UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

MERIT European Master of Research on Information and Communication Technologies

Analysis of Similarity based Representations on Heterogeneous Data

Master degree of

European Master of Research on Information and Communication Technologies

Universitat Politècnica de Catalunya

Author:	**Shreyas Seshadri**
Thesis Director:	**Lluis Belanche**
	Professor of Faculty of Computer Science, UPC
Assistant Thesis Director:	**Ferran Casadevall**
	Professor of Department of Signal Theory and Communications, UPC

June, 2014

Abstract

The most common methodology of similarity based learning is the k-nearest neighbour method. But this has the disadvantage of having to store and calculate distances (similarities) for all the instances in the dataset. A more efficient methodology is to calculate similarities to only a set of chosen prototypes and use this as a new representation for learning.

The current thesis deals with the implementation of these ideas using the Gower's similarity measure for heterogeneous data. The comparison of clustering and feature selection methods for prototype selection is explored. Different methodologies are implemented in an attempt to improve the Gower's similarity measure with the incorporation of weights for features. Novel methodologies to extract deep/higher level features from the similarity representation are proposed. The thesis provides preliminary results in these areas of research which are encouraging.

Contents

1 Introduction

Machine learning techniques aim to learn the underlying model that governs different aspects of the world, based on patterns extracted from observations. The methodology of extracting this knowledge is of crucial importance to machine learning [28, 29]. Traditional methods build models based on measurable heuristic properties of the real world called features. But there is a new trend in this field to learn from the notion of proximity rather than the features themselves [27, 30, 26]. It is the proximity of instances to each other that determines its description of belonging to a particular class or associating it to a particular target. Instances that are similar/close to each other (in some sense) will have similar targets, and those that are dissimilar tend to have dissimilar targets. This implies that the notion of proximity is more fundamental than the notion of a feature for the purpose of learning [31, 27]. Learners should hence be geared to learn from similarities rather than from features.

The most common methodology of learning from similarities is the k-nearest neighbour (KNN) approach. Here, the majority vote of the k nearest neighbours is chosen to find the class. It is described briefly with an example of classification as shown in figure 1. This shows a two class problem where an instance has to be classified. The value of k has to be tuned. This same mechanism is used for regression, but now the average value of the targets of the k closest neighbours is considered.

The problem with KNN and related algorithms is that the entire dataset is used each time a new instance has to be classified, which greatly increases the computational load and storage requirements. This is especially true for large datasets which are very common these days in online applications. Another disadvantage is that these methods only take into account the information from the closest neighbours and neglect the overall topological structure of the data.

The current thesis deals with the study of similarity based representations, which are an extension to nearest neighbour methods. Recent trends [27, 26] have been focussed

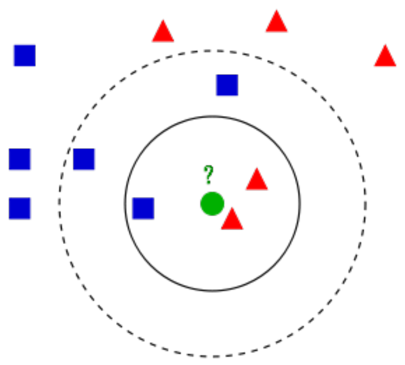


Figure 1: KNN - Example

The figure shows a simple two class problem. The instance at the centre(circle) is the one to be classified. If k is 3, then it is classified as triangle and if k is 5 it is classified as square

on moving to a new representation of data based on similarities or closeness. Here the similarity of instances to a chosen set of prototypes is considered as the new similarity space. In this new space a typical learner can be used to solve a classification or regression problem. The current work seeks to improve this methodology specifically for the problem of heterogenous data.

Heterogeneous data refers to data containing a mix of real, ordinal, categorical etc. data, possibly with missing values. Such data is very common in the real world. For instance, in the well known UCI machine learning repository [32] over half of the datasets contain categorical values [26]. Many times the type of data is not even reported. The problem is that typical learners are designed to work with real data. It becomes especially difficult when datasets contain a mixture of different types of data, which is typically the case. The thesis proposes to implement the Gower's similarity measure to calculate similarities. This representation is able to calculate similarities directly without having to guess the missing values or convert the data to a real representation.

The thesis explores the comparison of clustering and feature selection methods for the task of prototype selection within the framework of similarity representations. Different methodologies are also explored in an attempt to improve the Gower's similarity measure with the incorporation of weights for features. Novel methodologies to extract deep/higher level features from the similarity representation are proposed. The thesis provides preliminary results which are encouraging. Overall the thesis is an initial attempt at these methods. Further study and experimentation is required to create accurate methodologies from these ideas.

Section 2 discusses the related prior research in the field of similarity representations in the context of this thesis. Section 3 discusses the different datasets used in this study for the purpose of experiments. Sections 5, 6 and 7 discuss the proposed mechanisms of prototype selection, improved similarity measure and higher level similarities respectively.

2 Related prior art: Similarity Learning

The intuitive notion of *similarity* is very useful to group objects under specific criteria and has been used with great success in several fields within or related to Artificial Intelligence, like Case Based Reasoning [1], Information Retrieval [2] or Pattern Matching [3].

