Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Igor Andriushchenko

Semi-Supervised Learning Vector Quantization method enhanced with regularization for anomaly detection in air conditioning time-series data

Master's Thesis
Espoo, May 30, 2016

Supervisor: Professor Aristides Gionis
Advisor: Yoan Miche, D.Sc. (Tech.)

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Aalto University
School of Science

ABSTRACT OF
MASTER'S THESIS

Author:	Igor Andriushchenko	
Title:	Semi-Supervised Learning Vector Quantization method enhanced with regularization for anomaly detection in air conditioning time-series data	
Date:	May 30, 2016	**Pages:** 82
Major:	Information Computer Science	**Code:** T30618
Supervisor:	Professor Aristides Gionis	
Advisor:	Yoan Miche, D.Sc. (Tech.)	

Researchers of semi-supervised learning methods have been developing the family of Learning Vector Quantization models which originated from the well-known Self-Organizing Map algorithm. The models of this type can be characterized as prototype-based, self-explanatory and flexible.
The thesis contributes to the development of one of the LVQ models - Semi-Supervised Relational Prototype Classifier for dissimilarity data. The model implementation is developed based on the related research work and thesis author findings, and applied to the task of anomaly detection from a real-time air condition data. We propose a regularization algorithm for gradient descent in order to achieve better convergence and a new strategy for initializing prototypes. We develop an innovative framework involving a human expert as a source of labeled data. The framework detects anomalies of environment parameters in both real-time and long-run observations and updates the model according to findings.
The data set used for experiments is collected in real-time from sensors installed inside the Aalto Mechanical Engineering building located at Otakaari, 4, Espoo. Installation was done as a part of the project of VTT and Korean National Research Institute. The data consists of 3 main parameters – air temperature, humidity and CO_2 concentration. Total number of deployed sensors is around 150. One month recorded data observations contains approximately 1.5M of data points.
The results of the project demonstrate the efficiency of the developed regularized LVQ method for classification in given settings. Its regularized version generally overperforms its parent and various baseline methods on air conditioning, synthetic and UCI data. Together with the proposed classification framework, the system has shown its robustness and efficiency and is ready for deployment to a production environment.

Keywords:	Semi-Supervised Learning, Anomaly Detection, Time-Series, Learning Vector Quantization
Language:	English

Acknowledgements

Starting and implementing a project which involves theoretical modeling and research, practical integration and application - is never an ordinary task. Complexity estimates grow and evolve together with the progress on the topic. At the same time, the learning from the every singe obstacle and challenge on this way is truly beneficial.
I would like to share my sincere gratefulness to my supervisor Prof. Aristides Gionis and advisor D.Sc. Yoan Miche who guided me on this long journey through the exciting area of the Semi-Supervised Learning research, helped with priceless advices and spent hours helping this project to be finished.
Also, I would like to thank Prof. Jukka Nurminen and D.Sc. Sanja Stepanovic who made possible to get the access to the real-time sensor data after introducing me to the inspiring CIVIS project and its partners from VTT Research Institute and Korea Electronics Technology Institute. Thank you, Youngmin Ji for your efforts to maintain the data back-end and make observations available for my 24/7 experiments.
Finally, I must express an endless gratefulness to my biggest passion and inspiration in this world - my wife Maria Faticov, who has been providing her valuable opinions and motivating me every day.

Espoo, May 30, 2016

Igor Andriushchenko

This page was intentionally left blank.

Contents

Chapter 1

Introduction

In the modern world, the amounts of data generated in various areas of life have never been so big [50]. It is doubling every two years and will reach the total size of 44 trillion gigabytes by 2020 [51]. With the current state of the art data collection principles [52,53,54], data from any event related to an observable system, can be recorded and collected for further processing. However, an ability to get more data has downsides: the quality of retrieved observations is different for various variables [56]; most of the observations remain unlabeled; amounts of recorded data require engineers to seek for the balance of efficiency-accuracy compatible to the particular task [55,57].
Challenges also come from an increase in a storage space required for storing data and computational power needed to process it [52,53]. Aggarwal et al. in their book [57] point out that the modern context requires a data processing system to retrieve qualitative data and then utilize this unlabeled data on-the-fly without applying any of heavy-weight methods.
Data coming from various sensors that produce measurements of numerous characteristics within a time domain and send them for processing, is a great example of a challenging and technology demanding information source, in our opinion, is worth to be researched and experimented with.
For processing this type of data a pipeline needs to deal with constantly arriving data records with unknown labels and process them before the new records are started being added to the queue. Analytics on this data has to occur in real-time [58] generating a result that itself is a product of efficiency/accuracy trade-off [55,57]
According to the mentioned requirements, the question can be set - how to improve data analytics accuracy and avoid decreasing an efficiency which in this setting must be as close to real-time execution as possible? One of the possible alternatives is to use novel semi-supervised learning methods [1,25,60] which utilize a few labeled points for which the ground truth is known and make more accurate predictions from unlabeled data [59].
Semi-Supervised Learning (SSL) combines insights from labeled and unlabeled data in order to extract extra information from a dataset and therefore provide better classification results [1,60]. It requires much less of human effort comparing to supervised learning when all points in the training data are labeled [60], and fits the modern big data tendency - when the majority of the data arrives unlabeled [50,51,57].
SSL provides a wide variety of methods that can be applied to any task depending on the data structure, accuracy/efficiency requirements [60]. The classic literature [25,59,60] divide them into five main groups depending on underlying

assumptions about the data: generative models SSL, low-density separation SSL, graph-based methods SLL, co-training SSL methods, and self-training methods. We provide a wider overview of each of the classes in the next chapter.

- **Generative model method** is the oldest SSL method and it assumes that given large amounts of unlabeled data there exists a mixture distribution which the data is drawn from [60]. It is known for a good performance in case of well-clustered data [59].
- **Low-density separation** is the method based on the idea that the decision boundary should be located in the data low-density area [60,61]. A well-known example is TSVM - Transductive Support Vector Machines [61].
- **Graph-based** methods rely on the assumption that high-dimensional data is located on a low-dimensional manifold [62] when representing the data as a set of nodes and edges correspond to the distances between nodes.
- In the **Co-training method**, the data features are divided into two sets that are capable of being used as training data for two separate classifiers. Once predictions are made, the results of one classifier are included into the training set of the another [63]. The method demonstrates the best performance when features can be intuitively split into two separate sets [59].
- **Self-training** method is defined by the classifier that is firstly trained on labeled data. Then unlabeled data is fed to it, and data points with the highest classification confidence are added to labeled dataset, and the procedure repeats [64,65]

1.1 Thesis aims and goals

The project described in this thesis is aiming at solving a two-stage real-world problem. First, how to keep optimal conditions of a building indoors environment using multi-dimensional data from a set of sensors for detecting anomalies in environmental conditions. The setting also includes the human expert who is supervising the automated system by providing the labeled dataset. All the data from sensors comes unlabeled and needs to be classified as quick as possible. Here comes the second goal of the project - to experiment with not widely known clustering approach which can show good results if being put into requirements of big data processing systems [52,55,57]
The state-of-art time-series classification methods include such methods as Hidden Markov Models [67], Dynamic Time Warping-based methods [68,69,70],

Neural Networks [71] and others. In [66] authors note that the described methods usually require being trained on large labeled dataset to properly extract underlying knowledge. This fact makes it practically impossible to use one of the classic models to classify the sensors' data from this project.
Therefore, we propose the new use of a relatively young method - Semi-Supervised Learning Vector Quantization with conformal prediction [2] which we further refer to as SSL-RPC (LVQ). This method or any of its derivatives has yet not been applied to the real-time time-series data classification, as far as we know it.
The Learning Vector Quantization [9,23] family of algorithms recently expanded to a wide number of algorithms [1,2,6,7,10,11] suitable for different kinds of tasks. Originally, LVQ was described by a Finnish professor Teuvo Kohonen as an efficient prototype-based method aiming at finding near-optimal decision borders between classes [9,23]. The method was adopted to practical use by Sato and Yamada in 1995 [1] and named Generalized Learning Vector Quantization. Since that time, the family of models has been significantly extended with many new methods built on the top of GLVQ, and what is especially important, the semi-supervised extension has been created [2]. Supervised [6,7,10] and semi-supervised [2] GLVQ-based methods demonstrated their efficiency in cases of common and dissimilarity data representation.
Among others, one of the useful aspects of GLVQ-method extensions is the ability to deal with data in non-Euclidean space [6] and high efficiency when working with data represented in the form of dissimilarities [7].
The SSL-RPC-LVQ [2] method is an SSL method which we found suitable for the task as it shown excellent performance in when applied to standard datasets (CHROMO [72], SWISS [73] etc). Since it is a prototype-based, the final representation can be observed and the learning dynamic can be analyzed [9]. This is the key feature to enable closer expert-model interaction which is essential for our project - we aim at involving human not only into providing the initial dataset and at the same time making him correcting model's behavior when a clear mistake in classification is spotted. Finally, SSL-RPC-LVQ by the design [2,6] is able to produce instant classifications whit relatively small datasets.
Regarding the fact that we are not familiar with a published research on the efficiency and practical applications of LVQ-RPC-based methods to the stream data taken from time-series, in this project we explore the newest extensions of LVQ methods [2,6,7] and adopt RGLVQ-SSL extension [2] for applying to the task of semi-supervised clustering of time-series data that is being retrieved in real-time from a set of sensors. The developed model is incorporated into the outliers detection algorithm that we propose and it can be used in industry for a real-time monitoring environment conditions inside buildings.

1.2 Proposed framework

We develop a setup that includes a scientific contribution to the described classification method and employs it in order to solve a real-world industrial problem of optimizing indoor environment conditions. The problem can be decomposed into two levels of abstraction - first, how to efficiently classify huge amounts of data coming from sensors in the real-time; second how to process the obtained labels and produce a decision indicating whether the current data points' set corresponds to expected values or it is an outlier.
The framework was developed to be used in the future "as-is" by companies operating big office buildings to minimize expenses and losses caused by inefficient use of ventilation and heating systems. It collects data from the sensors, sends it to the cloud. The anomaly detection model analyzes the data and produces a decision. A human expert provides the system with initial labeling of a few points and can correct wrongly classified observation. The framework is fully designed and is ready to be installed on any plot with the access to data (Fig. 1.1).

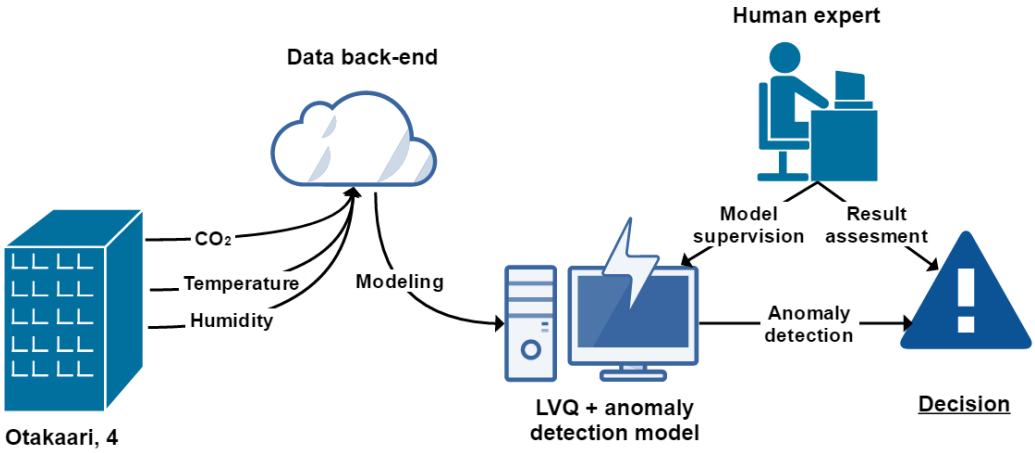

Figure 1.1: The big picture of the project. Data is fetched from the cloud, classified and anomalies are reported. A human expertise enhances the quality of predictions and can vary from a high level of involvement to only providing initial labeling to the supervised dataset.

In this more detailed setting (see Fig. 1.2), the expert has two points of a possible interaction with the system. Firstly, he/she provides the model with the labeled data. Once the model generated a label for a data point, the outlier detection algorithm is being run to produce a decision. If the expert is observing the system at this moment, he can correct the decision pointing that it was mistaken and providing the correct label. In this case, the label and data point is added to the labeled dataset automatically. Also, the expert has a right to modify the labeled dataset at any time. We must note that we

do not expect the expert to watch the system constantly and correct outlier detection outputs in real-time.

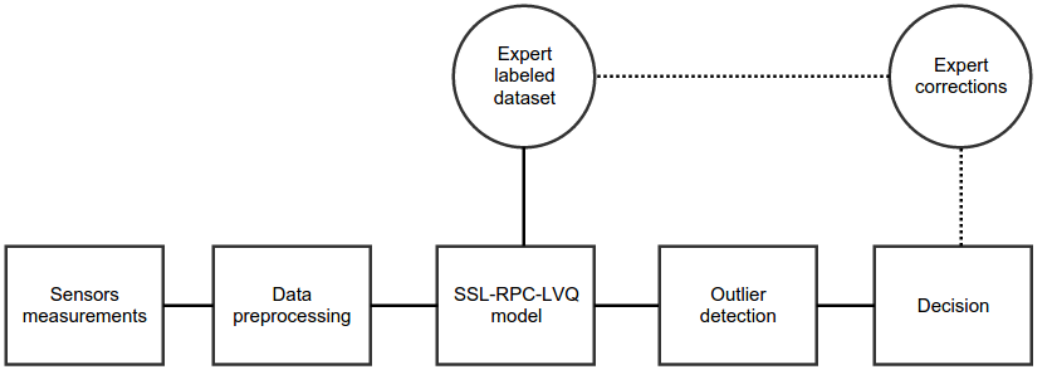

Figure 1.2: General project workflow

Using the proposed framework as a reference, we aim at reaching another motivating point for starting this project - satisfying a scientific interest of exploring abilities of the SSL-RPC method [2] and pushing it to the new direction such as a stream data instant classification. Furthermore, we aim at covering the wide range of research questions related to SSL-RPC which has not been previously covered in [2].
We start with the motivation of the method selection, the representation of data, time-series handling technique and other aspects. We compare SSL-RPC performance to the state-of-art models on synthetic and UCI data [112]; demonstrate its efficiency depending on the labeled/unlabeled data sizes; take a closer look and illustrate the model mechanics on the prototype fitting and conformal prediction [2] steps.
To overcome algorithm's issues related to convergence, we introduce the new regularization parameter λ which demonstrates the significant improvement of the convergence rate comparing to the original SSL-RPC algorithm.
We propose our implementation for the procedure of creating new SSL-RPC prototypes and integrate it to the general model workflow. According to our knowledge, this procedure has not previously been described in the literature.
Finally, we provide an analysis of algorithm's run-time and propose a technique for prototype initialization which allows the first iteration to converge two times faster comparing to the original initialization technique. We explore the algorithm's major disadvantage when it operates with big datasets. Altogether, our contribution allows other researchers to analyze what worked well and what did not in the case of SSL-RPC providing a direction of a possible research in the future.
Finally, we have to note that the selected family of methods (SSL-RPC) has proven its efficiency in general, although it has not been widely used in the field and is currently developed by a relatively small group of researchers [99]. The main method [2] that we base our classifier on has only 7 citations counted

by *scholar.google.com*. Also, as far as we know, LVQ classifiers are not widely present in existing software open-source implementations and libraries. Therefore, we decided to publish semi-supervised RPC and supervised RPC-LVQ classifiers as open-source packages for Python and share them publicly. We hope this will encourage a wider audience to employ LVQ methods in new directions and lead them to new findings.

1.3 Structure

The **section 2** provides a short motivation on the task that the project solves. It explores a scientific background of SSL methods and a history of LVQ development including a short taxonomy of existing LVQ-based methods. It also covers the basics of conformal prediction [2] with a relation to all main concepts used in the model algorithm development.
The data and its origins are described in the **section 3**. We provide an information about the sensors setup and the collaborative project that played the core role in building a data back-end and delivering the data to API.
The developed algorithm and all its essential components such as Stochastic gradient descent with regularization, Prototype generation, Conformal prediction procedure and others are described in **section 4**.
In **section 5**, we present conducted experiments with real, synthetic and UCI data and corresponding results.
We finalize the thesis with the discussion on findings and a further research direction in **section 6**.

Chapter 2

Background

2.1 Overview of Machine Learning

Various definition of Machine Learning exist in the literature, therefore, for means of simplicity, we refer to the following one. Mitchell et al. in [74] defined that given a task, training experience and performance measure, the program is considered as the one that learns if its performance improves with experience. In the earlier works (before 2000-s), two types of learning were distinguished – supervised and unsupervised [75].
According to Mitchell [74], Supervised Learning is defined as approximating an unknown function f from the experience represented in the form of training examples. The performance is measured my comparing actual values (labels) to predicted by the model. The approximated function f can be used to find unseen values (regression) or unknown labels for data (classification) [76].
Formally speaking, Supervised Learning is the task of utilizing the training set of pairs (x_i, y_i) for learning a mapping from x to y and approximating it by the means of an underlying function. Here $x_i \in X$ are training examples, and $y_i \in Y$ is set of target labels [25,74]. This type of learning is also referred to as "learning with a teacher".
The opposite type of learning in terms of given experience is unsupervised learning. In this case, labels of training examples always remain unknown. The aim of Unsupervised Learning is to approximate a function representing the data's hidden structure without an external teacher [77].
In 2000-s the new separate area of Machine Learning arose – Semi-Supervised Learning [25]. It has no requirement to operate on a fully labeled dataset which makes it less human (teacher) resource-demanding, and benefits from finding the underlying data distribution [59].

2.2 Semi-Supervised learning

Semi-Supervised (SSL) learning is the Machine Learning task lying in between of the Supervised and Unsupervised Learning. It utilizes both unlabeled data and supervised information for training [25]. Given dataset $X = (x_1, .., x_i)$, of i points, assume it can be divided into two parts. First, $X_l = (x_1, .., x_l)$ for which labels $Y_l = (y_1, .., y_l)$ are known, and l is the size of the labeled data. And the second part, $X_u = (x_{l+1}, .., x_{l+u})$ for which there are no known labels, and u is the size of unlabeled data.
We assume that unlabeled data X_u was sampled from the same distribution as

the labeled data X_l, so the knowledge that can be extracted from the unlabeled data is relevant to the labeled part, and vice-versa. This is the key prerequisite to the data [25]. Also, in this project we are counting on the following assumption about the SSL:

- Cluster assumption: if two points are located in the same cluster of the data, they belong to the same class.
- Low-density separation assumption: the decision boundary should be located in a low-density region.
- Semi-Supervised Smoothness assumption: if two points x_1 and x_2 located closely to each other in a high-density region, it is also true for their corresponding outputs y_1 and y_2.
- The Manifold assumption: The high-dimensional data lie on a low-dimensional manifold.

There exist 5 main types of SSL algorithms according to [59], while [25] distinguish only 3 main types. We will follow the 5 methods notation since it provides more insights on the various features and abilities of SSL.

2.2.1 Generative models in SSL

The SSL Generative Models perform a classification basing on extraction of the additional information from the marginal density or as clustering with additional information (i.e presence of labels for some subset of points) [25].
A generative modeling as part of SSL can be done as learning a class conditional density $P(x|y,\theta)$ and a class prior $P(y|\theta)$ for each output y. Using the obtained values, for a test point x according to generalized Bayes rule compute $P(y|x,\theta) \propto P(x|y,\theta)P(y|\theta)$ [83].
The advantage of the generative modeling in SSL is based on its knowledge about the structure of the problem [25]. The classic literature describes EM [84], kernel-based [85] generative models used for unsupervised learning. Their successors were also adopted for the needs of SSL. The generative model based on EM algorithm of a mixture of multinomial was introduced for text classification [86]. The mixture model applying the maximum entropy principle when using a discriminative training [87,88].
A strength of the generative approach is that knowledge of the structure of the problem or the data can naturally be incorporated into learning via modeling it. However, learning from the unlabeled data is possible only in the case of the correct mixture model [89]. This type of models is highly dependent on the construction of the mixture, as well as on the nature of the dataset – clusters should be well-defined [59].

2.2.2 Low-density separation models in SSL

This type of SSL models is based on the cluster assumption about any SSL model.

