^Λ is short for $d^\Lambda(\boldsymbol{w_j}, \boldsymbol{x})$ and denotes the distance to the closest prototype j. The distance to the second closest prototype (k) is denoted by d_k^Λ. The distance metric $d^\Lambda(\boldsymbol{w}, \boldsymbol{x})$ is the same as used in GMLVQ (Equation 2).

As highlighted by Equation (4), only two prototypes are involved during the inference phase, in contrast to the full set of M available prototypes. This choice was made to enhance interpretability, as considering only two prototypes simplifies the interpretation of the predicted survival function. Additionally, we observed that when more than two prototypes are used, their locations drift away from the data during training. This behaviour, although not harmful for predictive performance, is undesirable because it hinders interpretability as the prototypes are no longer meaningful cluster representatives. This behavior is akin to what is commonly observed in the probabilistic variants of LVQ [27].

4.1. The learning rule

Training the survival LVQ model is based on the minimization of a cost function. The presented cost function in this section is based on the Brier score [28]. The Brier score is a commonly used metric for performance assessment of survival prediction, thus using this score as cost function follows naturally. The Brier score was extended by Graf et al. [29] for survival problems with right censored data. Individual contributions to the empirical Brier score are reweighed according to the censored information:

$$BS_t = \frac{1}{N}\sum_{i=1}^{N} \alpha_{it}(\hat{S}_{it} - y_{it})^2 \tag{5}$$

Where $y_{it} = 1$ if and only if the corresponding event has $T_i \geq t$, and where α_{it} denotes the weight of the i-th instance and is defined as:

$$\alpha_{it} = \begin{cases} \delta_i / \hat{G}(T_i) & t \geq T_i \\ 1/\hat{G}(t) & t < T_i \end{cases} \tag{6}$$

Here $\hat{G}(t)$ denotes the Kaplan-Meier estimate of the censoring distribution, which can be computed using the Kaplan-Meier formula on the same data with reversed censoring status.

The Brier score gives a model performance for a single time t. The score can be integrated to produce a model performance score for a time interval. A discrete version of this Integrated Brier Score (IBS) is used as the cost function for training the survivalLVQ model. It is defined as $IBS = \frac{1}{T}\sum_{t=1}^{T} BS_t$. From this the SurvivalLVQ cost function follows:

$$E_{SLVQ} = \frac{1}{N\,T}\sum_{i=1}^{N}\sum_{t=1}^{T} \alpha_{it}(\hat{S}_{it} - y_{it})^2 \tag{7}$$

Here $\hat{S}_{it}$ indicates the predicted survival probability as predicted by the SurvivalLVQ system for sample i at time t (Equation 4).

During the training process, the cost function is minimized by using an approach similar to that used in GMLVQ. However, in the case of SurvivalLVQ, the prototype labels must also be defined and updated. Note that these labels have to be constrained to obtain a valid (monotonically decreasing) survival function. In our implementation, we opted to compute a Kaplan-Meier estimate for each prototype during the update phase, after every epoch. Each data point is assigned to the prototype it is closest to. Subsequently, the prototype survival curve $\boldsymbol{c} \in \mathbb{R}^T$ is determined by calculating its Kaplan-Meier estimate (Equation 3). To set the initial positions of the prototypes during our experiments, we used the K-means++ algorithm [30] applied to the feature vectors only. An overview of the learning process is provided in Algorithm 1.

Algorithm 1 Overview of the steps taken to train a SurvivalLVQ model.

Input: Survival dataset $\mathcal{D}_S = \{(\boldsymbol{x_i}, \delta_i, T_i)\}_{i=1}^{N}$

Input: Number of prototypes M, number of epochs N_{epochs}

function UpdateLabels()

  for $p = 1, \ldots, M$ **do**

    Define $\mathcal{D}_p = \{\, (\boldsymbol{x_i}, \delta_i, T_i) \in \mathcal{D}_S \mid \forall q \neq p,\, d^{\Lambda}(\boldsymbol{w_p}, \boldsymbol{x_i}) < d^{\Lambda}(\boldsymbol{w_q}, \boldsymbol{x_i}) \,\}$

    $\boldsymbol{c_p} \leftarrow$ Kaplan-Meier estimate from $\mathcal{D}_p$

  end for

end function

Initialize prototypes W using K-means++ on feature vectors

Initialize relevance matrix Λ (e.g., identity matrix)

UpdateLabels() ▷ Initialize prototype labels

for *epoch* $= 1, \ldots, N_{epochs}$ **do**

  Update W and Λ by minimizing E_{SLVQ} (Equation 7) with an optimizer

  UpdateLabels()

end for

Output: Prototypes $W = \{\boldsymbol{w_p}\}_{p=1}^{M}$, relevance matrix Λ ▷ Explanation

Output: Prototype survival curves $\{\boldsymbol{c_p}\}_{p=1}^{M}$ ▷ Prediction

5. Experiments

In this section, we outline the experiments conducted to test and evaluate the performance of SurvivalLVQ. We discuss the benchmark datasets used for this evaluation and give a brief overview of the other methods that were compared. We then describe how the experiments were set up and conclude with the results.

5.1. Datasets

SurvSet [31] is a collection of open-source survival analysis datasets designed for benchmarking machine learning algorithms and statistical methods. The collection contains 76 datasets, varying in dimensionality and background. Most of these datasets originate from biomedicine. The collection includes both long

datasets with fewer features than observations and wide datasets, with more features than observations (e.g., genomics datasets). Additionally, there is substantial variation in censoring rates across the datasets. The main properties of the dataset collection (number of observations, number of features, and censoring rate) are shown in Figure 1.