In classical artificial intelligence (AI) systems, the notion of *similarity* appears mainly in case-based reasoning (CBR), where learning by analogy and instance-based learning fuse. The popularity of CBR systems has boosted its use and signalled its key role in cognitive tasks [33]. As a matter of fact, some form of similarity is inherent in the majority of AI approaches. There is an agreement in that it is easier to respond intelligently to a stimulus if previous responses made under *similar* circumstances can be recalled. Similarity coefficients have had a long and successful history in the literature of cluster analysis and data clustering algorithms [15].

The origins of learning by similarity in artificial neural systems can be traced back to the pioneering works of Hebb and his now classic book [34]. He postulated that the functionality of ANNs had to be determined by the strengths of the neural connections. This functionality should be adjusted to *increase* the likeliness of getting a *similar* response to *similar* inputs in the future, provided the elicited response is the desired one. In the opposite situation, the weights should be adjusted to decrease this likeliness. However, this inspiring idea has not been fully exploited in the prevalent neuron models.

There have been few attempts to incorporate heterogeneous information into the workings of an artificial neuron in a principled way. The main contribution is perhaps encountered in heterogeneous distance proposals, where separate distance calculations are used for nominal and continuous variables [35], where the authors present an extension of the RBF model to nominal quantities and missing values. The RBF network is used in its original interpolative definition (i.e., there is one hidden node for each example in the training set). The extension consists in the use of the Value Difference Metric [36] for nominal distance computation. An Euclidean metric is used to account for the partial distances. Missing values are handled with a *pessimistic* semantics, defining the distance involving a missing value to be the maximum possible distance (actually a value of one). The results point to an increase in performance w.r.t. standard Euclidean distance when the amount of nominal information is significant, giving support to a deeper and more precise formulation of neuron models as similarity computing devices.

Section 2.1 introduces the problem of heterogeneous data and section 2.2 details the Gower's similarity measure. The intuition of similarity is extended to neural networks in 2.5. Finally 2.6 introduces the notion of a similarity space, using the above methods as pre-processing step.

2.1 Data types and missingness

In many important domains from the real world, objects are described by a mixture of continuous and discrete variables, usually containing missing information and characterized by an underlying uncertainty or imprecision. For example, in the well-known UCI repository [9] over half of the problems contain explicitly declared categorical attributes, let alone other data types, usually unreported. In the case of artificial neural networks (ANN), this *heterogeneous* information has to be encoded in the form of real-valued quantities, although in most cases there is enough domain knowledge to characterize the nature of the variables.

The integration of heterogeneous data sources in information processing systems has been advocated elsewhere [10]. In this sense, a shortcoming of the existent neuron models is the difficulty of adding prior knowledge to the model in a principled way. Current practice assumes that input vectors may be faithfully represented as a point in $\mathbb{R}^n$, and the geometry of this space is meant to capture the meaningful relations in input space. There is no particular reason why this should be the case. Moreover, the activity of the units should have a well defined meaning in terms of the input patterns [12]. Without the aim of being exhaustive, commonly used coding methods are (see, e.g. [13]):

Ordinal variables coded as real-valued or using a *thermometer* scale.

Categorical variables with c modalities are coded using a binary expansion representation (also known as a 1-out-of-c code).

Vagueness and uncertainty are considerations usually put aside.

Missing information is difficult to handle, specially when the lost parts are of significant size. Typical approaches remove the involved examples (or variables) or "fill in the holes" with the mean, median or nearest neighbor value. Statistical approaches need to model the input distribution itself [11], or are computationally very intensive [16].

Although these encodings may be intuitive, their precise effect on performance (very specially in relation to overfitting) is far from clear. This is due to the change in input distribution, the increase (sometimes acute) in input dimension and other subtler effects, derived from imposing an order or a continuum where there was none.

2.2 Similarity measures

Let us represent patterns belonging to a space $X \neq \emptyset$ as a vector x of n components, where each component x_k represents the value of a particular feature k. A *similarity measure* is a unique number expressing how "like" two patterns are, given these features. It can be defined as an upper bounded, exhaustive and total function $s : X \times X \to I_s \subset R$ with $|I_s| > 1$ (therefore I_s is upper bounded and $s_{max} \equiv \sup_{\mathbb{R}} I_s$ exists). A similarity measure may fulfill many properties, like:

Reflexivity: $s(x, y) = s_{max} \Leftrightarrow x = y$.

Symmetry: $s(x, y) = s(y, x)$.

Lower boundedness: $\exists a \in R$ such that $s(x, y) \geq a$, for all $x, y \in X$ (note this is equivalent to ask that $\inf{}_{\mathbb{R}} I_s$ exists).

Closedness: given a lower bounded s, $\exists x, y \in X$ such that $s(x, y) = \inf{}_{\mathbb{R}} I_s$ (equivalent to ask that $\inf{}_{\mathbb{R}} I_s \in I_s$).