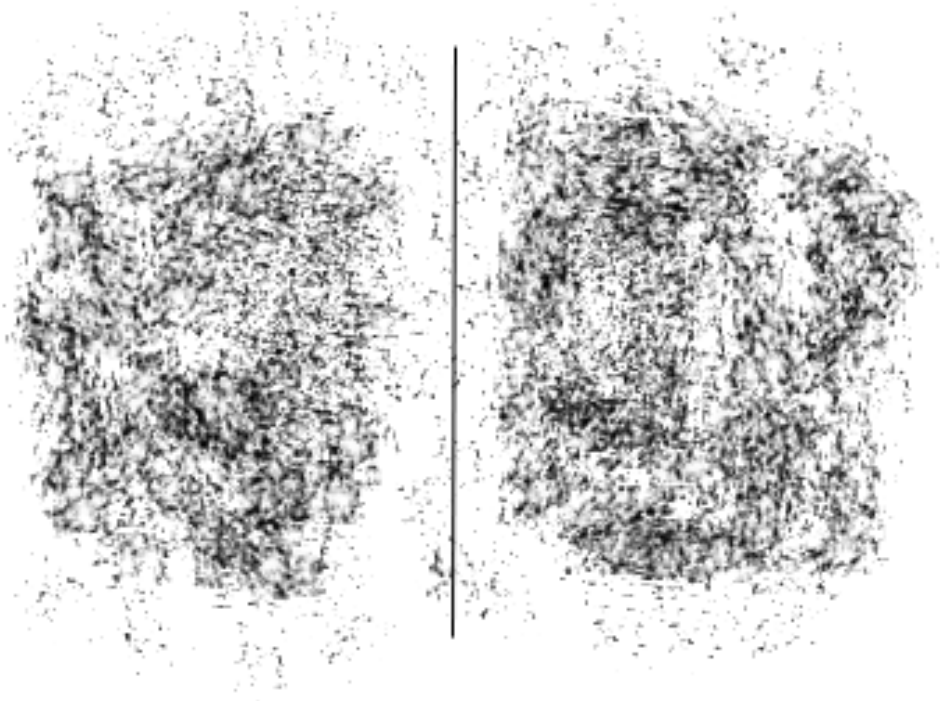

Figure 2.1: The decision boundary is located in the area with a low density

It says that the decision boundary between two clusters lies in the low-density area (or alternatively, cannot cross high-density areas) [25], see Fig. 2.1 This assumption shall generally be held for any SSL algorithm while the low-density separation models utilize this principle above all [90]. A well-known example of a model of this type is TSVM (Transductive SVM) [91]. It implements the transductive learning principle (as the opposite to inductive in Generative Models) which is based on utilizing labeled points and finding the hyperplane which is located as far from unlabeled points as possible. For solving this type of tasks, max-margin algorithms are most suitable, which are best-known as Support Vector Machines. In general, TSVM algorithm as seeking for such labeling for unlabeled data that the margin boundaries between clusters are maximal [91], i.e. Fig. 2.2.

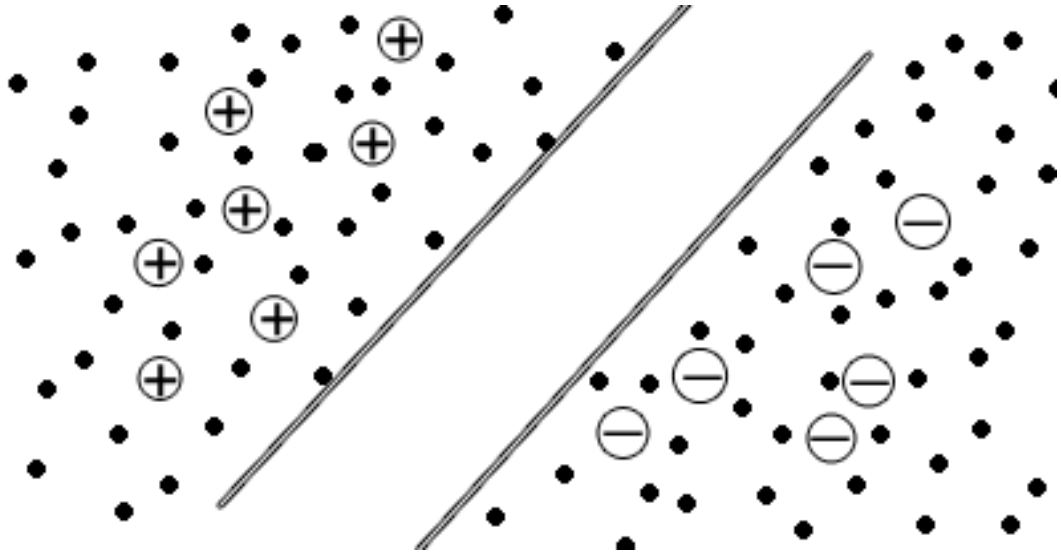

Figure 2.2: TSVM's final classification of points and margin boundaries fitted to data

In [25,90] authors described existing optimization algorithms used for solving the problem and mention it as a non-convex optimization problem which can be measured as NP-hard [59] and is also difficult to optimize due to slipping to local minima [83]. One of the first successful implementation of the TSVM algorithm is the SVM-light software package [92].

2.2.3 Graph-based methods

Instead of solving a non-ordinary task of finding a low-density separation, the graph-based methods estimate a function f that is being close to labels of the labeled data and at the same time is smooth on the whole graph [59].
Generally, the semi-supervised learning problem in the graph setting can be seen as a task of finding such labeling that is consistent with the initial labeling and at the same time fits the graph structure – as combination of edges and corresponding weights [25].
First, we measure consistency with initial labeling $\hat{Y} = (\hat{Y}_l, \hat{Y}_u)$ as: $\sum_{i=1}^{l}(\hat{y}_i - y_i)^2 = ||\hat{Y}_l - Y_l||^2$.
At the same time, the geometrical consistency can be found as following:

$$\frac{1}{2}\sum_{i,j=1}^{n} W_{ij}(\hat{y}_i - y_j)^2 = \hat{Y}^T L \hat{Y}$$

where $L = D - W$ is un-normalized Laplacian of the graph, W – the similarity matrix.
Given these equations, various methods aim at labeling cost and fitting it to the geometrical consistency, such as Laplacian-based regularization [93], Tikhonov regularization [94] and another type of graph-based methods for semi-supervised learning.

A different type of graph-based methods is represented by Label Propagation / Label Spreading algorithms [44,45]. Label propagation firstly defines a pairwise relationship W on the dataset X, where given a graph $G = (V, E)$ V represents the dataset X, edges E are weighted by W [45]. Nearest neighbors are joined by edges, weights are set according to the strength of correlation [25]. Labeled nodes propagate their labels to neighbors and update weights until convergence.
In general, Graph-based methods are most suitable to tasks which data can intuitively be represented in the form of graphs, i.e. computer vision, speech processing, text recognition. [95]

2.2.4 Co-training methods

The co-training methods [96] are based on the following assumptions:

- data features can be split into two sets
- it is possible to train a good classifier from each of these two sets
- two sets are conditionally independent

First, two classifiers are trained from labeled data of the corresponding feature set. Afterward, each of the models classifies the unlabeled data and the classifier spreads a small amount of data points classified with a high confidence to each other (see Fig. 2.3). The models are retrained with updated datasets, the process repeats until the convergence.

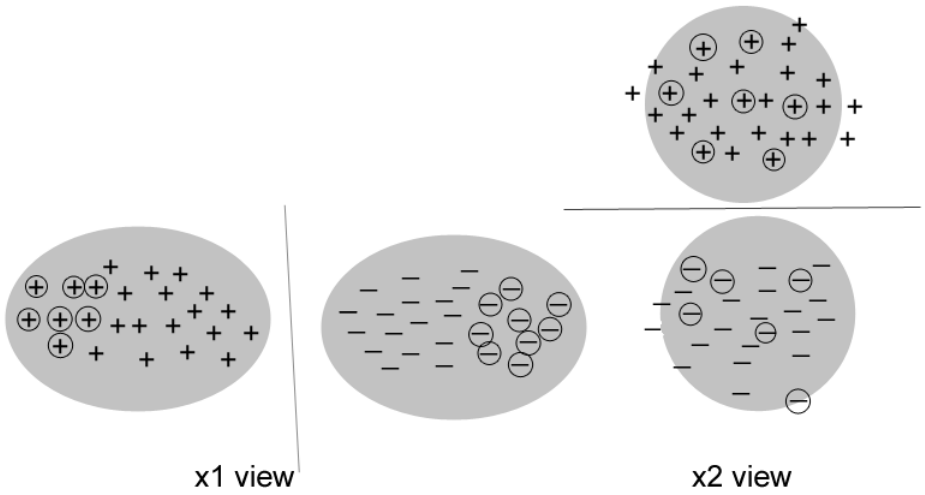

Figure 2.3: Conditional independence of features split assumption in co-training. High-confident data points in $x1$ view are randomly disposed on $x2$

The further development of Co-learning was directed into exploring conditions under which it performs better and relaxing the initial assumption about splitting features [59].

2.2.5 Self-training methods

This type of learning in SSL is not always considered as a separate sub-class of methods and is used as a wrapper for models of other classes [59]. The idea is as simple as training the classifier on labeled and then append classified points from the unlabeled set with the highest classification confidence, repeat until convergence.
This method was used for natural language classification [96], image classification [97] and Vector Quantization [2]. The actual method used as the base for developing this project's model is using an adaptation of the self-training principle when defining a low-confidence data cluster and creating a new cluster(s) from it with further appending the points to the prediction dataset.

2.2.6 Prototype-based methods

Prototype-based clustering operates on data points that are divided into a known number of clusters. Each cluster receives a representative point assigned to it, which either belongs to the inner cluster distribution or is an existing point of data itself. The prototypes positions are being iteratively improved by optimizing a cost function aiming at finding the best representation of clusters [78]. This method family is known for its interpretability [79].
One of the best-known prototype-based methods is Self-Organizing Map [24] developed by T. Kohonen. In this method, an artificial neural network is constructed usually as a two-dimensional grid representing the equal- or high-dimensional input space of training samples. Competitive unsupervised learning takes place so that a data point is mapped to a neuron that is identified as the best-matching one basing on its model vector weight coefficient(s) (also called stimulus). The coefficient vectors of neurons can be viewed as prototypes representing the data. After performing iterative updates and arranging prototypes positions, the best fitting model prototype vectors are found. The step of updating prototypes is a vector quantization. The resulting map preserves a topology of the network response to the inputs.
Neural Gas [80] is the another technique that uses prototypes and learning vector quantization. The difference is that neighbor neurons relations are updated basing on ranked dissimilarities calculated in the original input space of data and prototypes [78]. Not only the winning neuron and also the highest ranked ones are being updated. The idea not to use the "winner-takes-all" rule proposed in the NG update provides a general Competitive Vector quantization framework which was widely used in unsupervised learning [78] such as Generative Topographic Mapping [81], Nonlinear Dimensionality Reduction [82] and others.

For the means of supervised classification, Learning Vector Quantization was proposed [1,9]. It developed into a variety of prototype-based methods such as GLVQ[1], SSL-LVQ [2] and many others. Nowadays, researchers describe prototype-based methods as an attractive and highly capable framework for all types of learning specifically suitable for high-dimensional data [78].

2.3 Learning Vector Quantization

LVQ method was firstly proposed by Professor Teuvo Kohonen as a method for fine-tuning the Self-Organizing Map [24] – a neural network-based technique for pattern-classification developed by the same author. Firstly used with SOM, LVQ was introduced as a nearest prototype-based supervised classification algorithm [9,98] able to deal with multi-class problems.
The general form of the LVQ is aimed at approximating clustering by setting and fitting prototypes vectors that are placed into the input space. For given training data $\mathbf{D} = (\mathbf{x}_i, \mathbf{y}_i)$, where $\mathbf{D} \in \mathbb{R}^N : \{1, ..., C\}$, and C is the number of classes in data, N – the number of dimensions, $\mathbf{x}_i$ – the observation vector, y – corresponding labels for observations.
LVQ is composed of a number of prototypes which are described as a set of weights in the weight space $\gamma_i \in \mathbb{R}^N$ and corresponding set of class labels $c(\gamma_i) \in 1, ..., C$. The similarity measure d^α is fixed for $\mathbb{R}^N$. The classification is being done by the rule "winner takes all". A data point $\mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^N$ is assigned with a label $c(\mathbf{x}) = c(\gamma_i)$ of the prototype i, where $d^\alpha(\mathbf{x}_i, \mathbf{x}) \leq d^\alpha(\mathbf{x}_j, \mathbf{x})$ hold for each and every $i \neq j$ – points are mapped correspondingly to the closest prototype's label [10].
The receptive field of the prototype γ_i is defined as

$$R^i = \{x \in \mathbf{X} \mid \forall \gamma_j (j \neq i) \rightarrow d(\gamma_i, x) \leq d(\gamma_j, x)\}$$

In the original version of LVQ, no cost function for optimization prototype positions was proposed. Further development of the LVQ field focused on defining cost functions and fitting prototype positions [99].
A few heuristic methods were proposed by Prof. Kohonen in [9]. LVQ of type 1 (LVQ1) was updating only the winner prototype without pushing one of the others back from the class of the winner [12].
A different approach was used in the LVQ of type 2 which was referred to as LVQ2. This method became de-facto a standard reference for researchers working on LVQ methods development [99].
The LVQ2 [9,23] method aims at finding an efficient separation between closest prototypes belonging to different classes. Let's assume, $\gamma_j(t)$ and $\gamma_i(t)$ are two closest prototype vectors to x and $\gamma_i(t)$ belongs to the same class as x, when $\gamma_j(t)$ belongs to the different class.
Kohonen in [9,23] describes the essence of the fitting LVQ algorithm through putting the two vectors into initially wrong positions, with a discrimination surface defined as a midplane of γ_i and γ_j. Then the symmetric non-zero

length windows are being introduced and put onto midplane. Then, corrections to position of γ_i and γ_j are made only in the case when x located in the window on the wrong side of the midplane [23] (see Fig. 2.4):

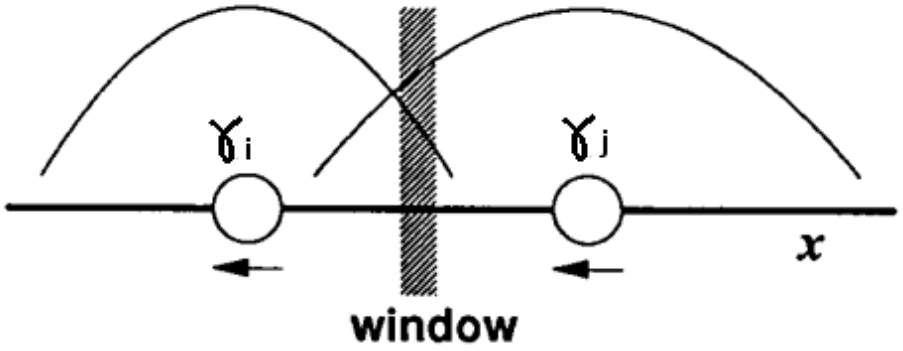

Figure 2.4: Fitting the window between two prototype vectors. The prototype position are shifted to fit x to the correct side of the midplane. The figure taken from [23] with modified notation

Prof. Kohonen proposed to update prototype positions with the following update rules:

$$\begin{aligned} \gamma_i(t+1) &= \gamma_i(t) - \alpha(t)(x - \gamma_i(t)) \\ \gamma_j(t+1) &= \gamma_j(t) - \alpha(t)(x - \gamma_j(t)) \end{aligned} \tag{2.1}$$

where $0 < \alpha(t) < 1$ and $\alpha(t)$ may decrease monotonically with time and $t = (0, 1, 2, ..)$ defines discrete-time formalism [12]
Following the update rule (Eq. 2.1), prototype vectors γ_i and γ_j are being moved such that the midplane moves in the direction of the limiting surface of the class distribution and therefore asymptotically matches the Bayesian decision border.
At the time of making this observation, Kohonen notes that the vector x is located within the window on the midplane of $\gamma_j(t)$ and $\gamma_i(t)$ if:

$$min(\frac{d_i}{d_j}, \frac{d_j}{d_i}) > \frac{1-\omega}{1+\omega} \tag{2.2}$$

where $d_i = |x - \gamma_i|$, $d_j = |x - \gamma_j|$ - distances from the data to each of the prototype vector, and ω is a window parameter $0 < \omega < 1$. Decision boundaries are being shifted to the Bayes limits, being moved closer to x if the prototype belongs to a correct class, and further - if the class is wrong [12]. Moving is performed as updating of prototype vector's γ coefficients.

2.4 Further development of Learning Vector Quantization

The LVQ2 algorithm was based on heuristics and fitting prototypes to data by pushing prototypes positions closer to the data or pulling them away. The mathematical reasoning of this process remains not-well defined according to [9].
In [9,20] it was proven that LVQ introduces a number of problems – slow convergence, and a possibility of converging in a local minimum. It was demonstrated in [20,21] that very slight changes to regularization coefficients in LVQ learning scheme led to unstable results.
Therefore, it became a starting point for the further research on the LVQ2 prototype-based methods, which aimed at finding non-heuristic solutions for the problem, improving techniques of fitting prototypes to data, achieving better performance and accuracy [99].

2.4.1 LVQ methods taxonomy

Three main directions of LVQ methods development were formed, according to the taxonomy proposed by Nova et al. in [99] (see Fig. 2.5).

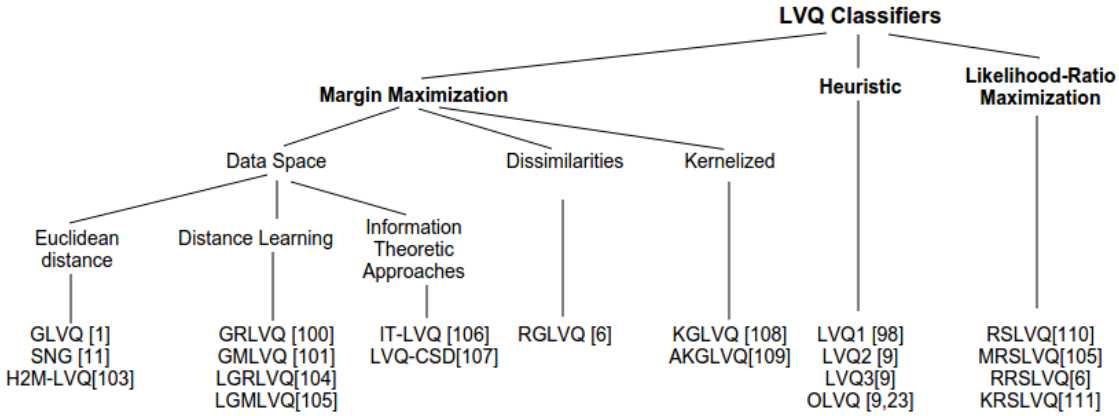

Figure 2.5: Taxonomy of LVQ methods. Original version proposed in [99], updated with methods related to the thesis and added corresponding references

- **Heuristic** type of LVQ methods unites originally proposed LVQ techniques based on heuristic Hebbian learning principles [9,23,98]. LVQ1 [98] updates only the winner prototype. LVQ2 [9] changes positions of a winner and the nearest prototype with a different label than the data point. LVQ3 [9] adds a stability factor to LVQ2 learning rules.
- **Margin maximization** methods operate with so-called hypothesis margin which represents the quantification of the distance that the classifier

hyperplane can be moved and no changes to classification rate are made [99].
The GLVQ method was the first to propose the hypothesis margin maximization cost function [1]. The most of the further effort in this area was directed at exploring GLVQ behavior in different data spaces.
Non-Euclidean distance measures between points and prototypes were taken for use in such methods as GRLVQ [100], GMLVQ [101] and others [24,104,105].
Information Learning-based techniques proposed a new measure for divergence based on Cauchy-Schwarz inequality [106]. This type of methods also uses fuzzy class labels in classifier training.
Kernelized methods as a subset of the Margin Maximization methods proposed to use kernels for LVQ classifiers in which a mapping function $\mathbf{\Phi}(.)$ is used to implement a nonlinear transformation from the data space $\mathbb{R}^N$ to a higher dimensional feature space $\mathbf{F}$ such that $\mathbf{\Phi} : \mathbb{R}^N \rightarrow \mathbf{F},\ x \rightarrow \Phi(x)$. The common choice of the kernel function is the Gaussian kernel [108].
Approaches based on computing relations between data, practically, dissimilarities in data and using it in LVQ are derived from RGLVQ [6], the Relational GLVQ. This methods family does not require the Euclidean data and can work with any data for which dissimilarities can be computed.

- **Likelihood-Ratio Maximization** methods are based on the Gaussian mixture probabilistic model representing the data. The RSLVQ method was the first method that proposed this approach [110] and created a new branch in the LVQ development. RSLVQ compares the resulting probability density functions that were generated by the Gaussian mixture model of a correct class and the one produced from the an incorrect class [99].

2.4.2 Project's classifier selection reasoning

For the aims of this project, we selected an RGLVQ-based classifier adopted for Semi-Supervised Learning [2]. The selection was made as this particular area of the LVQ research – relational prototype classifier – is relatively active, developed and proven as stable and reliable for applying to the real-world classification tasks [2,6,99]. The SSL application was already described in [2]. The dissimilarity representation of the data allows applying the classifier developed in this project to different areas, not only environment sensor data, which makes its contributions to be available for a wider use [6].

2.4.3 Generalized Learning Vector Quantization

During the last 25 years since the LVQ research started its development, the most explored direction was the class of Margin Maximization methods. The originator of this branch of research, the GLVQ method is still used as the basis for most of the modern LVQ classifiers [6,99].
Sato and Yamada in [1] discovered a significant drawback of LVQ technique. They have proven that *LVQ2.1* described by Kohonen may lead to diverging of prototype vectors in a longer run. They proved it with the fact that while closest correct- and wrong-labeled prototypes are being updated, others are left unattended.
To overcome this issue, they proposed a new method called Generalized Learning Vector Quantization (GLVQ). The aim of the method is to ensure that prototype vectors position are being updated basing on minimizing the cost function.
Sato and Yamada introduce a new parameter, the relative distance difference:

$$\mu(x) = \frac{d_1 - d_2}{d_1 + d_2} \tag{2.3}$$

where d_1 and d_2 are distances to the point x from γ_1 and γ_2 respectively, two closest prototype vectors to x. We assume, γ_1 has the same class label as x, when γ_2 represents a label of a different class.
The values of μ fall into the range: $-1 < \mu < 1$. If data x was classified correctly, μ gets negative value. Otherwise, μ is positive. To minimize error $\mu(x)$ should decrease for all inputs. Hence, the cost function S that has to be minimized is:

$$S = \sum_{i=1}^{N} f(\mu(x_i)) \tag{2.4}$$

where N corresponds to the number of input vectors, $f(\mu)$ is a monotonically increasing function.
Updates for γ_1 and γ_2 are generated using the steepest descent method:

$$\delta\gamma_i = \gamma_i - \alpha \frac{\partial S}{\partial \gamma_i} \text{ for: } i = 1, 2 \tag{2.5}$$

where α is a small positive constant.