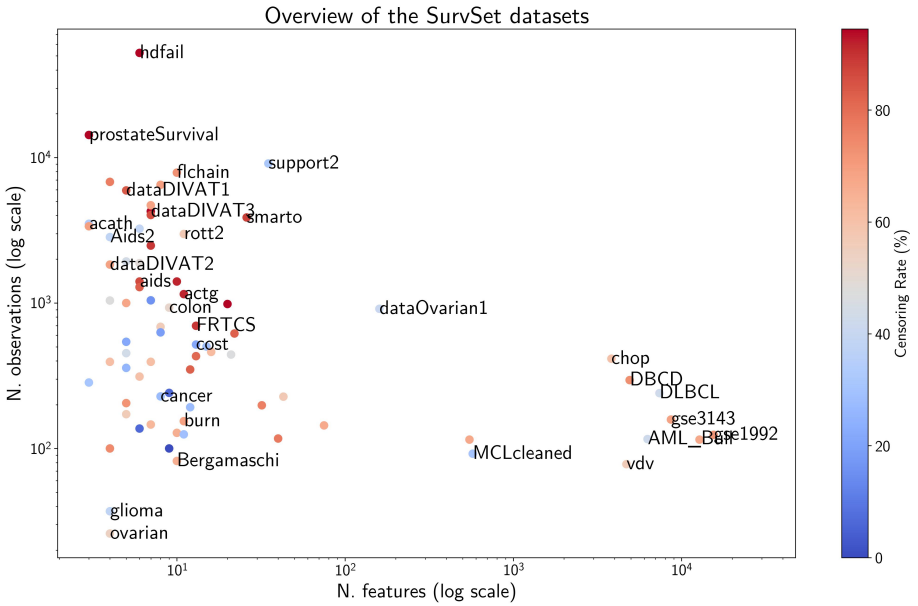


Figure 1: Overview of the 76 datasets contained in the SurvSet library. On top: large datasets; on the right side: high dimensional datasets

We use SurvSet to test survivalLVQ by comparing its performance across these diverse datasets and assessing its effectiveness in various scenarios. Furthermore we compare the performance to other interpretable algorithms.

5.2. Competing algorithms

We evaluated the performance of survivalLVQ by comparing it with several existing clustering methods and predictive methods. We primarily focused on approaches with interpretability and cluster discovery characteristics, as these are the central aspects of our work. To assess our algorithm's competitiveness in terms of predictive performance, we also included Random survival forests

(RSF) [32], a widely used method with lower interpretability.

5.2.1. K-means

We used the K-means++ algorithm [30] as a reference clustering method. This is equivalent to using the SurvivalLVQ method and refraining from performing any update steps after initialization, thus preserving the initial values for both the distance metric ($\Lambda = I$) and prototype locations. By adopting this baseline method, we could also assess whether SurvivalLVQ learns any meaningful information during training.

5.2.2. Cox proportional hazards model (CoxPH)

Cox regression, also known as the Cox proportional hazards model [33] (CoxPH), is a widely used statistical method in survival analysis. The primary goal of CoxPH is to estimate the hazard function, which describes the instantaneous risk of experiencing the event of interest at a given time, given that the subject has survived up to that point. The model assumes that the hazard functions are proportional across instances. CoxPH allows users to identify the variables that significantly impact the time-to-event and quantify their effects. Due to the variety in dimensionality of the datasets within SurvSet, we employed a regularized version of it involving elastic net.

5.2.3. Supervised sparse clustering (SSC)

The method introduced by Gaynor et al. [16] serves as an enhancement over the approach by Bair et al. [15]. Gaynor et al.'s method is referred to as Supervised Sparse Clustering (SSC) in the following sections. SSC begins by computing Cox scores for covariates through a multivariate model to ascertain their relationship with the target outcome. Gaynor et al. allocate weights to covariates in accordance with their Cox scores. These weighted covariates are subsequently incorporated into the K-Means clustering algorithm.

It is important to note that the SSC method is not inherently designed for prediction performance; if applied as it is, SSC yields only K unique predicted values, one for each cluster. To obtain a better performance assessment, we

modify the predictions of SSC in an analogous way to what is used in Survival-LVQ, as in Equation 4. Specifically, we generate individualized survival curves for new test samples by taking a distance-weighted average of the Kaplan-Meier curves from the two closest clusters. This modification allows for individualized predictions, ensuring a fair evaluation of SSC's predictive capabilities.

5.2.4. Survival trees

Survival trees [34, 35] are a type of decision tree used in survival analysis to model the relationship between predictor variables and survival time. In a typical regression tree, the leaf nodes contain the average value of the response variable for the observations that fall into that node. The leaf nodes of survival trees contain information about the survival probability or time for the observations that fall into that node. Specifically, the leaf nodes of a survival tree contain the estimated survival probability or median survival time for the individuals in the subgroup represented by that node. The split criterion used in survival trees is typically based on the log-rank test.

Survival trees can also be used for clustering, by setting the number of leaf nodes to be equal to the number of clusters. Consequently, all data points that are assigned to the same leaf node are considered to constitute a cluster. This approach facilitates a direct comparison between survival trees and clustering algorithms in terms of their grouping capabilities.

5.2.5. Random survival forest (RSF)

Random survival forests (RSF) [32] is an ensemble learning method for survival analysis, which extends the popular Random Forest algorithm to handle time-to-event data. The main objective of RSF is to model the relationship between predictor variables (features) and the time until an event of interest occurs (such as death or failure). RSF works by constructing multiple survival trees during the training process. Each tree is built using a bootstrapped sample of the original dataset, and at each split, only a random subset of features is considered for the splitting criterion. This approach introduces randomness

and diversity among the trees, improving the model's generalization ability and reducing overfitting. Once the ensemble of trees is built, predictions for a new observation are made by aggregating the outcomes of individual trees. In survival analysis, this is often done by computing the cumulative hazard function for each tree and then averaging them across the entire forest.