Sato and Yamada [1] propose to use the Euclidean distance $d_i = |x - \gamma_i|^2$ for obtaining the following learning rules:

$$\frac{\partial S}{\partial \gamma_1} = \frac{\partial S}{\partial \mu}\frac{\partial \mu}{\partial d_1}\frac{\partial d_1}{\partial \gamma_1} = -\frac{\partial f}{\partial \mu}\frac{4d^2}{(d_1 + d_2)^2}(x - \gamma_1) \tag{2.6}$$

$$\frac{\partial S}{\partial \gamma_2} = \frac{\partial S}{\partial \mu}\frac{\partial \mu}{\partial d_2}\frac{\partial d_2}{\partial \gamma_2} = -\frac{\partial f}{\partial \mu}\frac{4d^2}{(d_1 + d_2)^2}(x - \gamma_2) \tag{2.7}$$

The update rules for γ_1 and γ_2 can be derived as following:

$$\begin{aligned}\gamma_1 &\leftarrow \gamma_1 + \alpha \frac{\partial f}{\partial \mu} \frac{d_2}{(d_1+d_2)^2}(x-\gamma_1) \\ \gamma_2 &\leftarrow \gamma_2 + \alpha \frac{\partial f}{\partial \mu} \frac{d_1}{(d_1+d_2)^2}(x-\gamma_2)\end{aligned} \tag{2.8}$$

Prototypes positions are changed in relation to x, and the quantities of updating $\delta\gamma_1$ and $\delta\gamma_2$ depend on derivatives of μ. Therefore, the convergence property depends on the definition of μ [1].
$\frac{\partial f}{\partial \mu}$ is defined as a factor for updating, which values are always dependent from x. $\frac{\partial f}{\partial \mu}$ can bee seen as a weight for each x. The original GLVQ paper proposes to use $\frac{\partial f}{\partial \mu} = f(\mu,t)\{1-f(\mu,t)\}$, where t is a learning time and $f(\mu,t) = \frac{1}{1+e^{-\mu t}}$. This sigmoid function restricts input vectors to always stay around decision boundaries.
The work of Sato and Yamada has demonstrated efficiency of this approach which outperformed the original *LVQ2.1* technique [9]. However, some of the problems belonging to the original method were not solved [22]:

- the method demonstrated a poor performance on complex data with non-linear boundaries between classes
- the accuracy highly depends on the number of prototypes selected for classification and the model has a limited ability to determine a correct number of prototypes
- sensitive to initial prototype positions
- prototypes can permanently be stuck in local minima

The core findings of the research – the relative distance difference μ and prototype update rules – are used nowadays in various extensions of LVQ and GLVQ – RGLVQ [6], GRLVQ [100], GMLVQ [101] and others. In the next section we will provide a short overview of the work done in the field after Sato and Yamada introduced their method.

2.5 Dissimilarity data

Dissimilarity representation of data is described as an approach when data is transformed into values that correspond to the dissimilarity between objects with respect to all objects in the dataset [4]. Therefore, objects of the same class have smaller dissimilarities comparing to ones belonging to different classes.
This feature allows to apply statistical and machine learning-based approaches to data of any kind, preliminarily transformed into dissimilarity representation, as dissimilarities are used as discriminative features [4,5]

Different approaches of calculating dissimilarity representation were proposed [3,4,13], and we provide a short overview of them. It is worth to mention, in our project we are interested in applying distance-based metrics as data is numeric. In the *full representation approach*, a relatively small amount of objects from the dataset forms a representation set used for comparing all the other data to it. Each prototype forms a corresponding dimension that represents distances from all data points to a prototype. This approach has major disadvantages in the case if a dataset is small and some important prototypes can be missed [4]. Another approach is to use pairwise distances, where the dissimilarity representation is restricted by a matrix of pair-wise dissimilarities. This approach leads to a quadratic growth of space required for representation of the data. Therefore, it is not suitable in the case of a large dataset that needs to be processed as whole [15].
It is worth mentioning that in the modern Semi-Supervised learning, the majority of the methods are oriented to dealing with vectorial space data, represented by Euclidean distances between measurements [59]. However, in many cases, data can be represented in a form of dissimilarities between measurements – i.e, in bioinformatics, data mining, applied tasks of sociology. Proximity or dissimilarity data may not only come from distance-based measurements and does not need to always correspond to requirements of metric. It can be represented as Euclidean distance, Levenstein distance (known as the Word Edit Distance) or even in a form of empirical results retrieved from questionnaires or experiments [3].
Dissimilarity data representation in form of pairwise distances is an efficient way to transform dataset in the following cases:

- Data mostly consists of repeated measurements. In this case, pairwise dissimilarities for the majority of records in the dataset will be equal to 0. Therefore, sparse matrix data types can be used to efficiently handle such data
- Data used for processing is limited in size. To represent N records in the form of pairwise dissimilarities, we would need a matrix of size NxN. Therefore, it is crucial to keep amounts of data that is being sent for processing, as small as possible
- Data records are multi-dimensional. In this case, the transformation to dissimilarity representation flats the dimensions keeping only a measure of (dis-)similarity between objects

2.5.1 Overview of distance measures

Given $X = (x_1, ..., x_N) \in \mathbb{R}^{N \times d}$, $(x_i \in \mathbb{R}^d)$ and $Y = (y_1, ..., y_N) \in \mathbb{R}^N$ are sequences of measurements. The distance between two objects can be calculated with using different families of distance functions. The most widely used in Machine Learning [3] is the L_p group, or Minkowski family of distances

uniting Euclidean, Minkowski, Cityblock, Chebyshev. This type of dissimilarity measure is suitable for using with numerical data and satisfies properties of the metric.
Euclidean distance measure represents shortest distance between two points in L_2 or Euclidean space:

$$D_{EUC} = \left(\sum_{i=1}^{n} |x_i - y_i|^2) \right)^{\frac{1}{2}} \tag{2.9}$$

Minkowski distance:

$$D_{MK} = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{2.10}$$

Another family is the Inner Product Family [13] with such measures as Jaccard coefficient, Cosine similarity and others. This type of methods uses vector inner product as a base for determining dissimilarity between two objects represented by the vectors.
Jaccard coefficient:

$$D_{JAC} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} y_i^2 - \sum_{i=1}^{n} x_i y_i} \tag{2.11}$$

Cosine distance:

$$D_{COS} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}} \tag{2.12}$$

Choice of distance functions highly depends on the nature of data and characteristics of the model [13]. For example, the choice is depending on the fact either data is embeddable into Euclidean space or not. The edit distance is used when dealing with Nature Language Processing or DNA sequences.

2.6 Relational Learning Vector Quantization

2.6.1 Dissimilarity data in LVQ

Employing a dissimilarity representation in various clustering and classification methods was proposed by Pekalska and Duin in [14]. Researchers demonstrated that a dissimilarity representation can be more efficient for using in classification tasks comparing to the plain data. One of their next publication [15] introduced a density-based classifier, in [16] they also proposed Monte-Carlo simulation techniques for computing dissimilarities and classifying data.
Despite the fact that in the recent years data represented in the form of dissimilarity reached different areas of Machine Learning, the LVQ-based algorithms stayed aside the main trend as the core prototype-based techniques were mostly

known as such that require an input data to be embeddable into Euclidean vector space [1,2]. At the same time, proximity or dissimilarity data may not only come from distance-based measurements, and do not always correspond to requirements of metric [6].
In general, data can be represented as Euclidean distance, Levenstein distance (known as the Word edit distance) or even in a form of empirical results retrieved from questionnaires or experiments [3].
In [6] researchers extended Learning Vector Quantization Methods to use the relational data such as embedded into a pseudo-Euclidean space. This finding allows us to use (dis-)similarity representation of data for an extension of prototype-based method built on the top of LVQ.
To prove the hypothesis, researchers refer to the pairwise dissimilarity representation as $d_{ij} = d(v_i, v_j)$, where d_{ij} is a measure of dissimilarity between two objects v_i and v_j of a valid distance function. Pairwise dissimilarities together form data representation matrix D, with the following properties of its elements:

- $d_{ij} = d_{ji}$
- $d_{ii} = 0$

Pseudo-euclidean space is a real vector space equipped with a bilinear form $< x, y >_{m,n} = x^t I_{m,n} y$ where $I_{m,n}$ is a diagonal matrix consisting of m entries 1 and n entries of -1. The pair (m, n) is defined as the signature of the space. Value n shows how much the standard Euclidean norm should be corrected by negative eigenvalues in order to fit the given dissimilarity measure [6,7]. The dataset is referred to as Euclidean only if $n = 0$.
Such embedding can be computed for matrix D by means of eigenvalue decomposition of Gram matrix, computing vectors x^i where $d_{ij} = \langle x^i - x^j, x^i - x^j \rangle_{p,q}$ holds for every pair of points in data. Hence, it is possible to use LVQ methods in pseudo-Euclidean space as it relies only on vector operations.
Explicit transferring into pseudo-Euclidean space creates a problem that out-of-sample extensions of new data in form of dissimilarities cannot be computed immediately. Here and after, we will follow the assumption proposed in [7] that prototype positions can be restricted to linear combination of data points of the form:

$$\gamma^j = \sum_i \alpha_{ji} x^i, \text{ such that } \sum_i \alpha_{ji} = 1 \tag{2.13}$$

where γ^j is a j-th prototype for given D.

Researchers explain their assumption by the fact that prototypes are placed into representative points in the data space, therefore can be restricted to the affine subspace spanned by the given data points.
Hence, the distance between a prototype and a given point or a prototype and

another prototype can be computed basing only on the pairwise dissimilarity matrix:

$$d(x^i, \alpha^j) = [D \cdot \alpha_j]_i - \frac{1}{2} \cdot \alpha_j^t D \alpha_j \tag{2.14}$$

where $\alpha_j = (\alpha_{j1}, ..., \alpha_{jn})$ is a vector of coefficients describing the prototype implicitly. The details of the proof are described in [7]. This is the key observation that will be used in computing the cost function and update rules for the Proximity data Semi-Supervised Learning Vector Quantization model.

2.6.2 RGLVQ with dissimilarity data

As it is shown in the previous section, there is no such requirement to D to be embeddable into Euclidean space or satisfy metric conditions.
The method named RGLVQ proposed in [6] is a supervised clustering method which originated relational extensions of GLVQ. Its update rules and methods were taken as the basis for RPC-SSL [2], the method that we used in the project:
Given the data $x^i \in \mathbb{R}^N : i = 1, ..., m$, the prototypes are elements belonging to the same space $\gamma^j \in \mathbb{R}^N : j = 1, ..., k$ which decompose the data into receptive fields

$$R(\gamma^j) = \{x^i \ : \ \forall k \ \ d(x^i, \ \gamma^i) \leq d(x^i, \gamma^k)\}$$

where $d(x^i, \gamma^i) = ||x^i - \gamma^j||^2$ - squared Euclidean distance.
If labels for the data are known, each point in x^i has a corresponding class label $y(x^i) \in \{1, ..., K\}$ - the set of classes. Each prototype also has a fixed prototype class label $y(\gamma^j)$ from the same set of possible labels.
On the classification / clustering step, a data point from x is mapped to the closest prototype. The corresponding error of this action is calculated as $\sum_j \sum_{x^i \in R(\gamma^j)} \delta(y(x^i) \neq y(\gamma^j))$, where δ is the delta function for mapping which cannot be optimized explicitly due to vanishing gradients and discontinuities [6]. In the LVQ2 case the optimization takes place by applying pushing and pulling forces to the prototype positions, which is described by Kohohnen in [9] as Hebbian and unti-Hebbian updates of prototypes.
In the RGLVQ we use the cost function defined in GLVQ [1] and extending it as following:

$$E_{GLVQ} = \sum_i \Phi \left(\frac{d(x^i, \gamma^+(x^i)) - d(x^i, \gamma^-(x^i))}{d(x^i, \gamma^+(x^i)) + d(x^i, \gamma^-(x^i))} \right) \tag{2.15}$$

where Φ is an undifferentiable monotonic function; $\gamma^+(x^i)$ defines the closest prototype to the point x^i with the correct label - the same as the label of x^i, and $\gamma^-(x^i)$ is referred to as the closest prototype equipped with a different label.
According to the equation, a point from x contributes to the cost only if its distance to the closest correctly labeled prototype is smaller than to the closest

prototype that was labeled wrongly. From a general point of view, it pushes the correct prototype closer to data and the wrong prototype is pushed away. Expanding it to the terms of relational data (Eq. 2.14), the following cost function was obtained for so-called Relational Prototype Classifier [6]:

$$E_{RPC} = \sum_i \Phi \left(\frac{[D \cdot \gamma^+]_i - \frac{1}{2} \cdot (\gamma^+)^t D \gamma^+ - [D \cdot \gamma^-]_i + \frac{1}{2} \cdot (\gamma^-)^t D \gamma^-}{[D \cdot \gamma^+]_i - \frac{1}{2} \cdot (\gamma^+)^t D \gamma^+ + [D \cdot \gamma^-]_i - \frac{1}{2} \cdot (\gamma^-)^t D \gamma^-} \right) \quad (2.16)$$

where γ^+ is the closest correctly labeled prototype represented by the coefficients vector α^+ and γ^- is the closest prototype labeled differently represented by the coefficients vector α^-. D is the pairwise dissimilarity matrix.
A monotonic undifferentiable function is defined as $\Phi(x) = (1 + \exp(-x))^{-1}$.

Learning is performed by means of a stochastic gradient descent. Prototypes firstly are initialized randomly, representing the data point of x in a random order. Update rules for closest correctly and wrongly labeled prototypes therefore derived as:

$$\Delta\alpha_k^+ = -\Phi'(\mu(x^i)) \cdot \mu^+(x^i) \cdot \frac{\partial([D\alpha^+]_i - \frac{1}{2} \cdot (\alpha^+)^t D \alpha^+)}{\partial \alpha_k^+} \quad (2.17)$$

$$\Delta\alpha_k^- = -\Phi'(\mu(x^i)) \cdot \mu^-(x^i) \cdot \frac{\partial([D\alpha^-]_i - \frac{1}{2} \cdot (\alpha^-)^t D \alpha^-)}{\partial \alpha_k^-} \quad (2.18)$$

$$\mu(x^i) = \frac{d(x^i, \gamma^+) - d(x^i, \gamma^-)}{d(x^i, \gamma^+) + d(x^i, \gamma^-)} \quad (2.19)$$

$$\mu^+(x^i) = \frac{2 \cdot d(x^i, \gamma^-)}{(d(x^i, \gamma^+) + d(x^i, \gamma^-))^2} \quad (2.20)$$

$$\mu^-(x^i) = \frac{2 \cdot d(x^i, \gamma^+)}{(d(x^i, \gamma^+) + d(x^i, \gamma^-))^2} \quad (2.21)$$

The partial derivative can be expanded as:

$$\frac{\partial([D\alpha_j]_i - \frac{1}{2} \cdot \alpha_j^t D \alpha_j)}{\partial \alpha_{jk}} = d_{ik} - \sum_l d_{lk} \alpha_{jl} \quad (2.22)$$

After performing an adaptation step, the coefficients vector α is normalized to satisfy the condition $\sum_i \alpha_{ji} = 1$.
To compute clustering for previously unseen data point (or so-called out-of-sample extension of the dataset), we can use existing prototypes and map the point to the one having the smallest dissimilarity with this point:

$$d(x^i, \gamma_j) = D(x^i)^t \cdot \alpha_j - \frac{1}{2} \cdot \alpha_j^t \cdot D \alpha_j \quad (2.23)$$

where x is the out-of-sample point, γ_j – a prototype, and $D(x)$ – a pairwise dissimilarity matrix of the seen data extended with the unseen point of interest.

The extension mechanism will be described later in the "Algorithm" section.
For the RPC classifier which is also being the classifier of the RGLVQ method, the authors confirmed the overall performance on several known datasets (The Copenhagen Chromosomes data [28], Amazon47, Aural-Sonar, Face Recognition [29]) as comparable to the SVM method [6].

2.7 Semi-supervised prediction with RPC

The method was originally described in [2] and is based on estimating the prediction region containing possible cluster labels assignment for a data point and further seeks the best fit of the label to the data.
The training dataset is referred to as $T1$, where the pair of a data point and its corresponding label is $T1_i = (x^i, l_i) \in Z = V \times L$. The data point with an unknown label is denoted as x^{N+1}.
The method computes the conformal prediction for given training data $(T1_i)_i = 1, ..., N$ and the observed point x^{N+1} and results in finding for a chosen error rate ϵ the $(1-\epsilon)$-prediction region $\Gamma^\epsilon(T1_1, ..., T1_N, x^{N+1}) \subseteq L$ which contains label assignments for data points in Γ^ϵ ensuring the error rate not higher than ϵ. The prediction region Γ^ϵ is computed as follows:
Given the fixed nonconformity measure A, the significance level ϵ, data points $T1_1, ..., T1_N$, the observed data point x^{N+1} with unknown label, and a label l from the label set. The goal is to find whether $\Gamma^\epsilon(T1_1, ..., T1_N, x^{N+1})$ contains l or not:

Set:

$$T1_{N+1} = (x^{N+1}, l)$$

for each $i = (1, ..., N+1)$ compute:

$$\mu_i = A(\{T1_1, ..., T1_{N+1}\} \setminus \{T1_i\}, T1_i)$$

Set:

$$r_l = \frac{|\{i = 1, ..., N+1 \mid \mu_i \geq \mu_{N+1}\}|}{N+1}$$

Include l to Γ^ϵ only if $r_l > \epsilon$.

In the case of the Relational Prototype Classifier, the nonconformity measure A can be computed as follows:

$$\mu_i = \frac{d^+(T1_i)}{d^-(T1_i)} \tag{2.24}$$

where $d^+(T1_i)$ is the distance between $T1_i$ and the closest prototype labeled l and $d^-(T1_i)$ is the distance between $T1_i$ and the closest prototype labeled differently.

The heuristic approach to using $\Gamma^{\epsilon}(T1_1, ..., T1_N, x^{N+1})$ for predicting clusters of data is based on the underlying meaning of the nonconformity measure A. If the error rate value ϵ is close to 0, it means the conformal prediction can be done with no errors. Hence, the only possible case is when the prediction region $\Gamma^{\epsilon}(T1_1, ..., T1_N, x^{N+1})$ contains all the possible labels. When raising value of ϵ, the conformal prediction increases its information content and excludes labels that unlikely to fit to the data [2] – all that correspond to r_l being less or equal to ϵ. For them there will be only a few points $T1_i$ in the dataset $T1$ that can be as non-conformal as $T1_{N+1} = (x^{N+1}, l)$ which indicates that the point of interest $T1_{N+1}$ does not belong to the distribution of the region represented by label l.
The fitting process of this method is performed as trading between the error rate and the information content. The goal is to find nonconformities containing only one label which would mean a good degree of separation between clusters represented by labels.
Given an input label l_i, we compute the following error rates:

- ϵ_1^i – the r-value of the best fitting label.
- ϵ_2^i – the r-value of the second best fitting label.

For both error rates $|\Gamma^{\epsilon}(D, x^i)| = 1$ must be satisfied. In practice it means, we aim at finding the smallest possible ϵ_1^i and the largest possible ϵ_2^i to ensure the clear border between the best fitting class and others.
Researchers propose two convenient metrics based on the error rates to be employed in the model:

$$\text{credibility} : \epsilon_2^i = r_{y1st}$$

This measure estimates how good is the fit of the best label to data while not being an outlier.

$$\text{confidence} : 1 - \epsilon_1^i = 1 - r_{y2nd}$$

Confidence reflects how far the second best label and all the others are from the best fitting one.
After introducing these two metrics, it becomes possible to build a model for fitting the error rates and clustering the data.

2.8 Clustering quality estimation

After training the model and generating labels for unlabeled points, the question of verifying the quality of clustering has to be addressed. The main challenge is that the model does not have the ground truth information about the labels generated. The possible way of measuring how accurate was clustering is to use methods that apply the model itself to checking how good the new

clustering fits the data. We reviewed an existing publication on available indexes for estimating clustering quality.

Silhouette index [33] is referred to as $s(i)$ and calculated as follows:

$$s(i) = \frac{b(i) - a(i)}{max(b(i), a(i))}$$

where $a(i)$ corresponds to the average dissimilarity between the point i and all the other points belonging to the same cluster as i. $b(i)$ is the minimal average dissimilarity between the point i and other points belonging to other clusters than i. Silhouette index values fall in a range $[-1; +1]$, where a higher value represents a better cluster fit to data.

C-index [34] is defined as:

$$C_i = \left(\frac{d_w - min(d_w)}{max(d_w) - min(d_w)} \right)$$

where d_w represents the sum of all pairwise distances between points belonging to the same cluster, the number of such points is defined as n. $max(d_w)$ is the sum of n largest pairwise distances among all points in the dataset. $min(d_w)$ is the sum of n smallest pairwise distances.