5.3. Experimental setup

This section details our experimental setup, covering preprocessing, hyperparameter tuning for each algorithm, performance metrics used, and statistical methods for comparing algorithm results.

5.3.1. Data preprocessing

In our experiments, we found that different algorithms require varying degrees of preprocessing, influenced by their inherent properties. Distance-based algorithms like LVQ and K-means are sensitive to the scale of features, making data scaling essential. In contrast, tree-based methods and the CoxPH model are less affected by feature scaling as they rely on data distribution and event time rankings, respectively. We utilized z-scoring scaling (based on the training set) for standardization, as it proved advantageous for certain algorithms.

To counter skewed data impacting distance-based methods, we integrated an automated skew correction using the Fisher-Pearson coefficient of skewness [36], applying log-transformation to parameters when it lessened skewness in training data. We employed one-hot encoding solely for distance-based and regression methods, acknowledging that tree-based algorithms inherently handle categorical variables.

Despite LVQ's ability to manage missing data [24, 25], we uniformly applied median imputation across all methods for consistency.

5.3.2. Implementation

We used several well-known Python packages to apply various survival analysis algorithms in our experiments. Specifically, we used the CoxPH and tree-

based implementations from scikit-survival (0.18.0) [37], as well as the K-means implementation from scikit-learn (1.2.2) [38].

For the implementation of survivalLVQ, we used PyTorch [39]. We utilized Batch Gradient Descent with the Adam optimizer, which has been shown to be effective for GMLVQ by LeKander et al. [40]. A fixed batch size of 128 was used. To achieve a slower time-scale of metric learning compared to the prototype updates, we set the learning rate of the distance metric to be one order of magnitude smaller (one-tenth) than the general learning rate, as recommended by the original GMLVQ authors [7].

Several of the datasets have high dimensionality and have more more features than observations. To limit the number of trainable parameters for these datasets, we restricted the rank of the LVQ relevance matrix Λ using the low rank strategy described in Section 3.2. Since $\Lambda = \Omega^T\Omega$, we limited Ω such that $\Omega \in \mathbb{R}^{n \times p}$, where we set n to the number of observations and p is the number of features. During the initialization phase of training, we set Ω (and consequently Λ) to the identity matrix for datasets where the number of features is fewer than the number of observations. For datasets with more features than observations, we set Ω such that it approximates the identity matrix, making use of the Moore-Penrose inverse.

5.3.3. Cross-validation and hyperparameter tuning

To ensure a fair comparison between the different algorithms, we preformed a grid search to tune the hyperparameters using cross-validation. The C-index was used as the primary metric for the tuning process. The C-index was chosen over the log-rank score since it can be calculated for all evaluated methods. Further, the C-index was favored over the integrated Brier score (IBS) to avoid any bias towards SurvivalLVQ, which optimizes the IBS during training.

For small datasets (under 500 samples), we used five-fold cross-validation with nested loops for hyperparameter tuning via grid search. For larger datasets (500+ samples), we simplified to an 80/20 train/test split, maintaining hyperparameter tuning but eliminating the outer loop. All splits were stratified to

ensure equal proportions of censored samples.

The hyperparameter grid per algorithm is listed in Table 1. Parameters that are not listed were kept to the defaults.

Algorithm	**Hyperparameters**
SurvivalLVQ	learning rate: [1e-3, 1e-2, 1e-1] epochs: [16,32,64,128] #prototypes: [2,3,4,5]
K-means	#prototypes: [2,3,4,5]
SSC	#prototypes: [2,3,4,5], L1 ratio: min=0.001, max=0.999, steps=20 alpha: min=log10(1e-5), max=log10(100), steps=20
CoxPH	L1 ratio: min=0.001, max=0.999, steps=20 $\log_{10}$alpha: min=-5, max=2, steps=20
Survival trees	min weight fraction leaf: [0.005, 0.01, 0.02, 0.05, 0.10] max leaf nodes: [4, 8, 16, 32, 64, 128], max #features: [None, "sqrt", "log2"]
RSF	#trees: 100

Table 1: Hyperparameters tuned for the six algorithms

5.3.4. Performance metrics

The benchmark datasets lack explicit cluster labels, making traditional clustering metrics like the silhouette score unsuitable. Following Chapfuwa et al. [20], we used the log-rank score to assess clustering in survival analysis but recognized its limitations, as survival curve separation doesn't fully reflect clustering quality. Thus, we supplemented it with survival analysis metrics like the integrated Brier score and concordance index (CI), evaluating clustering effectiveness through alignment with survival outcomes.

The following metrics were calculated:

- **The integrated Brier score (IBS)**, computed as:

$$BS_t = \frac{1}{N}\sum_{i=1}^{N} \alpha_{it}(\hat{S_{it}} - y_{it})^2 \tag{8}$$

Where α is defined in Equation 6, and takes into account the censoring status and the censoring distribution.

- **The C-index**, also known as Harrell's concordance index, quantifies the pairwise concordance between predicted risk scores and observed survival times, ranging from 0 to 1, with higher values indicating better prediction performance.

- **The Inverse Probability of Censoring Weighted (IPCW) C-index** [41] is a modification of the traditional C-index that accounts for censoring in survival data. It adjusts for the probability of censoring, weighting the contribution of each observation to the concordance calculation based on the inverse of its censoring probability. The IPCW C-index is a more robust performance metric in the presence of censoring, as it accounts for the potential bias introduced by non-random censoring in survival analysis.