CH (Calinski) index [35]:

$$CH(k) = \frac{B/(k-1)}{W/(n-k)}$$

where B defines the between cluster sum of squares, and W - the within-cluster sum of squares. k is the number of clusters, n - the number of points in the data.

Dunn's validity index [36] is the metrics that explains how compact and well-separated a set of clusters is:

$$D = min_{1 \le i \le k} \left(min_{i+1 \le j \le k} \left(\frac{dist(c_i, c_j)}{max_{1 \le l \le k} diam(c_l)} \right) \right)$$

where $dist(c_i, c_j)$ is the distance function between two clusters c_i and c_j defined as $dist(c_i, c_j) = min_{x_i \in c_i,\ x_j \in c_j} d(x_i, x_y)$
$diam(c_l)$ is referred to as: $max_{x_{l_1}, x_{l_2} \in c_l}(x_{l_1}, x_{l_2})$

Various studies of the efficiency of described indexes were conducted [37, 38, 39]. In [37], researchers pointed out that CH-index provides the best results in the case of existence of equal sized groups of data. According to [38], the Silhouette index achieves relatively good performance when data is noisy or contains outliers comparing to other measurements. Dunn's validity index was

eliminated due to high computation loads in case of applying to the stream, and in [37] C-index failed to identify noisy clusters showing not satisfiable performance.
According to the review results, the Silhouette index was selected as the main clustering quality criteria to be used with the model.

Chapter 3

Data origins and processing

3.1 CIVIS project and the Green Campus Initiative

Originally, the sensor data setup and the data collection back-end were developed for the CIVIS [18] (Cities as Drivers of Social change) project involving 10 universities:

- Associazione Trento RISE
- Aalto university, Finland
- Imperial College London, UK
- ENEL Foundation, Italy
- Instituto Superior Tecnico, Portugal
- Karlsruhe Institute of Technology, Germany
- Kungliga Tekniska Hogskolan, Sweden
- Santer Reply SpA Italy
- Nederlandse Organisatie voor toegepast Natuurwetenschappelijkonderzoek, Netherlands
- Delft University of Technology, Netherlands

The goal of the project is to get the fusion of social aspects united with technology innovations. In this case, the drivers of innovations are smart grids [26] – electricity grids designed to provide the utility companies with full visibility of energy flows and consumption on each stage, from the producer to the end user. The grids have to become a part of social and cultural layers and need to be viewed as complex socio-techno-economics system with corresponding decision-making layers [18].
One of the CIVIS project objectives is to get an understanding of energy consumption patterns and efficiency for small city units, such as campuses, technoparks, production areas [18].
The Otaniemi campus was selected as one of the units. Under the supervision of the "Technical Research Centre of Finland" (VTT) the project "Green Campus" aims at increasing an energy efficiency of campus buildings through

building the innovative energy management and control system. It is expected to decrease the overall consumption by 15 % by 2030. To achieve this goal, VTT collaborated with K-MEG (South Korea), Korea Electronics Technology Institute and Samsung Corporation in a joint effort for providing advanced monitoring of the energy consumption [19]. In 2013, among the others 160 intelligent sensors were installed in within one of the Aalto University facilities located at Otakaari 4, Espoo [17] (see Fig. 3.1, Fig. 3.2).

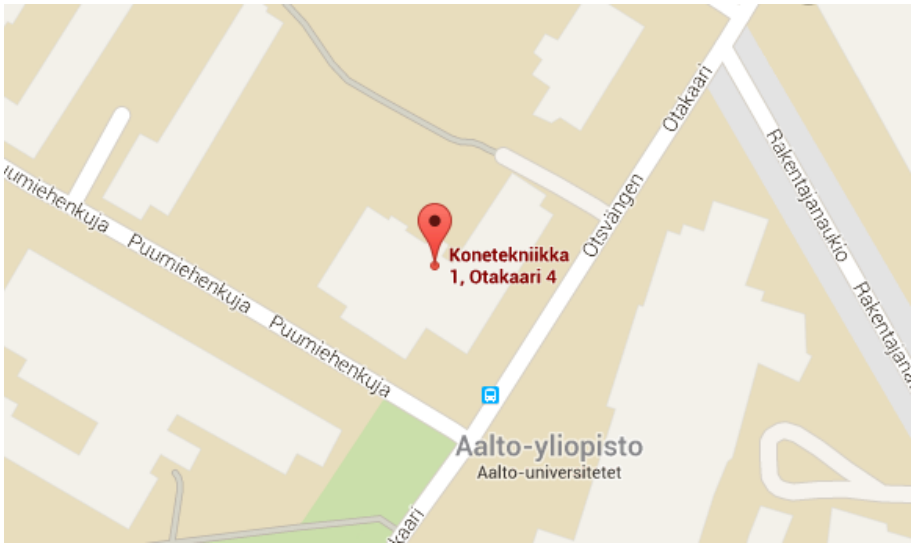

Figure 3.1: The building of Otakaari, 4 on the map taken from the Google Maps service

Figure 3.2: The building of Otakaari 4, the front view taken by Google Maps service

3.2 Data retrieving and processing

The rooms of the building have different patterns of energy consumption and environment conditions as they are being used for teaching (lecture halls, rooms and computer laboratories), research, administrative tasks, storage.
Sensors were constructed and shipped by the Korean company "MAXFOR". Installed sensors are located at the same distance from the floor (1.0-1.25 m), and cover about 10-15 m^2 of room area (Fig. 3.3). Each of the sensors communicates to the main hub through the Wi-Fi, rooms use shared hubs.

Figure 3.3: The pair of sensors pinned to the wall submits environment parameters to API. Photo taken from [18]

4 parameters are being recorded by sensors:

- temperature (Celsius degrees)
- relative humidity (percentage)

- CO2 concentration (percentage)
- lightness (lux)

Data records are transmitted to hubs once a change in parameter is detected, and then sent to the remote Data Processing System managed by KETI. Records can be retrieved through the API in a format of comma-separated values from each of 160 sensors.

Chapter 4

Algorithms and methods

In this chapter, we describe all methods and techniques that we used to solve the semi-supervised anomaly detection on time-series data. In the previous part, the most important concepts and approaches used for building the final algorithm were reviewed.

4.1 Approach overview

The general approach to solving the problem is built around the semi-supervised model that receives a labeled dataset from the Expert. The model combines it with unlabeled data from sensors and produces a classification label for a data record.

- Each parameter is being treated separately. In a chain of consecutive periodic API calls the **data is pulled from the API**. Each data record represents 3 independent parameters (Temperature, CO_2, humidity), all records accumulated into 3 corresponding vectors of the length L.
- Accumulated trinity vectors V_{acc} are separately transformed with a **rolling window technique** [31] and normalized.
- The **dissimilarity representation** is found with respect to all the data used in model training.
- We assume, priorly to start of training, the **labeled dataset T1 is provided** by a qualified analyst who sets labels for important data points (markers) which represent both anomalies and points belonging to the trivial flow.
- The training data with known labels (denoted as T1) and a new data (T2) is being fed to the model for each parameter separately, where the clustering is taking place.
- Output label for each point in T2 is produced. The new data and labels are appended to the $T2_{daily}$ dataset.
- Points which labels correspond to anomalies are detected. For each parameter, outliers are processed separately, the majority vote principle based on the anomaly severity is applied to determine whether the assembled 3-parameters data point is considered as an outlier.

- The nightly iteration of the SSL-RPC training utilizes labeled data $T1$ and unlabeled dataset $T2_{daily}$ containing all data points collected during last 24 hours.
- Points tagged as anomalies by the night run are reported to the Expert who reviews them. If the Expert corrects a label for any of such points, it is added to T1 as a labeled point.

While the main data stream is being constantly processed, every night the model generates the assembled prediction outliers on the full set of the daily data. This is done in order to make the model evolving under changing circumstances basing on the labels and decisions made by the Expert.

The generalized architecture of the approach is shown in Figure 4.1.

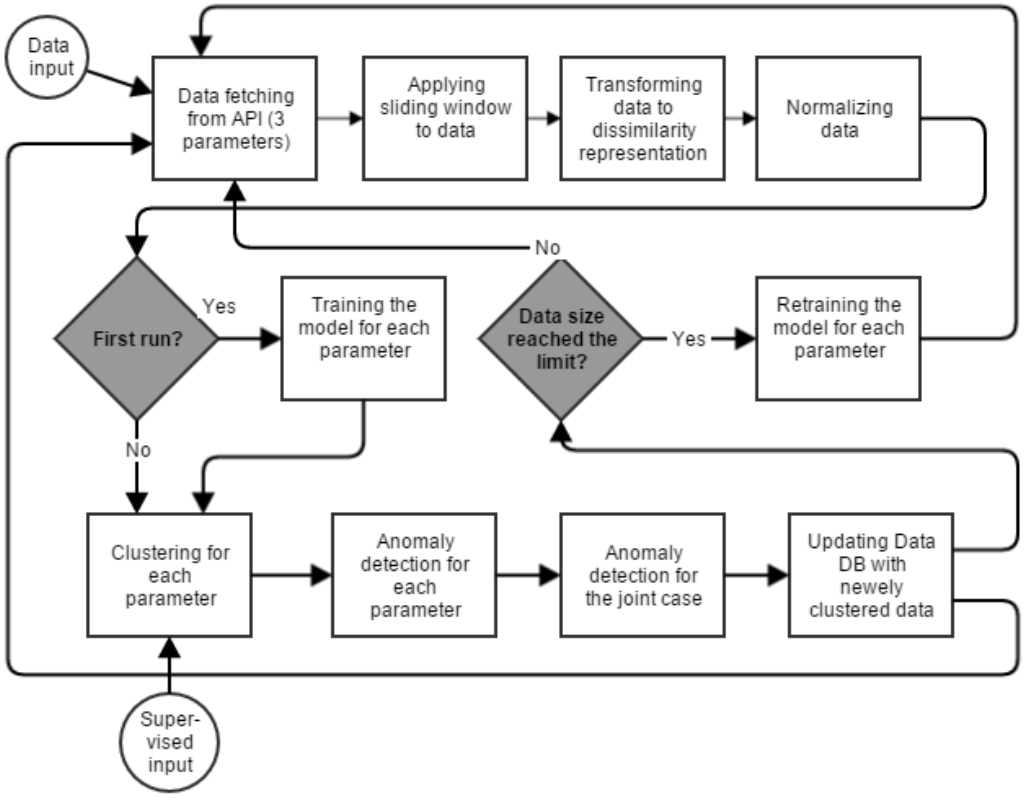

Figure 4.1: The clustering workflow at the full scale – from getting the data to updating the model

The implementation. For implementing the described approach, Python programming language was selected. The selection was made basing on its functionality and features making this language one of the standard choices for data processing and analysis tasks. Also, it has a broad selection of libraries suitable for Machine Learning. The valid training data was stored in a Mongo

DB instance. For retrieving the records from the API we used **json** library of Python. Mathematical operations were supported by **NumPy** and **SciPy** [30]. The DB access is organized with using of **SQLAlchemy**.

4.2 Data preprocessing

Data retrieving is being done in small batches occurring regularly within the predefined interval of time. In the project scope, we are mainly interested in 3 main environmental parameters – temperature, CO_2 concentration and humidity which are queried by API via a JSON request, i.e.:
http://121.78.237.162:8000/otakaari4?start=2014/08/25-01:00:00&end=2014/08/29-01:00:00&number=4201&type=co2&token=aaltootakaari4

The following request gets all CO2 measurements for room 4201 between 01:00 AM 25 Aug 2014 and 01:00 AM 29 Aug 2014.
Output example of the executed API command:

vtt_otaakari4_co2_4201 1408943161 3555 host=keti3 nodeid=4201 vtt_otaakari4_-co2_4201 1408943445 3555 host=keti3 nodeid=4201 ...

where "3555" is current CO2 value, and 10-digits number (i.e. 1408943161) is a timestamp of the measurement recorded by a sensor
We parse each line to extract the information from data:

- parameter name (CO_2 concentration, temperature or humidity)
- room number
- value

The API was designed by KETI specialists to generate a data point only in the case when a parameter value was changed.
Regular sampling is required as the data preparation step in order to normalize points distribution over the timeline. We transform the data to achieve a stable frequency Θ. In our case $\Theta = 30$ *sec* is the interval between neighbor data points. Therefore, one minute of observation corresponds to 3 data records – for 0 second, 30-th second and 60-th second. If there was no change, the previous value gets replicated to the next point which is 30 seconds further in time. In this setting, we transform time-series to the regularly sampled form as we aim at reducing amounts of information that have to be stored alongside with the main observation data. Another positive outcome of applying regular sampling to the data is avoiding values that sensor can produce when a measured parameter is placed on the edge of two consecutive minimal observation values which causes API to start reporting a parameter change on each transition between these values. This situation may lead to generating hundreds

of almost equal measurements registered by a sensor and affecting the model learning set.
The selection $\Theta = 30\ sec$ as the sampling frequency parameter was done naively with respect to the expected use cases of the system which do not require more frequent updates of parameter values. Also, this very basic technique allows to make the data trend smoother and avoiding raising an alarm on instant outliers that are rarely spotted on the trend in the long run.

Forming input data vectors. The approach to utilizing time-series data is based on the aims and goals of the thesis. As it was previously stated, the data classification in the framework occurs in two stages. The first stage, "fast classification" is done as soon as possible in order to verify whether or not is the recently arrived label an outlier. The second stage, "nightly classification" operates with all the data gathered within 24 hours.
Our approach to the data representation task is based on the idea of decomposing a time-dependent stream data of an infinite length into small fractions of the maximal length l_{max} which are treated as a single l_{max}-dimensional data record. It allows us to use the proposed SSL-RPC classifier operating with clusters of independent time-series of maximal length l_{max}.
Depending on the records sampling frequency Θ, the data vector L contains as many data points as it is required by the limit response time T parameter. If $T = 0$, the model must be working in a real-time, however, it is not the exact use case for the project's model. Minimal model-compatible value $T = \Theta * S_r$, where S_r is the size of the rolling window (see the next point).

Rolling window technique [31] is applied to data points collected in L before submitting it to the model input as T2. This is a basic technique in time-series analysis that allows to make peaks smoother and, therefore, avoid false positive outlier detection. Using the rolling window technique, the time-series information about preceding and succeeding data points is recorded in order to form an independent time-series mentioned in the previous item.
Let $\{y_i\}$ with $i = 1, ..., N$ be a sampled time-series data of the length N. Then W is an integer defining a size of the window such that $0 < W \leq N$ and $\forall k$ indexes of data points in $\{y\}$ where $W \leq k \leq N$. Then a new corresponding data point after applying window is $y_k^{'} = \{y_{k-W}, y_{k-W+1}, ..., y_k\}$.
The window size $W = 3$ was selected. The decision was made according to requirements of space, performance and data visualization possibility.

Normalization is taking a place at the next step of the input data preprocessing and aims at meeting the requirements for the input data that is fed to the Stochastic Gradient Descent outlined by LeCun et al. in [32]. Researchers suggest to remove an average of each of the input variables and normalize their variances. Therefore, for each parameter we subtracted the parameter mean μ

and divided the result by the standard deviation σ:

$$x^{'} = \frac{x - \mu}{\sigma}$$

The resulting data is known as such that grants better gradient descent performance and convergence rates [32].
The resulting data is transformed to the form of the **pairwise dissimilarity** using one of existing valid distance functions – in our case, Euclidean distance [8]. The Euclidean measure was selected as one recommended by RPC method authors [2,6]. However, they mention that non-Euclidean distance functions can also be applicable to the classifier which opens a space for performing a research which is being out of the scope of this project.

4.3 Model and training

The model for clustering data contains two main modules – the RPC classifier [2] and the Conformal Prediction algorithm [6].

RPC classfier is based on the approach described in the 2. It operates with the dataset denoted as $T1$ – the labeled dataset which serves as the training base for the RPC. The algorithm updates the prototypes positions in order to fit them to the current data and achieve the most accurate representation in terms of the cost function minimization.
RPC optimization is done using the Stochastic Gradient Descent method. In our implementation the original RPC update rules were transformed to the following form:

$$\Delta\gamma_k^+ = -\Phi^{'}(\mu(v_i)) \cdot \mu^+(v_i) \cdot \frac{\partial([D\gamma^+]_i - \frac{1}{2} \cdot (\gamma^+)^t D\gamma^+)}{\partial\gamma_k^+} \cdot \frac{1}{\lambda} \tag{4.1}$$

$$\Delta\gamma_k^- = -\Phi^{'}(\mu(v_i)) \cdot \mu^-(v_i) \cdot \frac{\partial([D\gamma^-]_i - \frac{1}{2} \cdot (\gamma^-)^t D\gamma^-)}{\partial\gamma_k^-} \cdot \frac{1}{\lambda} \tag{4.2}$$

where $0 < \frac{1}{\lambda} < 1$ is a regularization coefficient introduced in order to achieve a better convergence of the update steps. As we found out when performing visualized experiments, the original update rules lead to gradient descent steps which miss the global minimum areas and never converge. We applied a basic regularization technique to the gradient descent steps and reported a change of model performance in the next section. The exact values of λ might depend on the data nature and need to be investigated. Further, we perform empirical research on the model behavior under different λ values in order to find optimal limits for solving the project's task.

Semi-supervised Learning using RPC: Conformal prediction
In the chapter 2.8 we sketched the Semi-supervised extension of RPC. It is

based on forming a low-credibility region Γ and examining points from the unlabeled dataset $T2$ through *credibility* and $confidence$ parameters to find out whether the point belongs to Γ or not. The practical application of the method includes some additional comments and assumptions.
The key feature of the SSL-RPC approach is the automatic adaptation of the model complexity and the number of prototypes used for representing the data. Initially, we use labeled set T1 for the preliminary model training. T2 is used for verifying the quality of clustering after each iteration of the full RPC classifer update. Firstly, we compute *credibility* and $confidence$ for each data point in T1 to form the set $\boldsymbol{\beta}$ corresponding to the Γ region from the chapter 2.8. To recall, the low confidence for the point from $T2$ is defined as $(1-\epsilon_1^i) \leq (1-\frac{1}{L})$, low credibility: $\epsilon_2^i \leq \frac{1}{L}$, where L is the length of the dataset referred to as $T1$. In [2] researchers define the universal rule for adding points to $\boldsymbol{\beta}$:

$$\boldsymbol{\beta} = \Big\{ v_i \in T2 \ : \ (1-\epsilon_1^i) \leq (1-\frac{1}{L}) \ \vee \ \epsilon_2^i \leq \frac{1}{L} \Big\} \tag{4.3}$$

The size of the $\boldsymbol{\beta}$ region also works as stopping criterion for the prototype fitting algorithm. The researchers emphasize that the size is not supposed to be equal to 0 in order to confirm a good fit of the clustering. Vice versa, empty $\boldsymbol{\beta}$ indicates too dense clustering [2]. In the experiments on the simulated data it was found that optimal size of $\boldsymbol{\beta}$ is $\theta = 5$ points, therefore the convergence criterion for the algorithm we use is $\theta : |\boldsymbol{\beta}| \leq 5$

4.3.1 Classification algorithm: RPC + SSL

The Semi-Supervised RPC training algorithm can be represented in the pseudocode (see Alg. 1).

Algorithm 1 The model main algorithm: Semi-supervised RPC

1: $Beta := \emptyset$
2: **Initialize prototypes** W
3: $W \leftarrow$ **TrainRPC** on $T1$ from current W
4: $Wbest \leftarrow W$
5: $NewLabels \leftarrow$ **Conformal prediction** $T2$ with prototypes W
6: $NewLabelsBest \leftarrow NewLabels$
7: $Beta \leftarrow$ **points** from the region of uncertainty in T2
8: **while** $|Beta| \geq \theta$ **or** *max-iter-count* ≤ 50 **do**
9: **for** each **cluster** within $Beta$ **do**
10: $Beta\_unc \leftarrow$ **generate new prototype vector**(s) from Beta
11: $W \leftarrow W \cup Beta\_unc$
12: **end for**
13: $W \leftarrow$ **TrainRPC** on $T1$ from current W
14: $NewLabels \leftarrow$ **Conformal prediction** $T2$ with prototypes W
15: $S\_index \leftarrow calculate$ **Silhouette index** $for\ NewLabels$
16: **if** $S\_index \geq S\_index\_best$ **then**
17: $NewLabelsBest \leftarrow NewLabels$
18: $Wbest \leftarrow W$
19: $S\_index\_best \leftarrow S\_index$
20: **end if**
21: **end while**
22: **Return** $Wbest,\ NewLabelsBest$

This algorithm unites the principles and approaches explained in the previous chapters. After the first run of the RPC classifier and Conformal prediction (steps 1-6), the model forms the first set of labels denoted as $NewLabels$ for T2 and the first iteration of the prototypes W, which initial values were updated in order to achieve a better data fit.

From this point, the model starts fitting prototypes to the data using the set $\boldsymbol{\beta}$ consisting of T2 points with low confidence/credibility (step 6). In this loop, the model is trying to create new prototypes from $\boldsymbol{\beta}$ fitting to low-confidence data. It leads to more accurate classification results for unlabeled set T2. The data fit is measured by the Silhouette index [33] – the prototype configuration fitting both T1 and T2 the best is stored.