- **The log-rank score** [42] performs a hypothesis testing to compare the survival distributions of two populations, where the null hypothesis is that the two groups have identical hazard functions. Note that this metric can not be calculated for the non-clustering methods.

We consider an algorithm to be performant if the discovered clusters can effectively predict survival outcomes, as measured by metrics such as the IBS or the (IPCW) C-index. Additionally, the clusters should exhibit well-separated survival functions, which can be evaluated using the log-rank score.

5.4. Statistical methods

In our analysis, we used the Friedman test to assess significant differences in algorithm rankings across datasets. If significant, we applied the Nemenyi post-hoc test for pairwise comparisons to pinpoint significant performance differences, using a critical distance (CD) to define significant rank differences. Both tests were conducted at a 95% confidence level.

6. Experimental results

This section presents results from two experiments. The first focuses on the clustering algorithms, and includes the calculation of the log-rank score, which is an metric that can be computed for these clustering methods. The second evaluates the predictive performance of SurvivalLVQ and comparing it with that of the five other algorithms described previously.

6.1. Clustering experiment

In our first experiment we compared the performance of four clustering methods: SurvivalLVQ, K-means, SCC, and Survival trees. Four evaluation metrics were used: average log-rank score, C-index, IPCW C-index, and IBS.

To ensure a fair and meaningful comparison, we took into account the influence of the number of clusters on the log-rank score by conducting the analysis using several fixed cluster sizes. By keeping the number of clusters constant for each comparison, we eliminated the confounding effect of different cluster sizes, enabling a clearer analysis of the performance characteristics of the methods. Table 2 summarizes the results of this experiment, presenting the performance rankings of each method for each cluster size.

In general, among the methods tested, SurvivalLVQ demonstrated the best performance across multiple cluster sizes. It achieved the highest C-index, IPCW C-index, and IBS in three out of four cluster sizes, indicating its effectiveness in capturing survival patterns within the data. Log-rank score measures the difference in survival rates between clusters, and the SurvivalLVQ log-rank score for two of four cluster sizes, was higher compared to the other methods.

SSC showed competitive results, however, its performance scores were only higher than Survival LVQ in the four cluster set. K-means and Survival trees show similar performance to each other, with varying results across different cluster sizes. They exhibited lower C-index, IPCW C-index, and IBS scores compared to SurvivalLVQ and SSC. The log-rank scores for K-means and Survival trees were also generally lower, indicating a less pronounced distinction in survival rates between clusters.

Method	Log-rank	C-index	IPCW C-index	Brier
		2 clusters		
SurvivalLVQ	**2.08**	**1.58**	**1.61**	**1.58**
K-means	3.07	2.93	2.89	2.91
SSC	2.36	2.18	2.24	2.77
Survival tree	2.49	3.32	3.26	2.74
		3 clusters		
SurvivalLVQ	**2.11**	**1.66**	**1.70**	**1.72**
K-means	3.11	2.88	2.89	2.88
SSC	2.25	2.20	2.20	2.39
Survival tree	2.54	3.26	3.21	3.00
		4 clusters		
SurvivalLVQ	2.38	**1.80**	**1.73**	**1.79**
K-means	2.93	2.94	2.94	2.68
SSC	**2.33**	2.22	2.29	2.45
Survival tree	2.37	3.04	3.04	3.08
		5 clusters		
SurvivalLVQ	2.53	2.04	2.11	**1.93**
K-means	2.85	2.91	2.93	2.53
SSC	2.36	**2.03**	**2.00**	2.39
Survival trees	**2.26**	3.03	2.96	3.14

Table 2: Summary of performance metrics rankings for clustering. The best results are highlighted in bold.

A Friedman test was performed for each cluster size and performance metric under consideration. The results consistently indicated statistical significance, validating the subsequent execution of a Nemenyi post-hoc test to identify the critical distance. This critical distance quantifies the minimum difference in performance that must be exceeded to establish statistical significance between methods. Here, the critical distance was determined to be 0.54. As such, any difference in performance between the algorithms that surpass this value can be considered statistically significant.

Table 2 illustrates a distinct trend: SurvivalLVQ's performance advantage over other methods is more pronounced with fewer prototypes, as indicated by its lower average ranks. This trend does not signify a reduction in LVQ's efficiency with more prototypes, but rather the comparative improvement of other methods at higher cluster counts. The efficacy of SurvivalLVQ stems from its combined use of the relevance matrix and prototype locations for differentiating groups, a benefit that gradually diminishes as cluster numbers increase. Our cross-validation in Section 6.2 confirms this, showing the optimal number of LVQ clusters to average at 3.6. Notably, this efficiency with fewer prototypes is key for model interpretability, a vital consideration in many applications.

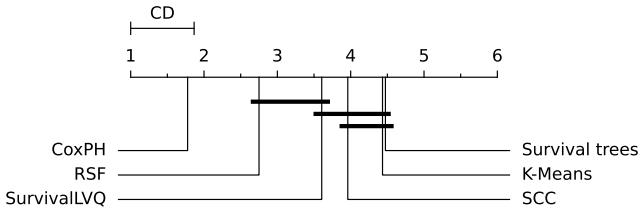
Overall, the results of this experiment suggest that SurvivalLVQ outperforms the other methods in terms of capturing survival patterns in the data. SSC, K-means, and Survival trees also showed promising results but were generally less effective in differentiating survival patterns among clusters. These findings provide valuable insights for researchers and practitioners when choosing a clustering method for survival analysis.