As the loop exit criterion, the size of the region $\boldsymbol{\beta}$ is used. When it falls below the threshold Θ, the model reached its training limit with current data, and there is no feasibility in creating new prototypes from $\boldsymbol{\beta}$. Another possible exit criterion is the fixed number of iterations. We use *max-iter-count* ≤ 50 which can be tuned according to the model needs and time restrictions.

It has to be kept in mind that each iteration performs Conformal Prediction with creating new prototypes from the low-confidence data. Therefore, if after N iterations the size of $\boldsymbol{\beta}$ is not decreasing it might be an indicator of inseparable clusters. Also, each new iteration of the loop increases computational

load as more and more prototypes are spawned to the data space.

4.3.2 RPC algorithm in details

The general RPC training procedure is very similar to one described in the literature [2,6] and in the previous chapters. Its pseudocode used in the thesis implementation is referred to as Algorithm 2:

Algorithm 2 Relational prototype classification step (RPC) in details

```
1: Given D, W                          ▷ D – T1 pairwise distances matrix
                                       W – initial prototype
2: maxiter ← 50
3: while iter < maxiter do
4:     for each row in D do            ▷ each row represents all dissimilarities of one point in T1
5:         γ⁺ ← the closest prototype with the same label
6:         γ⁻ ← the closest prototype with a different label
7:         Find ∇(γ⁺)
8:         Find ∇(γ⁻)
9:         W(γ⁺) ← W(γ⁺) + ∇(γ⁺) · λ    ▷ Update corresponding to γ⁺ prototype vector
10:        W(γ⁻) ← W(γ⁻) + ∇(γ⁻) · λ    ▷ Update corresponding to γ⁻ prototype vector
11:    end for
12:    Calculate the Cost function E
13:    if E < E_best then
14:        E_best ← E
15:        W_best ← W
16:    else
17:        if E − E_best > ρ then      ▷ ρ is a maximal allowed convergence threshold
18:            W ← W_best
19:            break
20:        end if
21:    end if
22: end while
23: Return W
```

The algorithm implements the RPC classifier described in previous chapters, and practically performs a limited number of full stochastic gradient descent iterations, updating prototypes in W in order to achieve the best possible fit to the data T1 represented in the form of dissimilarities. The convergence is reported when the cost function E starts to diverge and does not improve the overall result fit anymore.

In [2], the RPC convergence criterion is proposed as $E - E_{best} > 0$, which means, the algorithm breaks at the first iteration when the cost function starts growing. However, according to [1,6,7], one of the LVQ algorithm drawbacks is the fact that it can possibly get stuck in the local minima after updating the prototypes. We assumed if we use some small positive threshold ρ instead of **zero**, the model can continue descent after reaching the local minima. Therefore, we set the stopping criterion as $E - E_{best} > \rho$, where $0 < \rho < 1$.

4.3.3 New prototype creation algorithm

On the each iteration after forming the set $\boldsymbol{\beta}$, we are interested in improving the clustering fit by creating a new prototype fitting $\boldsymbol{\beta}$ and decreasing its size. The researchers in [2] do not provide sufficient details on how to generate a new prototype. Therefore, this procedure was developed and tested within out project (steps 9-12 of the Algorithm 1).
First of all, we know from [2] that the new prototype "is set to the representative data point (median) in $\boldsymbol{\beta}$". It is labeled with the same label as the nearest neighbor from $T1$.
The general procedure of getting new prototypes out from $\boldsymbol{\beta}$ and adding them to the main prototype set γ used for training and fitting the model is described in the pseudocode Algorithm 3:

Algorithm 3 New prototypes handling procedure

1: $\boldsymbol{\beta} \leftarrow$ T2 points of low confidence / credibility
2: **Find NN-clusters within $\boldsymbol{\beta}$**
3: **for** each cluster $\boldsymbol{\beta i}$ in $\boldsymbol{\beta}$ **do**
4: **for** each coordinate of data points in $\boldsymbol{\beta i}$ **do**
5: Find the median point position
6: **end for**
7: Form S_r coordinate-wise binary prototype vectors $\gamma_{\boldsymbol{\beta}_i^n}$
8: **for** each prototype vector in W **do**
9: Extend W with $|\gamma^n_{\boldsymbol{\beta_i}}|$-zeros: $W \leftarrow W \cup [0, .., 0]$
10: **end for**
11: **for** each prototype vector $\gamma^n_{\boldsymbol{\beta_i}}$ **do**
12: Assign the label of the $\boldsymbol{\beta}_{\boldsymbol{i}_{\boldsymbol{median}}}$'s NN from T1
13: Extend $\gamma^n_{\boldsymbol{\beta_i}}$ with $|T1|$-zeros: $\gamma^n_{\boldsymbol{\beta_i}} \leftarrow [0, .., 0] \cup \gamma^n_{\boldsymbol{\beta_i}}$
14: Append $\gamma^n_{\boldsymbol{\beta_i}}$ to the main prototype set W : $W \leftarrow W \cup \gamma^n_{\boldsymbol{\beta_i}}$
15: **end for**
16: Append points from $\boldsymbol{\beta_i}$ to T1
17: **end for**

1. The preliminary step is to determine how many prototypes does $\boldsymbol{\beta}$ contain (step 2 of Alg. 3). We aim at minimizing the number of prototypes to avoid overfitting the model, so the least number possible is equal to the number of unique clusters within $\boldsymbol{\beta}$. In order to solve the clustering

task and minimize the computational complexity we can use existing prototypes positions. The algorithm finds the nearest neighbor of each point $\boldsymbol{\beta}$ in the set formed of points that represent the corresponding prototype cluster in space. Basing on the definition [2] of W, we can calculate the position of i-th prototype as a dot product of each of prototypes and the data in T1:

$$pos_i = W_i \cdot T1$$

The vector P_{pos} of prototype positions is formed and labels of corresponding prototypes are assigned to their positions in P_{pos}.
Finally, the nearest neighbors for each point in $\boldsymbol{\beta}$ are found from P_{pos} and the corresponding label is assigned to the point in $\boldsymbol{\beta}$. The final number of clusters in $\boldsymbol{\beta}$ denoted as i, is equal to the number of unique labels of $\boldsymbol{\beta}$-points. Therefore, each of the clusters is denoted as $\boldsymbol{\beta}_i$.
All the following steps (2-6) are performed for each of $\boldsymbol{\beta}_i$ clusters separately.

2. Find the median point in $\boldsymbol{\beta}_i$.
   Assume, we have a list of data records denoted as $\boldsymbol{\beta}_i$. Finding the median in the case when each single point in $\boldsymbol{\beta}_i$ is having a data record of length *one* is an ordinary task. Nevertheless, in our case, each record is represented as a $1 \times S_r$ data vector, where S_r corresponds to the length of rolling window. The question is how to find a median for $1 \times S_r$-dimensional data record, and then use this information for creating a prototype having dimensions $1 \times N$, where N stands for the length of $\boldsymbol{\beta}_i$. The possible solution is to create a new prototype for all of S_r coordinates of all $\boldsymbol{\beta}$ points, assigning it to a corresponding point from the vector containing S_r-th coordinate's records of each of N points from $\boldsymbol{\beta}_i$:

$$[\boldsymbol{\beta}_i^1 = \boldsymbol{\beta}_{i_1}^1, \boldsymbol{\beta}_{i_2}^1, \boldsymbol{\beta}_{i_3}^1, .., \boldsymbol{\beta}_{i_N}^1]$$

$$[\boldsymbol{\beta}_i^1 = \boldsymbol{\beta}_{i_1}^2, \boldsymbol{\beta}_{i_2}^2, \boldsymbol{\beta}_{i_3}^2, .., \boldsymbol{\beta}_{i_N}^2]$$

$$...$$

$$[\boldsymbol{\beta}_i^{S_r} = \boldsymbol{\beta}_{i_1}^{S_r}, \boldsymbol{\beta}_{i_2}^{S_r}, \boldsymbol{\beta}_{i_3}^{S_r}, .., \boldsymbol{\beta}_{i_N}^{S_r}]$$

   Now the task is to find a median point in each of S_r-$\boldsymbol{\beta}_i^{S_r}$ vectors. According to the median definition, we sort all numbers in ascending orders and store the index of the one in the middle, if the number of elements is even. Otherwise, we examine two elements placed in the center and check which of them is closer to the real median of $\boldsymbol{\beta}_i$ computed by the NumPy.median function.

3. Create new prototypes $\gamma_{\boldsymbol{\beta}_i}$ representing the data points from $\boldsymbol{\beta}_i$
   At this step we need to find a prototype fitting to points from $\boldsymbol{\beta}_i$. Previously, we found the indexes of median data points in each of $\boldsymbol{\beta}_i^{S_r}$ coordinate vectors, where S_r is the length of the rolling window. Therefore,

we can create S_r prototype vectors each of them can be represented as a vector of the length $(|\boldsymbol{\beta}_i^{S_r}| - 1)$-**zeros** and the digit ***1*** placed in the position of the index corresponding to the index of the median point in $\boldsymbol{\beta}_i^{S_r}$ for each S_r selected for representing it, i.e.: $\gamma_{\boldsymbol{\beta}_i^{S_r}} = [0, 0, .., 0, 1, 0, .., 0, 0]$. This is done in order to assign the prototype to the median point and make sure that the requirement to a prototype $\sum_k \gamma_{\boldsymbol{\beta}_{i_k}^{S_r}} = 1$, where $\gamma_{\boldsymbol{\beta}_{i_k}^{S_r}}$ is the k-th element of the prototype vector $\gamma_{\beta_i^n}$, is satisfied.
As the outcome of this step, we form the set of S_r binary vectors $\gamma_{\boldsymbol{\beta}_i^{S_r}}$, and the data record for the median point which contains median points of each of S_r dimensions of data records in $\boldsymbol{\beta}_i$:

$$\boldsymbol{\beta}_{i_{median}} = (\boldsymbol{\beta}_{i_{median}}^1, \boldsymbol{\beta}_{i_{median}}^2, .., \boldsymbol{\beta}_{i_{median}}^{S_r})$$

4. Assign the label to S_r new prototypes $\gamma_{\beta_i^{S_r}}$ and points in $\boldsymbol{\beta}_i$.
   This label corresponds to the $\boldsymbol{\beta}_{i_{median}}$ nearest neighbor point's label from $T1$, denoted as $\boldsymbol{l}_{\boldsymbol{\gamma}_{\boldsymbol{\beta}_i}}$. All points in current $\boldsymbol{\beta}_i$ will receive this value.
   All newly created prototypes $\gamma_{\boldsymbol{\beta}_i^{S_r}}$ also receive the same label $\boldsymbol{l}_{\boldsymbol{\gamma}_{\boldsymbol{\beta}_i}}$

5. Extend existing prototype vectors W
   Assume, we recently added N points to T1 and appended new prototype vectors $\gamma_{\boldsymbol{\beta}_i^{S_r}}$. However, any of previously existing k prototype vectors denoted as W^k has dimensions $1 \times |T1|$, while the new $\gamma_{\boldsymbol{\beta}_i^{S_r}}$ are as large as $1 \times |T1| + |\boldsymbol{\beta}_i|$.
   Since existing prototypes W are the linear combination of the all T1 points, we need to extend each of old prototype vectors W^i by $N = |\boldsymbol{\beta}_i|$ records. To satisfy the requirement $\sum_k w_k^i = 1$ and not break current prototype-data fit, we extend each W^i vector with N zeros appended to the end of the vector.

6. Add the new prototype formed from $\boldsymbol{\beta}_i$ (step 2) to W.
   Each of S_r new prototypes $\gamma_{\boldsymbol{\beta}_i^{S_r}}$ needs to be appended to W. The horizontal dimension of W is equal to $|T1| + |\boldsymbol{\beta}_i|$, while the length of $\gamma_{\boldsymbol{\beta}_i^{S_r}} = |\boldsymbol{\beta}_i|$.
   To make appending possible, we need to extend $\gamma_{\boldsymbol{\beta}_i^{S_r}}$ with $|T1|$-zeros from the left, and after that to append the new prototype to W.

7. Append points from $\boldsymbol{\beta}_i$ to T1.
   This step is required since at this stage the model already determined the labels for the points and now is aimed at finding the best prototype representation of T1 $\vee$ T2. In the next iteration some of the points added in this step may again appear in $\boldsymbol{\beta}_i$ and potentially get different label. We assume N points are added to the end of the data vector T1.
   Their labels are added to the vector Y containing label assignments for T1 data points. It is important to note that points from T2 are labeled and added to T1 temporarily, within the current iteration scope.

For example, the prototype matrix W after performing the step 7 might have the following look:

$$W = \begin{bmatrix} 0.1 & 0.2 & 0.5 & 0.2 & 0 & 0 \\ 0.7 & 0.1 & 0.1 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

where W^3, W^4 are the newly added prototype vectors, the length $|\beta| = 2$, and as the median point corresponds to (β_1^1, β_2^2).

4.3.4 Initialization of prototypes

Initialization of prototypes is proposed to perform as a random procedure [1,2,6]. However, we noted that the developed SSL algorithm for RPC classifier is relying on initial assumptions about prototypes. On the steps 5 and 14 we perform so-called **Conformal prediction**, which forms the region β of points with uncertain assignments. In the case, the initial labels are wrong (i.e. randomly initialized prototypes were placed quite close in space with different labels assigned) the algorithm's performance might be affected by fitting the initially wrong prototype positions to data.
To improve the initial prototype knowledge, we decided to experiment with a semi-random or warm-start initialization. We create N prototypes as vectors of the length $l = |T1|$ composed of $(l-1)$-zeros and the digit **1** in the position of the corresponding point.
The warm-start still allows the algorithm to correct position of prototypes in further iterations relying on more accurate initial labeling.

Number of prototypes on the initialization
Since we experiment with the warm-start prototype initialization, a number of prototypes on the start might affect the accuracy of the model. It is a relatively cheap procedure, as in the each next iteration we will generate at least as many as S_r-prototypes out of $\boldsymbol{\beta}$. Therefore, it seems more beneficial to provide prototypes set better fitting the data in T1 from the initialization step.
The prototype generation techniques can depend on:

- data distribution – create more prototypes for dense areas in dataset
- initial number of clusters – given that we know the initial number of classes and relative distribution points over them. In this case, it is possible to create more prototypes for larger clusters in order to represent them more precisely
- employ information extracted from previous runs – as we always keep the recent history of measurements and assigned labels, it is also possible to automatically tune number of prototypes basing on the data of past executions

4.4 Algorithm computational complexity

The complexity of the Semi-Supervised Relational Prototype Classification is proven [2] to scale quadratically in complexity $O(N^2)$ in the number of training examples and linearly $O(N)$ in sizes of prototypes.
At the same time kernel approaches such as RBF scale as $O(N^3)$, as well as non-enhanced S^3VM – Semi-Supervised Support Vector Machines [40]. Which makes the SSL-RPC a reasonably good alternative in terms of running times and efficiency of computations.

4.5 Anomaly detection

Anomaly detection is based on the output generated by the SSL-RPC model. This procedure occurs in two stages – firstly the model detects outliers on each data vector L received for the input. The size of L is relatively small, predictions generated are local, light-weight in terms of computational power. However, they do not correspond to the whole picture. The second iteration happens every night when the daily data is analyzed, outliers are reported to the analyst who applies corrections to T1 if it is necessary.
The anomaly detection model has to be able to recognize 5 general clusters of data and produce numeric labels for parameters $l_{temp},\ l_{CO_2},\ l_{hum} \in (1, 2, 3, 4, 5)$. We propose the following classification:

1. Significant anomaly under the expected norm. This cluster represents the anomalies carrying a strong evidence of a deviation from the main data flow. In the target system perspective, this category represents the values that are definitely being anomalous and must be reported straight after finding them.

2. Medium anomaly under the expected norm. The cluster incorporates values that lay under the expected norm, however, still can be classified as non-harmful to neither a human health or the room's equipment. This is the transitional class which might show either a subnormal behavior of the system or be just a deviation from the norm that does not need to be reported as an anomaly.

3. Expected norm. Represents the widest cluster within the data. Most of the records belong to this category and are handled as a non-anomalous data with the highest level confidence.

4. Medium anomaly above the expected norm. Has the same severity as the Medium anomaly under the expected norm.

5. Significant anomaly above the expected norm. Has the same severity as the Significant anomaly under the expected norm.

This general classification is applied "as-is" to the **Temperature data** which is required to be within the corridor of the comfortable environment. Significant deviation from it may lead to too cold or too hot rooms.
In the case of measuring $\boldsymbol{CO_2}$, the classes denoted as 1 and 2 are eliminated from the process. $\boldsymbol{CO_2}$ concentrations being under the average are not affecting human performance in any way.
Humidity parameter follows the same anomaly detection rules as the **Temperature**. Either very humid or dry conditions may lead to establishing a non-comfort and unacceptable environment inside the building rooms.
The following classification rules were established to detect whether the data is anomalous with **high confidence** (corresponding to classes 1 and 5 from the SLL-RPC output), **low confidence** (classes 2,4) or normal (class 3).

Multi-parameter anomaly detection
As stated before, the developed model operates with 3 different environment parameters – temperature, CO_2 and humidity. The outputs $l_{temp},\ l_{co2},\ l_{hum}$ of each parameter need to be combined with others. The resulting binary label $l_{final} \in (0, 1)$ triggers the system anomaly signal and sends the data to the ventilation controlling systems which can perform an action in order to improve the situation (the corrective action system development is beyond the current project's topic).
The label $l_{final} \in (0, 1)$ is generated by the anomaly detection procedure based on the confidence determined for each of parameters in the previous steps. Simple rules were taken in use:

- If there was detected any of high confidence labels: $l_{temp|hum} \in (1, 5)$ or $l_{CO_2} = 5$, the system anomaly is detected: $l_{final} \leftarrow 1$
- If there was detected one or more of high confidence labels: $l_{temp|hum} \in (2, 4)$ or $l_{CO_2} = 4$, the majority vote principle is applied to the labels $l_{temp},\ l_{co2},\ l_{hum}$. Two or more low confident labels lead to $l_{final} \leftarrow 1$. If there is only one low confident label and two other are being within the normal limits, $l_{final} \leftarrow 0$
- If all labels are located within normal limits $l_{temp},\ l_{CO_2},\ l_{hum} = 3$, then $l_{final} \leftarrow 0$

All low confidence reported during the daily execution can also be logged in the special report that can be further analyzed.

Human teacher supervision of the model
The model is constantly supervised by an analyst who checks the results of the anomaly detection and can edit labels generated for data points. During this procedure the Expert modifies an entry in the Main Database and applies the human set label to it. Improvements made by the human allow making classification results more solid and reliable.

4.6 Storing data and updating it from newest observation

The dataset denoted as T1 is stored in the Main Database in the form of:

- Observation date and time (in the format of dd-mm-yyyy)
- Parameter name
- Value
- Label

We developed the ensemble technique for keeping the dataset of reasonable size and ensuring that new observations are handled with more importance than old ones.
Therefore, made of sliding window approach [41] united with the modified version of the forgetting factor approach [42]. A sliding window of size T corresponds to a constant interval of time for which we keep the records in the DB. It can be defined in values of hours, days or months depending on the system needs. All values which are older than T are dropped.

Chapter 5

Experiments and results

The goal of this section is to represent the achieved results and describe the experiments performed in model training and anomaly detection.
We aim at providing a broad overview of the model nature since, in our opinion, it was not widely explored in a known literature. Therefore, we use three main dataset classes in experiments. The first one named the synthetic or toy data representing a number of known artificial datasets (such as Gaussians) aimed to better demonstrate generalization and prediction properties of the model. The third class unites the datasets often used by researchers who develop SSL algorithms. We will use UCI datasets to compare results produced by our model to the original RPC results. The third class represents a real data received from the sensors over the time. This is the main data that the project is applied to.

5.1 Experiments with synthetic data

5.1.1 T1/T2 experiment

This experiment plays the core role in the verifying RPC-SSL model's eligibility to be applied to the project's specific data and its processing pipeline. According to the project goal definition, the main requirement to SSL-RPC is to be able to handle unlabeled data set having significantly larger length comparing to labeled, and otherwise.
The first case is related to most of the known semi-supervised learning setups and is mostly related to the Model Nightly run, the second case corresponds to a real-time run.
As we previously stated, the back-end handles the data when it arrives, accumulates it before sending to the model for clustering. The length of the data vector L being sent to the model is definitely short since observations arrive consequently with some frequency Θ. For example, the vector of length $|L| = 1..10$ would contain predictions generated every 0.5 – 6 minutes in the current project model setting.
At the same time, authors in [2] clearly state that the labeled dataset T1 is used for training the model. The conformal prediction part including generating a new label for an arrived data point is performed over the unlabeled dataset T2. The Conformal Prediction procedure also includes the step of fitting the model to new labels and generating prototypes basing on new data points. In the author's setting, length $|T2|$ was dominating over $|T1|$. The default proportion of unlabeled data to labeled has been introduced as 3:1 or 4:1. However, researchers do not comment the motivation behind the selection

of this proportion and how the model behaves in other scenarios.
In our daily iteration setting, the proportion of lengths T2:T1 is the opposite, as we aim at delivering real-time or close to real-time results in the first phase of prediction (the Real-time run), and we cannot afford to accumulate the dataset T2 to follow the proportion proposed in [2]. Therefore, before we can proceed with main experiments, we must discover the model's behavior on the opposite proportion, when T2 is smaller than T1, and verify that the model is generally suitable for using with this kind of the dataset.
To answer this question we constructed two types of randomized datasets:

- **G-1**. Randomized toy dataset of two isotopic Gaussians blobs with binary class labels $y = 0, 1$, Standard deviation $\Sigma = 1.0$, and blobs centers located at (3,3) and (0,0). This dataset represents two clusters of data which are located close to each other and in most of the cases could not be separated linearly, see Fig. 5.1.