6.2. Prediction experiment

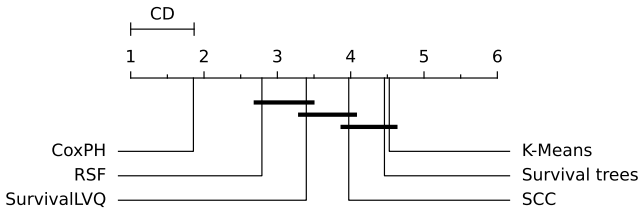
In our second experiment we compared the predictive performance of all previously introduced six methods on the 76 datasets. Three evaluation metrics were used: C-index, IPCW C-index, and the IBS. The average ranks of each method across these datasets were calculated for each evaluation metric. A Friedman test conducted for each metric revealed statistically significant differences across all metrics, and were therefore followed by Nemenyi tests. The results are summarized in the critical difference plots in Figure 2. The raw prediction outcomes from the evaluated algorithms are available as digital supporting material[1].

In all three metrics CoxPH demonstrated the best performance with the lowest average rank, followed by Random survival forest and then Survival-

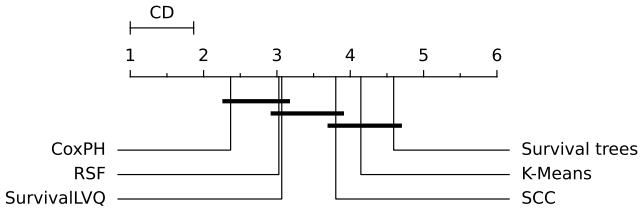
[1]Additional digital supporting material: https://itec.kuleuven-kulak.be/survivallvq/



(a) Nemenyi test, with C-index as measure of performance



(b) Nemenyi test, with IPCW C-index as measure of performance



(c) Nemenyi test, with Integrated Brier score as measure of performance

Figure 2: Critical difference plots, comparing the performance of various methods using the Nemenyi post-hoc test.

LVQ. The average ranks of SurvivalLVQ were notably lower than K-means and Survival tree, which had the highest average ranks for all metrics. The difference between the ranks of CoxPH and SurvivalLVQ exceeds the critical distance (CD), for the C-index and the IPCW C-index, showing a statistically significant difference in performance between these two methods. However, the difference between the ranks of SurvivalLVQ and RSF did not exceed the CD, for any of the three metrics.

Focusing on the SurvivalLVQ method, its performance in terms of C-index, IPCW C-index, and IBS were better than K-means and Survival trees, which

were observed to have the highest average ranks among the six methods. Compared to SSC, SurvivalLVQ performed better, especially in terms of the IBS. This makes it the best performant clustering method in terms of prediction performance, although the differences between SurvivalLVQ and SCC were not statistical significant.

Interestingly, RSF also outperformed SurvivalLVQ in all metrics but was second to CoxPH. Considering the critical distance, the differences in average ranks between SurvivalLVQ and RSF were never statistically significant. Among the three metrics, SurvivalLVQ ranks lowest on the IBS metric. This aligns with our expectations, given that we employed the IBS as the cost function during our implementation of SurvivalLVQ.

In conclusion, SurvivalLVQ exhibits competitive performance, particularly when compared to the clustering methods and survival trees. However, in this analysis, CoxPH and RSF show better performance across all the considered metrics. Further investigation may be required to understand the specific scenarios in which SurvivalLVQ may provide advantages over other methods in terms of prediction.

7. Example analysis on the PBC3 dataset

In this section we describe an example analysis of the PBC3 dataset [43]. The goal of this analysis is to demonstrate the use of SurvivalLVQ, the information it provides, and some considerations while performing an analysis.

The PBC3 clinical trial took place in six European hospitals and followed 349 patients with primary biliary cirrhosis (PBC). Patients were randomly assigned to receive either Cyclosporin A treatment (176 patients) or a placebo (173 patients). The initial aim was to observe the treatment's impact on survival time. However, due to a rise in liver transplants for these patients, the main outcome was changed to the time until "failure of medical treatment", meaning either death or needing a liver transplant [43]. The dataset contains 19 features in total after one-hot encoding, of which 7 are continuous and 12

are binary (zero or one). A summary of the features are listed in Table 3. Furthermore, the dataset contains the relevant survival analysis labels, namely $\delta_i \in \{0, 1\}$ denotes the censoring status (1 if event was observed, 0 otherwise) and T_i the event or censoring time. Here i refers to the patient id.

Feature index	
0-4	hospital (0: Copenhagen, 1: Hvidovre, 2: London, 3: Lyon, 4: Munich)
5	treatment (0: placebo, 1: CyA)
6	sex (1: males, 0: females)
7-9	histological stage (2, 3, 4)
10	histological missing (1: yes, 1: no)
11	previous gastrointestinal bleeding (1: yes, 0: no)
12	age (years)
13	creatinine (micromoles/L)
14	albumin (g/L)
15	bilirubin (micromoles/L)
16	alkaline phosphatase (IU/L)
17	aspartate transaminase (IU/L)
18	body weight (kg)

Table 3: Summary of the dataset features after one-hot encoding.

For our analysis, we adopted the preprocessing steps outlined in Section 5.3.1. This analysis primarily serves as a demonstration of the method, so we did not conduct an exhaustive hyperparameter search. After minimal manual tuning, we set a learning rate of 0.01, ran for 64 epochs, and used 3 prototypes. We employed an 80% training and 20% validation split. The learned LVQ system is visualized in Figure 3.

In Figure 3, both the validation data and the 3 prototypes are projected onto the first two eigenvectors of the relevance matrix (Λ). This projection facilitates a 2D visualization of the data and prototype positions. The figure reveals that Prototype 1 covers data points with higher survival times, Prototype 2 corresponds to a cluster of censored observations, and Prototype 3 is located

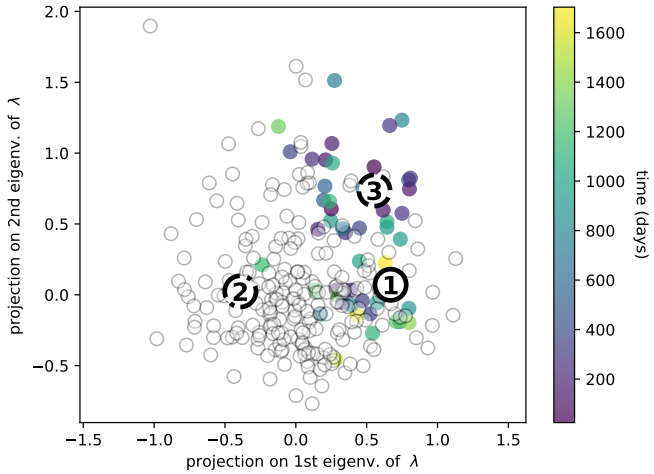


Figure 3: Scatter plot of validation data and 3 prototypes on the first two eigenvectors of the relevance matrix. Data point colors represent survival time, with white dots indicating censored observations.

amidst data points with relatively shorter survival times.

After looking at the scatter plot visualization, we can also look at the labels of the prototypes: The corresponding survival curves. Figure 4 provides a visualization of these curves.

Figure 4 confirms our previous observations: Prototype 1 displays a relatively high survival probability over time. Prototype 2 exhibits even superior survival rates, indicating that its associated cluster of censored datapoints represents patients who neither required a liver transplant nor died during the study. As previously mentioned, Prototype 3 is positioned among data points with shorter survival times and the corresponding survival curve reflects this trend. A K-sample log-rank hypothesis test of identical survival functions [44] shows that that the hazard rate of at least one group significantly differs from the others at some time ($p < 0.001$).

For more detail, Figure 5 presents several plots with information regarding feature importance.

Figure 5a displays the learned relevance matrix (Λ), offering insights into both feature importance (diagonal elements) and intercorrelation (off-diagonal

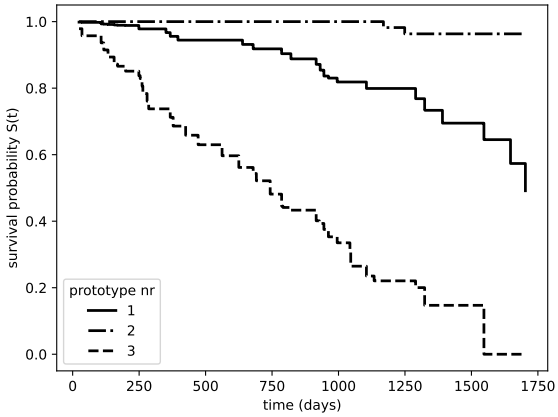


Figure 4: Visualization of the survival curves corresponding to the prototypes: The prototype labels.

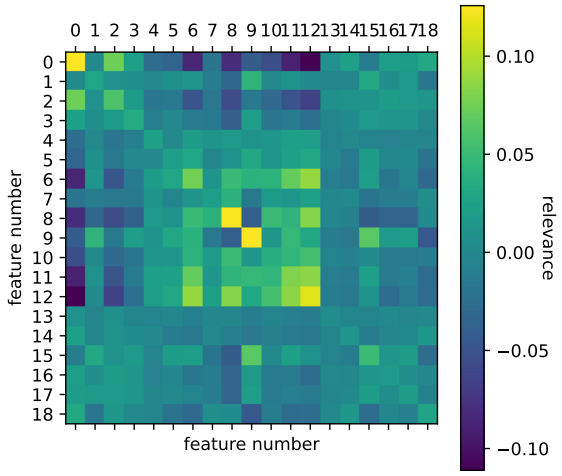
elements). It reveals disparities in the value of certain features for clustering and prediction. Figure 5b focuses on the matrix's diagonal, emphasizing individual feature significance. Notably, treatment in the Copenhagen hospital (0), histological stage (8,9), and age (12) emerge as key features. These findings could be provided to medical specialists for clinical interpretation.

Interestingly, the treatment (5) appears less consequential in our study, aligning with the original authors' findings that detected no significant univariate difference between the groups [43].

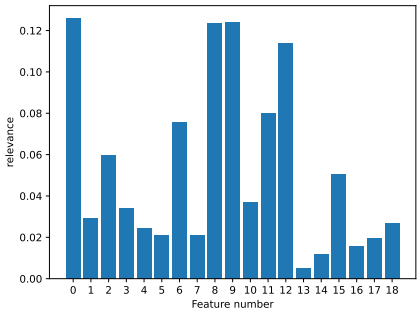
Figure 5c shows the sorted eigenvalues of the relevance matrix. In our analysis the first two eigenvectors cover 83% of the variance.

8. Conclusion

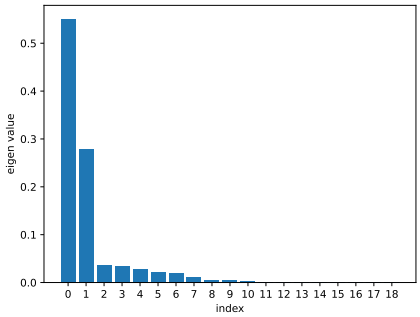
Clustering has limitations in survival analysis due to the censored nature of survival data and the need to leverage available survival time information. To address these limitations, we proposed a novel method called survivalLVQ, which adapted Learning Vector Quantization (LVQ) to the survival analysis setting. LVQ is a supervised clustering algorithm that partitions data points into clusters based on their similarity and class information. However, LVQ had



(a) Visualisation of the relevance matrix



(b) Diagonal elements of the relevance matrix



(c) Sorted eigenvalues of the relevance matrix

Figure 5: Plots showing elements and properties of the relevance matrix (Λ)

not previously been extended to survival analysis, and our study aimed to fill this gap.