- **G-2**. Randomized toy dataset of two isotopic Gaussians blobs with binary class labels $y = 0, 1$, standard deviation $\Sigma = 1.0$, and blobs centers located at (2,2) and (0,0). This dataset represents two clusters of data which contain areas where points of opposite classes overlap, see Fig. 5.2.

The intuition behind the selection of this two types of data is based on known characteristics of RPC [6] and RPC-SSL [2]. We assume if the proportion $T2 : T1 >= 3 : 1$ is essential in training the model, then the accuracy of the opposite proportion must be generally worse than for the original. The second reason for using both **G-1** and **G-2** is that in the case of the hardly separable dataset **G-2** the most of the important information for clustering is extracted on the Conformal Prediction step involving and depending on T2.
We generated datasets G-1 and G-2, where the size of G-1 is static $|\text{G-1}| = 50$, while the size |G-2| is changing in every iteration of clustering basing on the proportion T2:T1

$$|\text{G-2}| = (|T1| * 0.1, |T1| * 0.15, , |T1| * 0.2..., |T1| * P)$$

where $P = 3.0$.
The RPC-SSL model's initial prototype selection technique was set to the "point from the data", regularization coefficient λ for RPC training was set to $\lambda = 1/1000$, maximal number of RPC descent steps – 10.

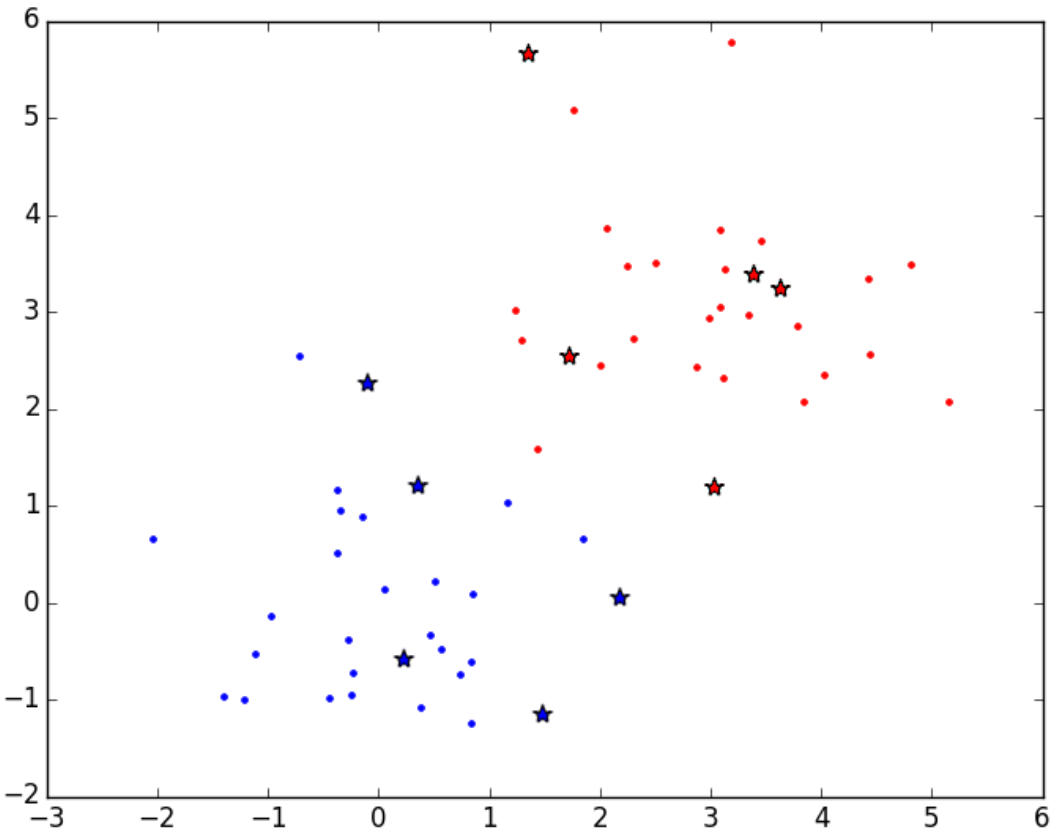

Figure 5.1: Random initialization of G-1 data with partially overlapping blobs with centers (3,3),(0,0)

Figure 5.1 contains an example of the G-1 initialization. The data T1 is plotted with round dots, where the dot color corresponds to the cluster label. The stars denote the dataset T2 and carry the same color coding information.

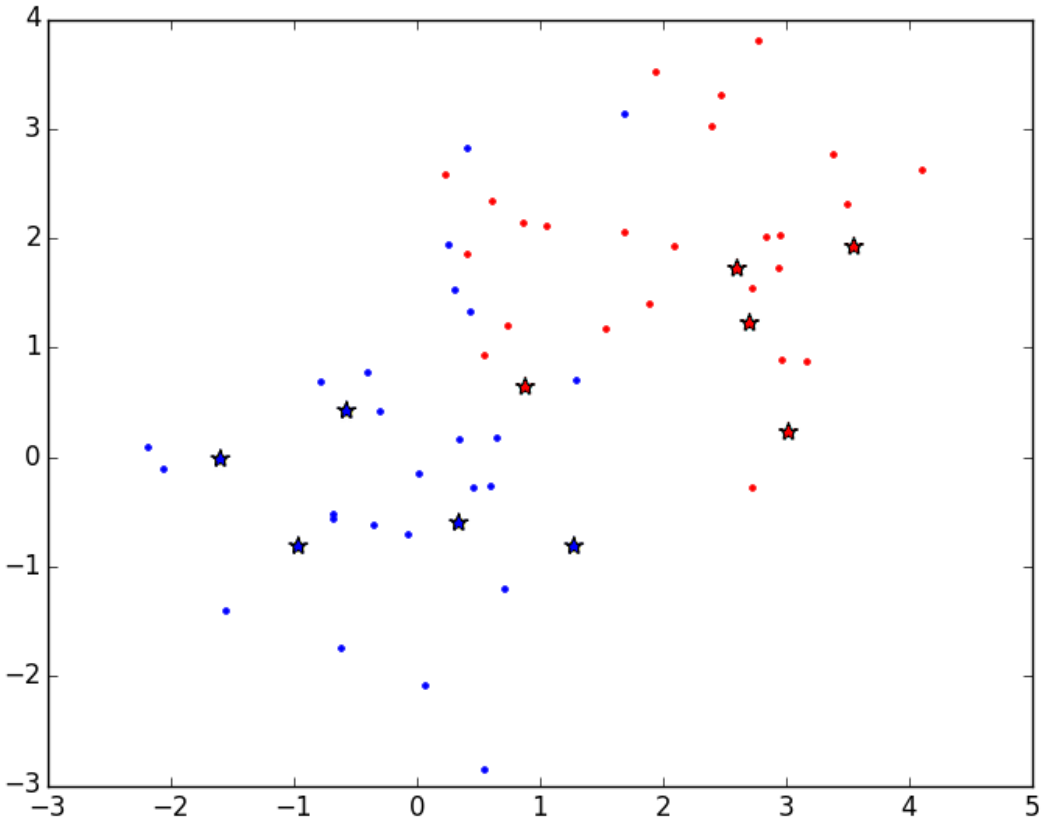

Figure 5.2: Random initialization for G-2 with overlapping blobs with centers (2,2), (0,0)

The goal is to train the model on T1, where $|T1| = 50$ and then apply clustering to T2 which has the size that is changing from 5 elements to $|T1| * P$. We compare final labels produced by the algorithm to the target labels and find accuracy as $acc = num_{correct}/num_{all}$, where $num_{correct}$ is the number of points with the correct label, and num_{all} – the total number of points in T2.
To eliminate deviations related to randomized initialization of Gaussians, we perform 10 independent isolated runs of the RPC-SSL model clustering against T1 and T2 for each fixed size of T2 from N and then produce the averaged accuracy of all 10 runs.
As we previously set maximal proportion T2:T1 to $P = 3$, therefore the length of N is set as $|N| = 30$, which corresponds to 600 independent RPC-SSL runs for datasets G-1 and G-2.
The final accuracy vector containing 30 averaged accuracies for each G-1 and G-2 is plotted against the proportion: $|T2|/|T1|$. See Fig. 5.3:

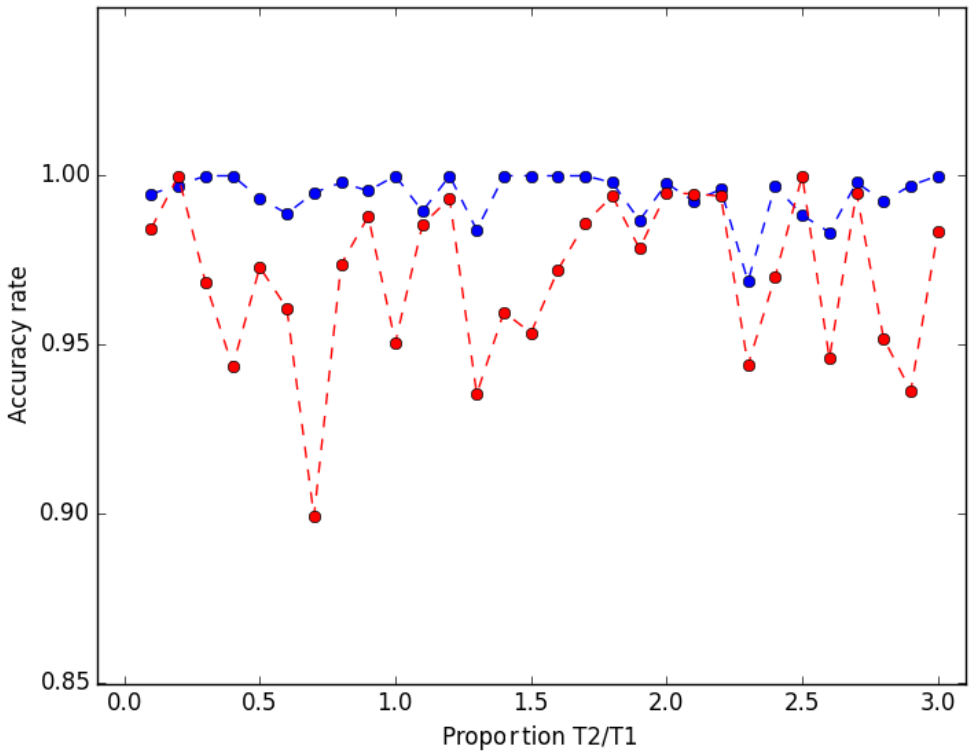

Figure 5.3: Results of two iterations of T1-T2 experiment. Blue points correspond to the G-1, red – to G-2 data

The resulting average accuracy per dataset type:

$$acc_{G1} = 0.9944, \quad acc_{G2} = 0.9702$$

As we can see, the model is capable of being trained on the dataset T1 and correctly fit its complexity using T2 without losing its clustering properties. There is no empirical difference between clustering accuracy for relatively small and large T2 comparing to the size of T1. This property of the model can be explained by its prototype creation procedure. With more points added to T2, RPC-SSL simply creates more prototypes in order to better fit the data.
As the Fig. 5.3 demonstrates more complex data G-2 is also not leading to significantly worse results with small-sized T2 (1-10).
Another observation can be made from the general trends of both accuracy curves in their latter parts – corresponding to T2 two or three times larger than T1. Accuracies reported by runs on datasets do not show trends on improvements in accuracy connected to increasing the size of T2. Therefore, the logical conclusion which can be made from [2] is that T2 must be larger than T1 in order to make training successful - is not satisfied.
Now we can state that SSL-RPC model can be trained and applied to T2 of any proportion or size – from 1 to 1 million points.

5.1.2 SSL-RPC performance comparing to other models

We compare SSL-RPC performance to other semi-supervised and unsupervised models. The synthetic data was generated using the same technique as in the T1-T2 experiment. We run the model on 25 different sizes of the labeled data T1 ranging from 2 to 50 points while the size of the unlabeled data T2 remains the same - 100 points. The dataset is generated identically to the **G-2** case of the T1-T2 experiment – as two Gaussian blobs with centers (2,2) and (0,0)
We chose a number of known models that are being used in supervised and unsupervised learning to compare it to SSL-RPC. The selection was performed to gather models which implement as different approaches to learning as possible.

- Semi-supervised label propagation [44] with RBF kernel (denoted as LP-rbf), where the model parameters was selected accordingly to [44,46] where the kernel function is

$$\exp(-\gamma|x-y|^2),\ \gamma = 1$$

- Semi-supervised label propagation with KNN kernel, where K = 7 denoted as LP-KNN7. This type of the model was selected as it represents a state-of-art graph-based method with different types of kernel function [59].

- Semi-supervised label spreading [45] with RBF function (denoted as LS-rbf). The kernel function and γ parameter are the same as in the LP case.

- Semi-supervised label spreading with KNN-kernel, where K = 7 denoted as LS-KNN7.

- Supervised Support Vector Machine [47] model with RBF kernel. The error penalty parameter $C = 1$, RBF gamma-coefficient was set to $\gamma = 1/\text{num\_features}$. Known labels were fed to SVM model as input alongside with the data, unknown labels were predicted by the model.

- Unsupervised K-means [48]. The initialization strategy "k-means++" [49] was selected. Note: since KMEANS demonstrated significantly worse results than other models, we decided to limit X-axis representing the model accuracy of Fig. 5.4 to 0.65 in order to improve a readability of other models' visualizations.

- Transductive SVM [61] based on the open-source implementation [116] of the TSVM parser. The model kernel is selected as "rbf", the error penalty parameter $C = 1$, the kernel coefficient parameter $\gamma = 0.5$.

- Self-learning wrapper based on the standard SVM-rbf model from the scikit-learn package [117] with the error penalty parameter $C = 1$, RBF gamma-coefficient was set to $\gamma = 1/\text{num\_features}$.

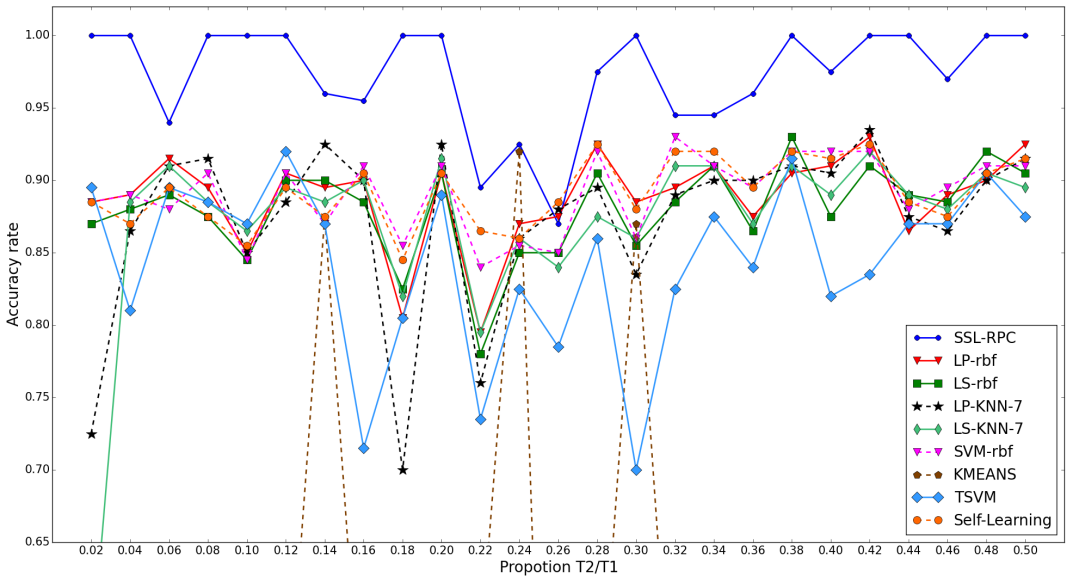

Figure 5.4: Averaged accuracies for SSL-RPC and other models on different T2/T1 size proportions. SSL-RPC (solid blue line) outperforms other models

In the resulting experiment, we found that SSL-RPC outperformed other models in 30 cases of 30 with an average accuracy $acc_{SSL-RPC} = 0.97$. The resulting accuracies for selected pairs $(|T1|, |T2|)$ can be found in the Table 1. The best resulting model is highlighted with a bold font, the second best – cursive.

5.1.3 Time constraints

As was stated previously, the data represented in the form of dissimilarity creates challenges for storing dissimilarity matrices of size $N \times N$, where N is the initial data size. In this experiment, we aim at finding the limits for sizes of datasets T1 and T2 with SSL-RPC.
Since we are interested in relative growths of run time, we initialize the dataset similarly to T1-T2 experiment setting – as two Gaussian blobs with centers (3,3) and (0,0). It will allow to decrease the conformal prediction complexity and allow to estimate time spent on calculations involving operations on pair-wise distances matrices. The initial T1 size is set to 50 points, $|T2| = 25, 50, 75, 100, 125, 150, 200, 250, 300, 500, 1000$ points. We run the model against each pair $(|T1|, |T2|)$ only once.

Model name \ T1 size	2	10	20	30	40	50
SSL-RPC	**1.0**	**1.0**	**1.0**	**1.0**	**0.975**	**1.0**
LP-rbf	0.885	0.85	0.905	*0.885*	0.91	*0.925*
LS-rbf	0.87	0.845	0.905	0.855	0.875	0.905
LP-KNN-7	0.725	0.85	*0.925*	0.835	0.905	0.915
LS-KNN-7	0.565	0.865	0.915	0.86	0.89	0.895
SVM-rbf	0.885	0.845	0.91	0.86	*0.92*	0.91
KMEANS	0.565	0.185	0.48	0.87	0.46	0.485
TSVM	*0.895*	*0.87*	0.89	0.7	0.82	0.875
Self-learning	0.885	0.855	0.905	0.88	0.915	0.915

Table 5.1: Synthetic data results shows SSL-RPC outperforming other models in the case of simple random blob data. The other model demonstrate close results for most of the cases.

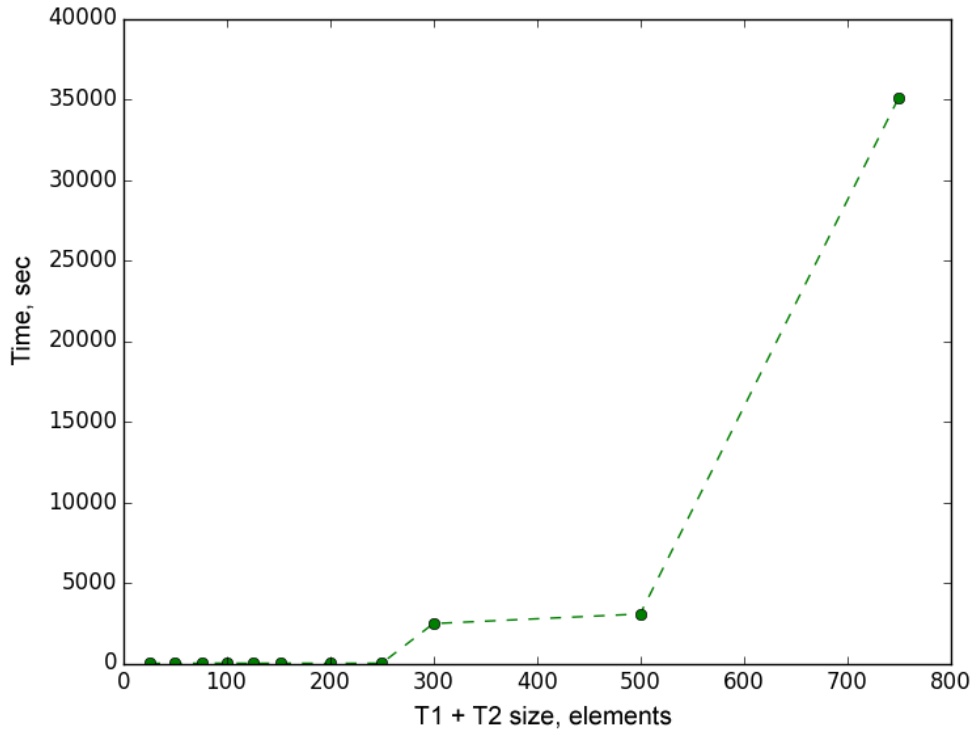

Figure 5.5: Execution time for different sizes of T2. The time growth has an exponential form

The results demonstrate that run times for $|T2| \leq 250$ are relatively small. For $|T2| = 300, 500$ we see significant increase of execution time – up to 50-100 times comparing to the previous size group. Finally, the runtime explodes with 750 points in T2 taking approximately 9 hours to complete the task.

5.1.4 Comparing initialization techniques

We created two Gaussian clusters with centers in [2,2] and [0,0], with standard deviation $\sigma = 1$, the same technique that we used in previous examples. The labeled dataset size $|T1| = 30$, unlabeled $|T2| = 150$ points. We compared two initialization types:

- random initialization of two prototypes – random float numbers were put to their prototype vectors γ_1, γ_2, all summing up to 1.
- dataset initialization – two prototypes were assigned to a randomly labeled point in each class.

The experiment was performed 100 times for each initialization type and the resulting execution time was averaged. Results reported in the Table 5.2.