Our proposed method, SurvivalLVQ, leverages the clustering power of LVQ to group individuals based on their survival probabilities. It assigns a survival curve to each cluster based on the collective survival behavior of the individuals within it. Additionally, it can predict individualized survival curves for individuals located in-between clusters by computing a weighted average of adjacent cluster survival curves.

When comparing clustering methods, SurvivalLVQ consistently outperformed the competing clustering methods across various cluster sizes, demonstrating higher scores in C-index, IPCW C-index, integrated IBS, and log-rank scores. This further emphasizes the usefulness of SurvivalLVQ in capturing survival patterns within the data.

In terms of predictive performance, SurvivalLVQ showed competitive results, especially when compared to clustering methods like K-means, Survival trees, and SCC. It achieved a lower average rank for several performance metrics, indicating better predictive performance in capturing survival patterns. While CoxPH showed the best overall performance, SurvivalLVQ still exhibited promising results compared to other clustering methods, highlighting its usefulness in survival analysis. However, it is important to note that CoxPH and RSF outperformed SurvivalLVQ across all metrics.

Finally, in the example analysis of the PBC3 dataset we showed that SurvivalLVQ can offer crucial insights into survival patterns. The clear visualizations simplify complex data, making it easier to interpret. Through this demonstration, we underscore the potential of SurvivalLVQ as a robust tool for survival analysis, capable of effectively analyzing complex datasets.

As for future research directions, our approach used K-means to initialize prototype points and the identity matrix for initial feature importance can be optimised. Given the strong results shown by SCC, it is worth considering whether starting with CoxPH scores for feature importance can further enhance SurvivalLVQ's performance, or at least speed up convergence. Additionally, our choice of Euclidean distance may not be optimal, especially given that many variables in the datasets were categorical. Future work should consider alternative metrics to refine the method's effectiveness.

Declaration

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work re-

ported in this paper. During the preparation of this work the authors used ChatGPT (OpenAI) in order to improve readability and language. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Acknowledgement

The authors acknowledge the support from the Flemish Government (through the AI Research Program) and the Research Fund Flanders (project G0A2120N).

References

[1] S. Hirano, X. Sun, S. Tsumoto, Comparison of clustering methods for clinical databases, Information Sciences 159 (3-4) (2004) 155–165.

[2] M. S. Islam, M. M. Hasan, X. Wang, H. D. Germack, M. Noor-E-Alam, A systematic review on healthcare analytics: application and theoretical perspective of data mining, in: Healthcare, Vol. 6, MDPI, 2018, p. 54.

[3] H. Alashwal, M. El Halaby, J. J. Crouse, A. Abdalla, A. A. Moustafa, The application of unsupervised clustering methods to alzheimer's disease, Frontiers in computational neuroscience 13 (2019) 31.

[4] F. Doshi-Velez, B. Kim, Towards a rigorous science of interpretable machine learning, arXiv preprint arXiv:1702.08608 (2 2017).

[5] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, Artificial Intelligence 267 (2019) 1–38.

[6] T. Kohnen, Learning vector quantization for pattern recognition, Technical report no. TKK-F-A601, Helsinki., Tech. rep. (1986).

[7] P. Schneider, M. Biehl, B. Hammer, Adaptive relevance matrices in learning vector quantization, Neural computation 21 (12) (2009) 3532–3561.

[8] M. Kaden, K. S. Bohnsack, M. Weber, M. Kudła, K. Gutowska, J. Blazewicz, T. Villmann, Learning vector quantization as an interpretable classifier for the detection of sars-cov-2 types based on their rna sequences, Neural Computing and Applications 34 (2022) 67–78.

[9] M. Nilashi, R. A. Abumalloh, H. Ahmadi, S. Samad, A. Alghamdi, M. Alrizq, S. Alyami, F. K. Nayer, Electroencephalography (EEG) eye state classification using learning vector quantization and bagged trees, Heliyon 9 (4) (2023) e15258.

[10] R. van Veen, N. R. B. Tamboli, S. Lövdal, S. K. Meles, R. J. Renken, G. J. de Vries, D. Arnaldi, S. Morbelli, P. Clavero, J. A. Obeso, M. C. Oroz, K. L. Leenders, T. Villmann, M. Biehl, Subspace corrected relevance learning with application in neuroimaging, Artificial Intelligence in Medicine 149 (2024) 102786.

[11] B. J. Jain, D. Schultz, Asymmetric learning vector quantization for efficient nearest neighbor classification in dynamic time warping spaces, Pattern Recognition 76 (2018) 349–366.

[12] E. Ahlqvist, P. Storm, A. Käräjämäki, M. Martinell, M. Dorkhan, A. Carlsson, P. Vikman, R. B. Prasad, D. M. Aly, P. Almgren, et al., Novel subgroups of adult-onset diabetes and their association with outcomes: a data-driven cluster analysis of six variables, The lancet Diabetes & endocrinology 6 (5) (2018) 361–369.

[13] M. B. Eisen, P. T. Spellman, P. O. Brown, D. Botstein, Cluster analysis and display of genome-wide expression patterns, Proceedings of the National Academy of Sciences 95 (25) (1998) 14863–14868.