	Random	Dataset
Execution time(sec)	2.529	**1.316**

Table 5.2: Training speed depending on the prototype initialization type

The dataset initialization was 1.92 times faster comparing to the random initialization.

5.1.5 Regularized SSL-RPC experiment

One of our contributions to the SSL-RPC model is the proposal to use a regularization coefficient $\frac{1}{\lambda}$ on the Stochastic gradient descent update steps for prototype vectors. During the experiments we noticed an instability in the SSL-RPC performance related to the wide range of a resulting accuracies between two runs on random Gaussian datasets. From the plots, we spotted that in various cases prototypes were pushed or pulled too far away from actual data in wrong locations and the model diverged. We plotted the exact updated prototype positions and realized that the model was performing too large steps when updating prototypes. We decided to regularize update rules with $\frac{1}{\lambda}$ and achieved a stabilized model performance.
To demonstrate the described behavior of the model of the non-regularized SSL-RPC we run the λ-experiment. We varied the value of λ from 1 (the non-regularized case) to 1/2000. For each particular λ value, 30 runs were completed on the dataset **G-2** of two random Gaussians with centers (2,2) and

(0,0). The averaged run accuracies and standard deviations were calculated and plotted for $1 \leq \lambda \leq 500$. The results are represented in the Table 5.3 and Fig. 5.6. Not included results for $600 \leq \lambda \leq 2000$ follow the general trend with a stable average accuracy $acc_{600 \leq \lambda \leq 2000} = 0.9603$ and standard deviation $std_{600 \leq \lambda \leq 2000} = 0.0746$.

As it can be seen on the Fig. 5.6, the standard deviation of performed runs

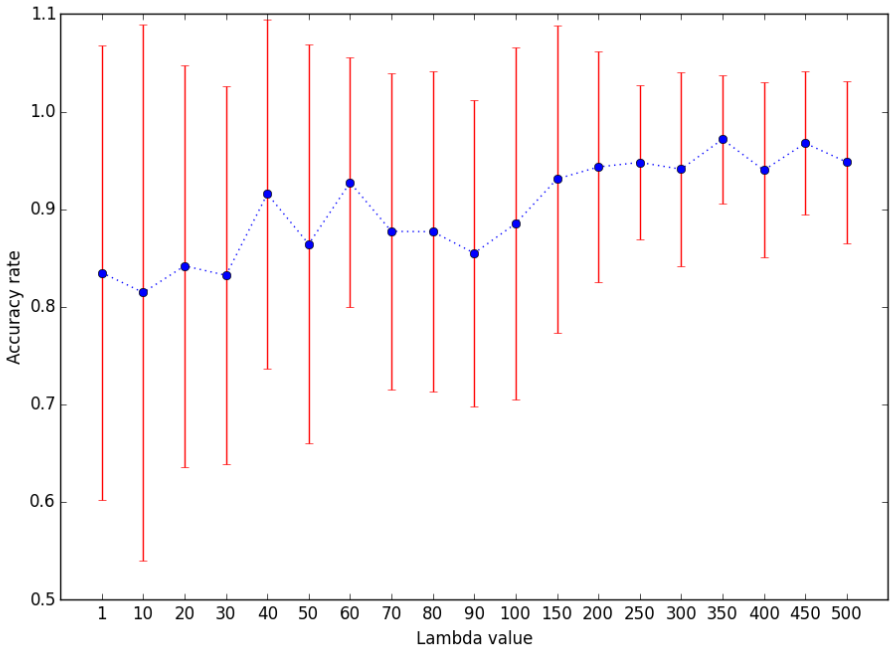

Figure 5.6: λ-experiment. The red vertical bar over each point corresponds to the standard deviation

tends to decrease with increasing λ in the regularization coefficient $\frac{1}{\lambda}$.
For values $1 \leq \lambda \leq 250$, the averaged accuracies are generally worse than for larger values of λ, and the standard deviation is located in the range $[0, 12, 0.27]$. While for larger λ-s, the average deviation is being in limits $[0.03, 0.08]$. This experiment demonstrates the efficiency of the introduced regularization criteria $\frac{1}{\lambda}$. At the same time, its important to tune the overall number of the gradient descent steps performed by the SSL-RPC model in one iteration. It has to be increased with increasing the value of λ. The optimal configuration depends on the requirements to performance/accuracy trade-off in each particular task.

λ	Accuracy	Standard deviation
1	0.835	0.233
10	0.815	0.275
20	0.842	0.205
30	0.832	0.194
40	0.916	0.179
50	0.864	0.204
60	0.927	0.128
70	0.877	0.162
80	0.877	0.164
90	0.855	0.157
100	0.885	0.180
150	0.931	0.158
200	0.943	0.118
250	0.948	0.079
200	0.941	0.099
350	0.972	0.066
400	0.940	0.089
450	0.970	0.073
500	0.948	0.083

Table 5.3: SSL-RPC results with varying λ. The larger λ is, the better results the model demonstrates, increasing the accuracy and decreasing the standard deviation of classification results

5.1.6 Model mechanics explained

As the part of experiments on the toy data, we aim at demonstrating the model's principle of work on the practical example and also verifying that prototype learning is occurred as predicted.
In order to construct T1 we set three random Gaussian data blobs with centers in (-1,-1), (0,0) and (1,1) carrying different labels, 50 points in each. T2 is represented by 30 random points created in the same way – 10 for each class. On the first iteration the initial prototypes are created – we mark them with small transparent circles on the plot. The larger marker size corresponds to the latter iteration while initial prototypes are the smallest. Prototype positions are defined by points in T2 which formed the set Beta containing data with low confidence/credibility of belonging to one or another cluster (Fig. 5.7). On the next plot (Fig. 5.8), with a different initialization of the data, we also marked points of T2 as stars and prototype positions as circles. Under the main data layer we built a simple clustermap – the cell grid color reflects the possible label assignment according to current prototype positions. We have to note that it does not mean that any T2 point located in one of a colored clusters would receive the same label. On the conformal prediction step, this assignment would

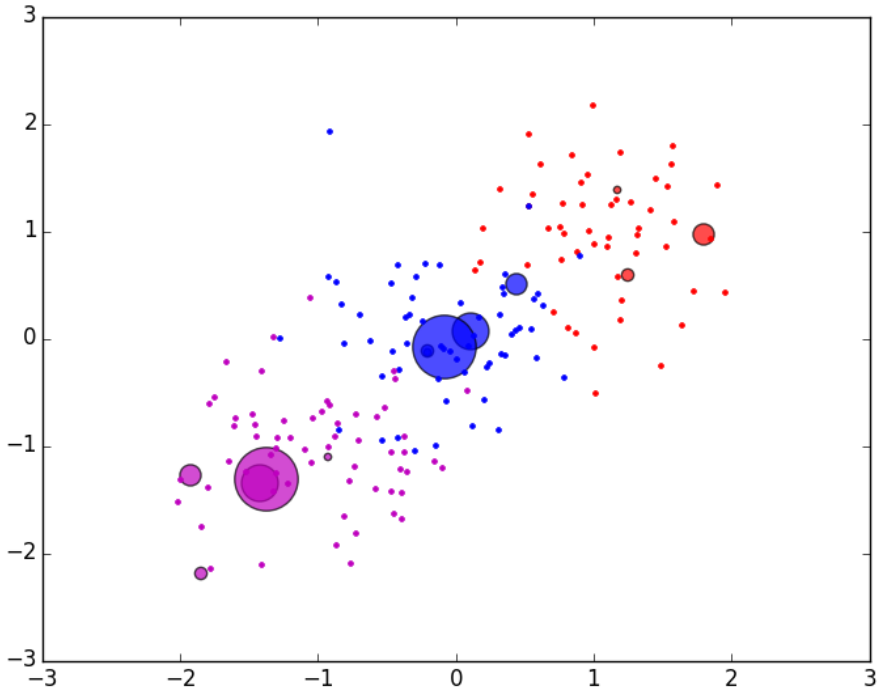

Figure 5.7: Results of four iterations of RPC. The prototype creation procedure takes place 4 times for purple and blue cluster and twice for the red cluster

shift borders of each cluster.
In general, this visualization provides us an insight on the projected model behavior and predicted cluster borders.
Remembering the basic principle of GLVQ [1] stating that prototypes positions are being changed with pushing or pulling powers making them better fit the data. From the position of starred points we can observe this influence – in the bottom left corner blue points from T1 and T2 are interfering with the pink cluster. On the plot, large pink circles show that new prototypes of this cluster were created at around the center of the area where T2 points belong. Also, the first prototype was created closer to the blue border in order to compensate the influence of blue points and push the class borders away from the center.
Another observation that could be made is that the border of the Beta region is highly related to T2 points – red starred dots above the red Beta border are placed closer to the cluster center.

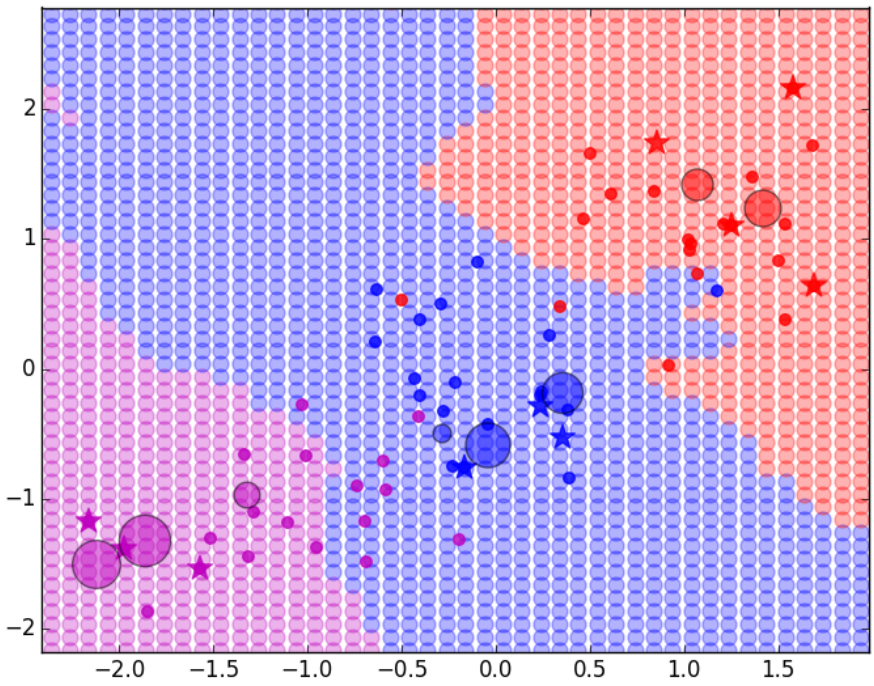

Figure 5.8: RPC clustermap for 3 Gaussian blobs colored accordingly to the class assignment. Prototype positions marked as transparent circles; T1 points shown as dots; T2 points correspond to stars.

Locations of T2 points (starred) changed the cluster borders significantly. We can see that some of pink points appeared to be on the blue side. It happened due to the dense concentration of blue T2 points (starred) next to them. Three blue prototypes were created in a nearly the same region in order to push blue cluster borders away from the starred points.

5.2 Experiments with real-word datasets

We selected three UCI [112] datasets widely used by Semi-Supervised Learning researchers, according to the publication statistics from the UCI website.

- **Haberman's Survival Dataset** [113]. Consists of 306 samples of the study of breast cancer surgery patients with 3 features available – an age of a patient, a year of the operation, a number of positive axillary nodes detected. The samples were labeled as "0" if a patient lived more than 5 years after the operation, and "1" in the case if a patient died within 5 years after the surgery.

- **Breast Cancer Wisconsin Diagnosis data** [114], often referred to as WDBC dataset. It consists of 569 records corresponding to images of tumors and 30 features composed of the image information (i.e. tumor radius, texture, perimeter etc.) The class labels are "0" (benign tumor) and "1" (malicious tumor). The nature of the dataset allows to research how well models perform on the multidimensional data (30x).

- **Ecoli dataset** [115] composed of 336 data records with 8 numerical attributes each. We selected all records that belong to 5 most common classes. 3 classes "omL", "imL", "imS" containing 5, 2 and 2 records correspondingly were left out of the experiment. The remaining classes were encoded with consecutive numbers from 0 to 4. This dataset was selected to demonstrate abilities of models to work with a large amount of attributes in multi-class environment.

For each of the dataset we performed 30 independent runs sampling randomly 10 labeled and 100 unlabeled samples. We used this proportion as we aim our results to be comparable to other SSL-related research that in most of the cases used a relatively small labeled datasets, as we concluded from the analyzed literature [25,59].
Also, as a continuation of the λ-experiment, we added the non-regularized SSL ($\lambda = 1$) to the model's list.
The resulting accuracy for each model is the average accuracy of all 30 runs.
The results are represented in Table 5.4:

Model \ UCI data	Haberman	WDBC	Ecoli
SSL-RPC$_{\lambda=1000}$	**0.85**	**0.907**	**0.805**
SSL-RPC$_{\lambda=1}$	*0.805*	0.866	0.61
LP-rbf	0.622	0.625	0.462
LS-rbf	0.605	0.625	*0.769*
LP-knn-7	0.647	0.715	0.641
LS-knn-7	0.621	0.88	0.749
SVM(supervised)	0.64	0.839	0.383
KMEANS	0.49	0.479	0.222
TSVM	0.57	0.543	0.62
Self-Learning SVM	0.62	*0.89*	0.44

Table 5.4: UCI data results. SSL-RPC with regularization demonstrates a leading performance. Non-regularized SSL-RPC shows above the average results

SSL-RPC clearly outperformed all other methods. In the case of a WDBC dataset containing a large number of features, SSL-RPC showed the accuracy of 0.85 when the closest result from one of the state-of-art methods was 0.647. Good performance was demonstrated by the original supervised and semi-supervised SVM methods, especially when applying self-learning wrapping technique on the WDBC data.
Among all the models, only regularized SSL-RPC demonstrated a stable performance without a dependence on the dataset nature. Results of non-regularized SSL-RPC were worse than its regularized alternative, although clearly above the average.
Referring to the original results that researchers of the basic SSL-RPC achieved (Table 5.4), we can see that in the SSL-RPC method with our contribution significantly outperforms the original method on the Haberman data. In the case of the WDBC data set, the original method performs slightly better. In our setup, we performed 30 independent runs sampling randomly 10 labeled and 100 unlabeled samples. It is not stated in [2], how exactly the researchers divided the data set and how many samples did they use. Therefore, the comparison of results can give only the initial understanding on the comparison of how the models perform.

Model \ UCI data	Haberman	WDBC
SSL-RPC$_{\lambda=1000}$	**0.85**	0.907
SSL-RPC$_{\text{original}}$	0.73	**0.933**

Table 5.5: SSL-RPC results comparison between the developed model and the original method

5.3 Experiments with sensors data

The sensor data received over the time was mostly containing normal values or such that only slightly deviate from the expectation. We registered only 5 cases of low-confidence anomalies (classes 2,4) and 1 case of a high-confidence anomaly (class 1, temperature).
We manually generated some disturbances in the data by heating up the air and increasing humidity. The model successfully reacted on changes and classified them according to the expectation in 10 cases of 10.
Basing on 1400 model runs, the average classification time was 3.2 seconds for $|T1| = 15$, $|T2| = 5$.
The outlier detection algorithm reacted in 0.5 seconds after getting results from the model and successfully raised the alarm when the situation required a human intervention.
In the nightly runs when $|T1| = 15$, $|T2| = 200$ the average running time was 300 seconds. 98.2 % of unlabeled data was classified correctly.

5.4 Open-source community contribution

We implemented RPC classifiers for supervised / semi-supervised learning (incl. Conformal prediction) in Python as separate modules that can be imported as external packages to any of Python programs. We followed the standard Scikit-Learn [117] naming guidelines for model functionality:

- RPC_classifier.fit(T1,T2,T1Labels) – the method fits RPC classifier basing on T2 only. Takes datasets T1 and T2 as inputs and fits the model using supervised labels T1Labels.
- RPC_classifier.predict() – the method fits RPC classifier to the T2 data basing on the T1 labels
- RPC_classifier.score(T2Labels) – the method evaluates performance of the RPC classifier if T2Labels are given.

The implemented SSL-RPC package is available for downloading and sharing at the public repository: https://github.com/doshyt/ssl-rpc-regularized.git

Chapter 6

Discussion and conclusions

6.1 SSL models and proposed directions

In this project we designed and created a complex framework for real-time data classification based on the SSL-RPC model enhanced with a human expert involvement as a source of the labeled data. The proposed framework is ready for integration to an industry setting and can be used as-is in an industrial setup for any kind of data-based anomaly detection / data clustering. The proposed setting can be used for various intelligent systems which unite machine and human intelligence. In our opinion, involving a human as a source of labels (Oracle) in the Semi-Supervised Learning specificity can help build more complex and accurate models. We believe, the research of human involvement into automated decision making in SSL-setting can become a rewarding research direction for the further use of different SSL models.
Another starting point for a research might be an introduction of a forgetting factor [41,42] which can be applied to the labeled data provided by the expert. In the case, some of labeled data points can be acquired from the unlabeled data after classifying it and receiving and approval from the expert, it might be possible to remove outdated points from the labeled data making it contain the most recent and important points only.

6.2 SSL-RPC development proposals

We designed a method for extending prototypes to unknown data which was not covered by the original papers [2,6].

- Model instability was observed and a new regularization criteria $\frac{1}{\lambda}$ was introduced to the stochastic gradient update rules which added more stability and accuracy to the model.
- The prototype creation and updating method was developed in order to fill the gap of the paper describing the initial method and leaving this part out of its scope. The proposed method shown its efficiency and stability
- The new initial prototype initialization technique allowed to decrease the SSL-RPC running time twice
- Time and performance limits of both enhanced and standard SSL-RPC were explored

- The developed model was incorporated into the proposed framework collecting the real-data from sensors and successfully applied to it classifying anomalies in the required time limits, nearly in real-time. The model demonstrated its capability to work with numerous amount of features which means, expanding the rolling window length and storing longer time-series will not cause notable time difference in classification. This feature makes the method attractive for the usage with classic time-series, DNA sequences, clustering long series of records transformed into the form of dissimilarities

Basing on our experience with SSL-RPC, we see various research opportunities in developing more intelligent stopping criteria for SSL-RPC model:

- Gradient descent exit criteria. Currently, we use fixed values which were obtained empirically by researchers in [2,6] and from our experiments. The model stops descending after 10 iterations or in the case when the value of the error function of the new iteration $E_{new} - E_{old} > 0.5$.

- Size of the β-region in Conformal prediction. In [2], researchers empirically estimated that the model should stop creating new prototypes from the low-confidence/credibility region when it size $|\beta| = 5$. This conclusion does not take into account any information about the nature of the dataset, number of points in T2 and other possible parameters

- Clustering fit criteria. We proposed to use the Silhouette criteria to find the proper match of the clustering to the data. At the same time, other clustering measures can be explored

- Distance functions in SSL-RPC research. In our work, we used the basic Euclidean distance to form a dissimilarity matrix between data points. As far as we know, a research of the model performance and use cases when using different distance measures has not yet been conducted, and it might provide insights for better reasoning of a choice of a distance function depending on the task SSL-RPC is applied to

From the experiments we noticed that SSL-RPC initial prototype assignment plays a significant role in the model performance (both run time and reported accuracy). Therefore, the better the initial prototype fits data, the better the final result is. We propose to initialize a non-equal number of prototypes on the initial step, basing on the available labeled data. The number of prototypes for each class can correspond to the proportional class distribution in T1.

6.2.1 SSL-RPC drawbacks

Despite a superior performance over standard models, SSL-RPC has some significant drawbacks that limit its wide usage in possible applications.
The use of the pair-wise dissimilarity matrix makes it operating with 25000

numbers matrix in case of only 500 records present in the initial data. When the gradient descent is calculated, the distances to prototypes from each of points are estimated and re-estimated for each iteration of RPC. The process takes extremely long amounts of time in case when the dataset is larger than 500 samples. However, the dimensionality of a data sample does not add an additional complexity to the model steps since RPC uses a dissimilarity matrix. This significant limitation might block a wider usage of SSL-RPC. Therefore, improvements to existing methods or development of new techniques for updating prototype positions are required.
Another possible drawback is related to the LVQ-methods family disadvantages. Researchers in [99] point out that LVQ methods rely on empirical properties of pushing and pulling prototypes to or from a window on the midplane, and therefore, learning behavior of the model is not well understood [25]. Comparing to LVQ, other SSL methods with better proven mathematical grounds might be preferable to use, since they do not rely on heuristics.

6.3 Latest scientific achievements

Based on the recent development of the LVQ model family, new methods and extensions of SSL-RPC were proposed, such as SSL-RPC based on self-learning [118], sparse conformal prediction for SSL-RPC [119] and learning prototype models with kernels [120]. These publications provide excellent opportunities to improve our contributions by employing the proposed methods to time-series and different types of real-time data classification which now is possible using the proposed open-source implementation of SSL-RPC.

Chapter 7

References

[1] Sato, Atsushi, and Keiji Yamada. "Generalized learning vector quantization." Advances in neural information processing systems (1996): 423-429.

[2] Zhu, Xibin, Frank-Michael Schleif, and Barbara Hammer. "Semi-supervised vector quantization for proximity data." In Proc. of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2013), Louvain-La-Neuve, Belgium, pp. 89-94. 2013.

[3] Cha S. Comprehensive survey on distance/similarity measures between probability density functions. Int J Math Models Meth Appl Sci. 1:300–307. 2007

[4] Dinh, Cuong V., Robert PW Duin, and Marco Loog. "A study on semi-supervised dissimilarity representation." In Pattern Recognition (ICPR), 2012 21st International Conference on, pp. 2861-2864. IEEE, 2012.

[5] R. Duin and E. Pekalska. The dissimilarity representation for structural pattern recognition. In Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, volume 7042, pages 1–24, 2011

[6] Hammer, Barbara, Frank-Michael Schleif, and Xibin Zhu. "Relational extensions of learning vector quantization." In Neural Information Processing, pp. 481-489. Springer Berlin Heidelberg, 2011.

[7] Hammer B. and Hasenfuss A.: Topographic Mapping of Large Dissimilarity Data Sets. Neural Computation 22(9), 2229-2284 (2010)

[8]. E. Pekalska and R.P.W. Duin. "The Dissimilarity Representation for Pattern Recognition". Foundations and Applications. World Scientific, Singapore, December 2005.

[9] Kohonen, Teuvo. "The self-organizing map." Neurocomputing 21, no. 1 (1998): 1-6.

[10] Biehl, Michael, Barbara Hammer, and Petra Schneider. "Matrix learning in learning vector quantization." Insitute of Informatics, Clausthal University of Technology (2006): 06-14.

[11] Hammer, Barbara, Marc Strickert, and Thomas Villmann. "Supervised neural gas with general similarity measure." Neural Processing Letters 21, no. 1 (2005): 21-44.

[12] Kohonen, Teuvo. "Improved versions of learning vector quantization." In Neural Networks, 1990., 1990 IJCNN International Joint Conference on, pp. 545-550. IEEE, 1990.

[13] Tan, Pang-Ning, Michael Steinbach, and Vipin Kumar. "Introduction to data mining Addison-Wesley." (2005): 76-79.

[14] Pękalska, Elżbieta, and Robert PW Duin. The dissimilarity representation for pattern recognition: foundations and applications. No. 64. World Scientific, 2005.

[15] Pękalska, Elżbieta, and Robert PW Duin. "Dissimilarity representations allow for building good classifiers." Pattern Recognition Letters 23, no. 8 (2002): 943-956.

[16] Pekalska, Elzbieta, and Robert PW Duin. "Beyond traditional kernels: Classification in two dissimilarity-based representation spaces." Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 38, no. 6 (2008): 729-744.

[17] Kauppinen, Timo. "District level control center - a possibility for ongoing commissioning", NCBC Denver, presentation material (2013)

[18] Civis project - european union 7th framework programme. http://www.civisproject.eu/. Accessed: 2014-06-09

[19] Janne Peltonen. Presentation on vtt otaniemi greencampus summary, 2013.

[20] Biehl, Michael, Anarta Ghosh, and Barbara Hammer. "Learning vector quantization: The dynamics of winner-takes-all algorithms." Neurocomputing 69, no. 7 (2006): 660-670.

[21] Biehl, Michael, Anarta Ghosh, and Barbara Hammer. "Dynamics and generalization ability of LVQ algorithms." The Journal of Machine Learning Research 8 (2007): 323-360.

[22] Grbovic, Mihajlo, and Slobodan Vucetic. "Learning vector quantization with adaptive prototype addition and removal." In Neural Networks, 2009. IJCNN 2009. International Joint Conference on, pp. 994-1001. IEEE, 2009.

[23] Kohonen, Teuvo. "The self-organizing map." Proceedings of the IEEE 78, no. 9 (1990): 1464-1480.

[24] Hammer, Barbara, and Thomas Villmann. "Generalized relevance learning vector quantization." Neural Networks 15, no. 8 (2002): 1059-1068.

[25] Oliver Chapelle, Bernhard Scholköpf, and Alexander Zien. "Semi-supervised Learning". MIT Press (2006).

[26] Hassan Farhangi. "The path of the smart grid". Power and Energy Magazine, IEEE 8, 1 (2010), 18–28.

[27] Heckbert, Paul S., and Michael Garland. Survey of polygonal surface simplification algorithms. Carnegie-Mellon University, Pittsburgh, School of Computer Science, 1997.

[28] Neuhaus, Michel, and Horst Bunke. "Edit distance-based kernel functions for structural pattern classification." Pattern Recognition 39, no. 10 (2006): 1852-1863.

[29] Chen, Yihua, Eric K. Garcia, Maya R. Gupta, Ali Rahimi, and Luca Cazzanti. "Similarity-based classification: Concepts and algorithms." The Journal of Machine Learning Research 10 (2009): 747-776.

[30] Jones, Eric, Travis Oliphant, and Pearu Peterson. "SciPy: Open source scientific tools for Python." (2014).

[31] Schittenkopf, Christian, Peter Tiňo, and Georg Dorffner. "The benefit of information reduction for trading strategies." Applied Economics 34, no. 7 (2002): 917-930.

[32] LeCun, Yann A., Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. "Efficient backprop." In Neural networks: Tricks of the trade, pp. 9-48. Springer Berlin Heidelberg, 2012.

[33] Rousseeuw, Peter J. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis." Journal of computational and applied mathematics 20 (1987): 53-65.

[34] Hubert, Lawrence J., and Joel R. Levin. "A general statistical framework for assessing categorical clustering in free recall." Psychological bulletin 83, no. 6 (1976): 1072.

[35] Caliński, Tadeusz, and Jerzy Harabasz. "A dendrite method for cluster analysis." Communications in Statistics-theory and Methods 3, no. 1 (1974): 1-27.

[36] Dunn, Joseph C. "Well-separated clusters and optimal fuzzy partitions." Journal of cybernetics 4, no. 1 (1974): 95-104.

[37] Guerra, Luis, Víctor Robles, Concha Bielza, and Pedro Larrañaga. "A comparison of clustering quality indices using outliers and noise." Intelligent Data Analysis 16, no. 4 (2012): 703-715.

[38] Ansari, Zahid, M. F. Azeem, Waseem Ahmed, and A. Vinaya Babu. "Quantitative evaluation of performance and validity indices for clustering the web navigational sessions." World of Computer Science and Information Technology Journal 1, no. 5 (2011): 217-226.

[39] Liu, Yanchi, Zhongmou Li, Hui Xiong, Xuedong Gao, and Junjie Wu. "Understanding of internal clustering validation measures." In Data Mining (ICDM), 2010 IEEE 10th International Conference on, pp. 911-916. IEEE, 2010.

[40] Chapelle, Olivier, Vikas Sindhwani, and Sathiya S. Keerthi. "Optimization techniques for semi-supervised support vector machines." The Journal of Machine Learning Research 9 (2008): 203-233.

[41] Keogh, Eamonn, Selina Chu, David Hart, and Michael Pazzani. "Segmenting time series: A survey and novel approach." Data mining in time series databases 57 (2004): 1-22.

[42] Graepel, Thore, and Nicol N. Schraudolph. "Stable adaptive momentum for rapid online learning in nonlinear systems." In Artificial Neural Networks—ICANN 2002, pp. 450-455. Springer Berlin Heidelberg, 2002.

[43] Guyon, Isabelle. "Design of experiments of the NIPS 2003 variable selection benchmark." In NIPS 2003 workshop on feature extraction and feature selection. 2003.

[44] Zhu, Xiaojin, and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.

[45] Zhou, Dengyong, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. "Learning with local and global consistency." Advances in neural information processing systems 16, no. 16 (2004): 321-328.

[46] Delalleau, Olivier, Yoshua Bengio, and Nicolas Le Roux. "Efficient non-parametric function induction in semi-supervised learning." In Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, pp. 96-103. 2005.

[47] Hearst, Marti A., Susan T. Dumais, Edgar Osman, John Platt, and Bernhard Scholkopf. "Support vector machines." Intelligent Systems and their Applications, IEEE 13, no. 4 (1998): 18-28.

[48] Hartigan, John A., and Manchek A. Wong. "Algorithm AS 136: A k-means clustering algorithm." Applied statistics (1979): 100-108.

[49] Arthur, David, and Sergei Vassilvitskii. "k-means++: The advantages of careful seeding." In Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, pp. 1027-1035. Society for Industrial and Applied Mathematics, 2007.

[50] Assunção, Marcos D., Rodrigo N. Calheiros, Silvia Bianchi, Marco AS Netto, and Rajkumar Buyya. "Big Data computing and clouds: Trends and future directions." Journal of Parallel and Distributed Computing 79 (2015): 3-15.

[51] Turner, Vernon, John F. Gantz, David Reinsel, and Stephen Minton. "The digital universe of opportunities: Rich data and the increasing value of the internet of things." Framingham (MA): IDC (2014).

[52] Chen, Min, Shiwen Mao, and Yunhao Liu. "Big data: A survey." Mobile Networks and Applications 19, no. 2 (2014): 171-209.

[53] Chen, Hsinchun, Roger HL Chiang, and Veda C. Storey. "Business Intelligence and Analytics: From Big Data to Big Impact." MIS quarterly 36, no. 4 (2012): 1165-1188.

[54] Aman, Saima, Yogesh Simmhan, and Viktor K. Prasanna. "Energy management systems: state of the art and emerging trends." Communications Magazine, IEEE 51, no. 1 (2013): 114-119.

[55] Saha, Balaram, and Divesh Srivastava. "Data quality: The other face of big data." In Data Engineering (ICDE), 2014 IEEE 30th International Conference on, pp. 1294-1297. IEEE, 2014.

[56] Woodall, Philip, Alexander Borek, Jing Gao, Martin Oberhofer, and Andy Koronios. "An Investigation of How Data Quality is Affected by Dataset

Size in the Context of Big Data Analytics." In Proceedings of the International Conference on Information Quality. 2014.

[57] Kaisler, Stephen, Frank Armour, Juan Antonio Espinosa, and William Money. "Big data: Issues and challenges moving forward." In System Sciences (HICSS), 2013 46th Hawaii International Conference on, pp. 995-1004. IEEE, 2013.

[58] Aggarwal, Charu C. Managing and mining sensor data. Springer Science & Business Media, 2013.

[59] Zhu, Xiaojin. "Semi-supervised learning literature survey." (2005).

[60] Nigam, Kamal, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. "Text classification from labeled and unlabeled documents using EM." Machine learning 39, no. 2-3 (2000): 103-134.

[61] Bennett, Kristin, and Ayhan Demiriz. "Semi-supervised support vector machines." Advances in Neural Information processing systems (1999): 368-374.

[62] Blum, Avrim, and Shuchi Chawla. "Learning from labeled and unlabeled data using graph mincuts." (2001): 19.

[63] Blum, Avrim, and Tom Mitchell. "Combining labeled and unlabeled data with co-training." In Proceedings of the eleventh annual conference on Computational learning theory, pp. 92-100. ACM, 1998.

[64] Yarowsky, David. "Unsupervised word sense disambiguation rivaling supervised methods." In Proceedings of the 33rd annual meeting on Association for Computational Linguistics, pp. 189-196. Association for Computational Linguistics, 1995.

[65] Riloff, Ellen, Janyce Wiebe, and Theresa Wilson. "Learning subjective nouns using extraction pattern bootstrapping." In Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4, pp. 25-32. Association for Computational Linguistics, 2003.

[66] Wei, Li, and Eamonn Keogh. "Semi-supervised time series classification." In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 748-753. ACM, 2006.

[67] Kim, Seyoung, Padhraic Smyth, and Stefan Luther. "Modeling waveform shapes with random effects segmental hidden Markov models." In Proceedings of the 20th conference on Uncertainty in artificial intelligence, pp.

309-316. AUAI Press, 2004.

[68] Rodríguez, Juan J., and Carlos J. Alonso. "Interval and dynamic time warping-based decision trees." In Proceedings of the 2004 ACM symposium on Applied computing, pp. 548-552. ACM, 2004.

[69] Sakoe, Hiroaki, and Seibi Chiba. "Dynamic programming algorithm optimization for spoken word recognition." Acoustics, Speech and Signal Processing, IEEE Transactions on 26, no. 1 (1978): 43-49.

[70] Keogh, Eamonn, and Chotirat Ann Ratanamahatana. "Exact indexing of dynamic time warping." Knowledge and information systems 7, no. 3 (2005): 358-386.

[71] Nanopoulos, Alex, Rob Alcock, and Yannis Manolopoulos. "Feature-based classification of time-series data." International Journal of Computer Research 10, no. 3 (2001).

[72] Boeckmann, Brigitte, Amos Bairoch, Rolf Apweiler, Marie-Claude Blatter, Anne Estreicher, Elisabeth Gasteiger, Maria J. Martin et al. "The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003." Nucleic acids research 31, no. 1 (2003): 365-370.

[73] Neuhaus, Michel, and Horst Bunke. "Edit distance-based kernel functions for structural pattern classification." Pattern Recognition 39, no. 10 (2006): 1852-1863.

[74] Mitchell, Tom M. "Machine learning." McGraw Hill series in computer science (1997): I-XVII.

[75] Michalski, Ryszard S., Jaime G. Carbonell, and Tom M. Mitchell, eds. Machine learning: An artificial intelligence approach. Springer Science & Business Media, 2013.

[76] Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalkar. Foundations of machine learning. MIT press, 2012.

[77] Carbonell, Jaime G., Ryszard S. Michalski, and Tom M. Mitchell. "An overview of machine learning." In Machine learning, pp. 3-23. Springer Berlin Heidelberg, 1983.

[78] Biehl, Michael, Barbara Hammer, and Thomas Villmann. "Prototype-based models in machine learning." Wiley Interdisciplinary Reviews: Cognitive Science (2016).

[79] Strickert, Marc, Barbara Hammer, Thomas Villmann, and Michael Biehl. "Regularization and improved interpretation of linear data mappings and adaptive distance measures." In Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on, pp. 10-17. IEEE, 2013.

[80] Martinetz, Thomas M., Stanislav G. Berkovich, and Klaus J. Schulten. "Neural-gas' network for vector quantization and its application to time-series prediction." Neural Networks, IEEE Transactions on 4, no. 4 (1993): 558-569.

[81] Bishop, Christopher M., Markus Svensén, and Christopher KI Williams. "GTM: The generative topographic mapping." Neural computation 10, no. 1 (1998): 215-234.

[82] Gisbrecht, Andrej, and Barbara Hammer. "Data visualization by nonlinear dimensionality reduction." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 5, no. 2 (2015): 51-73.

[83] Chapelle, Olivier. "A Taxonomy of Semi-Supervised Learning Algorithms." In Yahoo!. 2005.

[84] McLachlan, G. J. "9 The classification and mixture maximum likelihood approaches to cluster analysis." Handbook of statistics 2 (1982): 199-208.

[85] Murray, G. D., and D. M. Titterington. "Estimation problems with data from a mixture." Applied Statistics (1978): 325-334.

[86] Nigam, Kamal, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. "Text classification from labeled and unlabeled documents using EM." Machine learning 39, no. 2-3 (2000): 103-134.

[87] Fujino, Akinori, Naonori Ueda, and Kazumi Saito. "A hybrid generative/discriminative approach to semi-supervised classifier design." In Proceedings of the National Conference on Artificial Intelligence, vol. 20, no. 2, p. 764. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[88] Fujino, Akinori, Naonori Ueda, and Kazumi Saito. "Semisupervised learning for a hybrid generative/discriminative classifier based on the maximum entropy principle." Pattern Analysis and Machine Intelligence, IEEE Transactions on 30, no. 3 (2008): 424-437.

[89] Cozman, Fabio Gagliardi, Ira Cohen, and Marcelo Cesar Cirelo. "Semi-supervised learning of mixture models." In ICML, pp. 99-106. 2003.

[90] Chapelle, Olivier, and Alexander Zien. "Semi-Supervised Classification by Low Density Separation." In AISTATS, pp. 57-64. 2005.

[91] Vapnik, Vladimir Naumovich, and Vlamimir Vapnik. Statistical learning theory. Vol. 1. New York: Wiley, 1998.

[92] Joachims, Thorsten. Making large scale SVM learning practical. Universität Dortmund, 1999.

[93] Joachims, Thorsten. "Transductive learning via spectral graph partitioning." In ICML, vol. 3, pp. 290-297. 2003.

[94] Belkin, Mikhail, and Partha Niyogi. "Laplacian eigenmaps for dimensionality reduction and data representation." Neural computation 15, no. 6 (2003): 1373-1396.

[95] Subramanya, Amarnag, and Partha Pratim Talukdar. "Graph-based semi-supervised learning." Synthesis Lectures on Artificial Intelligence and Machine Learning 8, no. 4 (2014): 1-125.

[96] Blum, Avrim, and Tom Mitchell. "Combining labeled and unlabeled data with co-training." In Proceedings of the eleventh annual conference on Computational learning theory, pp. 92-100. ACM, 1998.

[96] Yarowsky, David. "Unsupervised word sense disambiguation rivaling supervised methods." In Proceedings of the 33rd annual meeting on Association for Computational Linguistics, pp. 189-196. Association for Computational Linguistics, 1995.

[97] Dópido, Inmaculada, Jun Li, Prashanth R. Marpu, Antonio Plaza, Jose M. Bioucas Dias, and Jon Atli Benediktsson. "Semisupervised self-learning for hyperspectral image classification." Geoscience and Remote Sensing, IEEE Transactions on 51, no. 7 (2013): 4032-4044.

[98] Kohonen, Teuvo. "An introduction to neural computing." Neural networks 1, no. 1 (1988): 3-16.

[99] Nova, David, and Pablo A. Estévez. "A review of learning vector quantization classifiers." Neural Computing and Applications 25, no. 3-4 (2014): 511-524.

[100] Hammer, Barbara, Marc Strickert, and Thomas Villmann. "Relevance lvq versus svm." In Artificial Intelligence and Soft Computing-ICAISC 2004,

pp. 592-597. Springer Berlin Heidelberg, 2004.

[101] Schneider, Petra, Michael Biehl, and Barbara Hammer. "Adaptive relevance matrices in learning vector quantization." Neural Computation 21, no. 12 (2009): 3532-3561.

[102] Gisbrecht, Andrej, Bassam Mokbel, Frank-Michael Schleif, Xibin Zhu, and Barbara Hammer. "Linear time relational prototype based learning." International journal of neural systems 22, no. 05 (2012): 1250021.

[103] Qin, A. Kai, P. Suganthan, and Jing J. Liang. "A new generalized lvq algorithm via harmonic to minimum distance measure transition." In Systems, Man and Cybernetics, 2004 IEEE International Conference on, vol. 5, pp. 4821-4825. IEEE, 2004.

[104] Hammer, Barbara, Marc Strickert, and Thomas Villmann. "Relevance lvq versus svm." In Artificial Intelligence and Soft Computing-ICAISC 2004, pp. 592-597. Springer Berlin Heidelberg, 2004.

[105] Schneider, Petra, Michael Biehl, and Barbara Hammer. "Distance learning in discriminative vector quantization." Neural computation 21, no. 10 (2009): 2942-2969.

[106] Torkkola, Kari, and William M. Campbell. "Mutual information in learning feature transformations." In ICML, pp. 1015-1022. 2000.

[107] Villmann, Thomas, and Sven Haase. "Divergence-based vector quantization." Neural Computation 23, no. 5 (2011): 1343-1392.

[108] Qin, A. Kai, and Ponnuthurai N. Suganthan. "A Novel Kernel Prototype-Based Learning Algorithm." In ICPR (4), pp. 621-624. 2004.

[109] Schleif, F-M., Thomas Villmann, Barbara Hammer, and Petra Schneider. "Efficient kernelized prototype based classification." International journal of neural systems 21, no. 06 (2011): 443-457.

[110] Seo, Sambu, and Klaus Obermayer. "Soft learning vector quantization." Neural computation 15, no. 7 (2003): 1589-1604.

[111] Hofmann, Daniela, Andrej Gisbrecht, and Barbara Hammer. "Efficient approximations of kernel robust soft lvq." In Advances in Self-Organizing Maps, pp. 183-192. Springer Berlin Heidelberg, 2013.

[112] Bache, Kevin, and Moshe Lichman. "UCI machine learning reposi-

tory." (2013).

[113] Haberman, Shelby J. "Generalized residuals for log-linear models." In Proceedings of the 9th international biometrics conference, pp. 104-122. 1976.

[114] Street, W. Nick, William H. Wolberg, and Olvi L. Mangasarian. "Nuclear feature extraction for breast tumor diagnosis." In IS&T/SPIE's Symposium on Electronic Imaging: Science and Technology, pp. 861-870. International Society for Optics and Photonics, 1993.

[115] Horton, Paul, and Kenta Nakai. "A probabilistic classification system for predicting the cellular localization sites of proteins." In Ismb, vol. 4, pp. 109-115. 1996.

[116] https://github.com/tmadl/semisup-learn

[117] Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel et al. "Scikit-learn: Machine learning in Python." The Journal of Machine Learning Research 12 (2011): 2825-2830.

[118] Zhu, Xibin, Frank-Michael Schleif, and Barbara Hammer. "Secure semi-supervised vector quantization for dissimilarity data." In Advances in Computational Intelligence, pp. 347-356. Springer Berlin Heidelberg, 2013.

[119] Schleif, Frank-Michael, Xibin Zhu, and Barbara Hammer. "Sparse conformal prediction for dissimilarity data." Annals of Mathematics and Artificial Intelligence 74, no. 1-2 (2015): 95-116.

[120] Schleif, Frank-Michael, Xibin Zhu, and Barbara Hammer. "Sparse conformal prediction for dissimilarity data." Annals of Mathematics and Artificial Intelligence 74, no. 1-2 (2015): 95-116.