[14] S. J. Shah, D. H. Katz, S. Selvaraj, M. A. Burke, C. W. Yancy, M. Gheorghiade, R. O. Bonow, C.-C. Huang, R. C. Deo, Phenomapping for novel classification of heart failure with preserved ejection fraction, Circulation 131 (3) (2015) 269–279.

[15] E. Bair, R. Tibshirani, Semi-supervised methods to predict patient survival from gene expression data, PLoS biology 2 (4) (2004) e108.

[16] S. Gaynor, E. Bair, Identification of relevant subtypes via preweighted sparse clustering, Computational statistics & data analysis 116 (2017) 139–154.

[17] J. Luo, L. Xie, H. Yang, X. Yin, Y. Zhang, Machine learning for time-to-event prediction and survival clustering: A review from statistics to deep neural networks, in: BenchCouncil International Symposium on Intelligent Computers, Algorithms, and Applications, Springer, 2023, pp. 174–192.

[18] E. Xia, X. Du, J. Mei, W. Sun, S. Tong, Z. Kang, J. Sheng, J. Li, C. Ma, J. Dong, et al., Outcome-driven clustering of acute coronary syndrome patients using multi-task neural network with attention., in: MedInfo, 2019, pp. 457–461.

[19] S. C. Mouli, L. Teixeira, J. Neville, B. Ribeiro, Deep lifetime clustering, arXiv preprint arXiv:1910.00547 (2019).

[20] P. Chapfuwa, C. Li, N. Mehta, L. Carin, R. Henao, Survival cluster analysis, in: Proceedings of the ACM Conference on Health, Inference, and Learning, 2020, pp. 60–68.

[21] L. Manduchi, R. Marcinkevičs, M. C. Massi, T. Weikert, A. Sauter, V. Gotta, T. Müller, F. Vasella, M. C. Neidert, M. Pfister, et al., A deep variational approach to clustering survival data, arXiv preprint arXiv:2106.05763 (2021).

[22] D. Nova, P. A. Estévez, A review of learning vector quantization classifiers, Neural Computing and Applications 25 (2014) 511–524.

[23] A. Sato, K. Yamada, Generalized Learning Vector Quantization, Advances in neural information processing systems 8 (1996) 423–429.

[24] R. van Veen, Analysis of missing data imputation applied to heart failure data, Ph.D. thesis, Faculty of Science and Engineering (2016).

[25] E. Eirola, G. Doquire, M. Verleysen, A. Lendasse, Distance estimation in numerical data sets with missing values, Information Sciences 240 (2013) 115–128.

[26] E. L. Kaplan, P. Meier, Nonparametric estimation from incomplete observations, Journal of the American statistical association 53 (282) (1958) 457–481.

[27] P. Schneider, M. Biehl, B. Hammer, Distance learning in discriminative vector quantization, Neural computation 21 (10) (2009) 2942–2969.

[28] G. W. Brier, et al., Verification of forecasts expressed in terms of probability, Monthly weather review 78 (1) (1950) 1–3.

[29] E. Graf, C. Schmoor, W. Sauerbrei, M. Schumacher, Assessment and comparison of prognostic classification schemes for survival data, Statistics in Medicine 18 (17-18) (1999) 2529–2545.

[30] D. Arthur, S. Vassilvitskii, K-means++ the advantages of careful seeding, in: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, 2007, pp. 1027–1035.

[31] E. Drysdale, SurvSet: An open-source time-to-event dataset repository, arXiv preprint arXiv:2203.03094 (2022).

[32] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, M. S. Lauer, Random survival forests, Annals of Applied Statistics 2 (3) (2008) 841–860.

[33] D. R. Cox, Regression models and life-tables, Journal of the Royal Statistical Society: Series B (Methodological) 34 (2) (1972) 187–202.

[34] M. R. Segal, Regression trees for censored data, Biometrics (1988) 35–47.

[35] M. LeBlanc, J. Crowley, Survival trees by goodness of split, Journal of the American Statistical Association 88 (422) (1993) 457–467.

[36] D. P. Doane, L. E. Seward, Measuring skewness: a forgotten statistic?, Journal of statistics education 19 (2) (2011).

[37] S. Pölsterl, scikit-survival: A library for time-to-event analysis built on top of scikit-learn, Journal of Machine Learning Research 21 (212) (2020) 1–6.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[39] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, Advances in neural information processing systems 32 (2019).

[40] M. LeKander, M. Biehl, H. de Vries, Empirical evaluation of gradient methods for matrix learning vector quantization, in: 2017 12th international workshop on self-organizing maps and learning vector quantization, clustering and data visualization (WSOM), IEEE, 2017, pp. 1–8.

[41] H. Uno, T. Cai, M. J. Pencina, R. B. D'Agostino, L.-J. Wei, On the c-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data, Statistics in medicine 30 (10) (2011) 1105–1117.

[42] R. Peto, J. Peto, Asymptotically efficient rank invariant test procedures, Journal of the Royal Statistical Society. Series A (General) 135 (2) (1972) 185–207.

[43] M. Lombard, B. Portmann, J. Neuberger, R. Williams, N. Tygstrup, L. Ranek, H. Ring-Larsen, J. Rodes, M. Navasa, C. Trepo, et al., Cy-

closporin a treatment in primary biliary cirrhosis: results of a long-term placebo controlled trial, Gastroenterology 104 (2) (1993) 519–526.

[44] T. R. Fleming, D. P. Harrington, A class of hypothesis tests for one and two sample censored survival data, Communications in Statistics-Theory and Methods 10 (8) (1981) 763–794.