These axioms should be taken as *desiderata*. Some similarity relations may fulfill part or all of them [15]. Other properties (like transitivity) may be of great interest in some contexts, but are not relevant for this work. However, it is not difficult to show that reflexivity implies a basic form of transitivity [6].

2.3 Similarity measures for different variable types

We present in this section specific similarity measures defined in a common codomain $I_s = [0, 1]$. Not only it is possible to find different types of variables, also different similarity measures could be used for different variables of the same type. For notational convenience, we use s_{ijk} to mean $s_k(x_{ik}, x_{jk})$.

2.3.1 Nominal variables

It is assumed that no partial order exists among these values and the only possible comparison is equality. The basic similarity measure for these variables is the *overlap*. Let x_{ik}, x_{jk} be the modalities taken by two examples x_i, x_j, then $s_{ijk} = 1$ if $x_{ik} = x_{jk}$ and 0 otherwise.

2.3.2 Ordinal variables

These variables can be seen as a bridge between categorical and continuous variables. It is assumed that the values of the variable form a linearly ordered space $(\mathcal{O}, \preceq)$. Let $x_{ik}, x_{jk} \in \mathcal{O}$, such that $x_{ik} \preceq x_{jk}$, and P_{lk} be defined as above. Then,

$$s_{ijk} = \frac{2 \log(P_{ik} + \ldots + P_{jk})}{\log P_{ik} + \log P_{jk}} \tag{1}$$

where the summation runs through all the ordinal values x_{lk} such that $x_{ik} \preceq x_{lk}$ and $x_{lk} \preceq x_{jk}$ [8].

2.3.3 Continuous variables

Let $x_{ik}, x_{jk} \in A = [r^-, r^+] \subset \mathbb{R}, r^+ > r^-$. The standard metric in $\mathbb{R}$ is a metric in A. Therefore, for any two values $x_{ik}, x_{jk} \in A$:

$$s_{ijk} = \hat{s}\left(\frac{|x_{ik} - x_{jk}|}{r^+ - r^-}\right) \tag{2}$$

where $\hat{s} : [0,1] \longrightarrow [0,1]$ is a decreasing continuous function. A very simple family is $\hat{s}(z) = (1 - z^\beta)^\alpha, 0 < \beta \leq 1, \alpha \geq 1$. We use here the simplest choice $\alpha = \beta = 1$.

2.4 Missing value treatment

Missing information is a recurrent problem in data analysis because there are many causes for the absence of a value. The problem acquires more relevance when significant parts of a data sample are lost or unknown. There are basically three ways of dealing with *missing values*: fill in the examples, extend the learning methods to cope with incomplete data or discard the examples (or the variables) with missing values. We advocate for the second possibility, for which there exist some possible approaches:

1. The first proposal is based on Gower's general similarity measure [17]. When a partial similarity s_i is *missing*, the *flag* δ_i that goes with this partial similarity is set to 0 (otherwise it is 1).

   The work of Gower in general similarity measures [17] shows some partial coefficients of similarity for three different types of features: Dichotomous (Discrete or Categoric), Qualitative (Categoric) and Quantitative (Continuous and Discrete) features, that in addition are shown to produce PSD (similarity) matrices. These functions can therefore be seen as Kernels.

   For any two points $x_i, x_j \in X$ to be compared on the basis of a feature k a score

s_{ijk} is defined, described below. First δ_{ijk} is defined as 0 when the comparison of x_i, x_j cannot be performed on the basis of feature k for some reason; for example, by the presence of missing values, by the feature semantics, etc; δ_{ijk} is 1 when such comparison is meaningful. The coefficient of similarity between x_i, x_j is defined as the average score over all the partial comparisons:

$$S_{ij} = \frac{\sum_{k=1}^{n} s_{ijk}}{\sum_{k=1}^{n} \delta_{ijk}} \tag{3}$$

If $\delta_{ijk} = 0$ for all the features, then S_{ij} is not defined. An equivalent form to (3) is

$$S_{ij} = \frac{\sum_{k=1}^{n} s_{ijk}\delta_{ijk}}{\sum_{k=1}^{n} \delta_{ijk}}$$

The scores s_{ijk} are defined as follows:

i) *For Dichotomous (binary) features*: The presence of the feature is denoted by $+$ and its absence by $-$. When there are no missing values of feature k, then

	Values of feature k			
Point x_i	+	+	-	-
Point x_j	+	-	+	-
s_{ijk}	1	0	0	0
δ_{ijk}	1	1	1	0

ii) *For Qualitative features*: Define

$$s_{ijk} = \begin{cases} 1, & \text{if } x_{ik} = x_{jk}; \\ 0, & \text{if } x_{ik} \neq x_{jk} \end{cases}$$

iii) *For Quantitative features*: With values $x_1, x_2, ..., x_n$ of featrue k for the total sample of n points define

$$s_{ijk} = 1 - \frac{|x_{ik} - x_{jk}|}{R_k}$$

where R_k is the *range* of feature k (the difference between the maximum and minimum values); it may be the population range or the range in the sample (the training set, in practice).

Gower proves that, if there are no missing values, the similarity matrix $S = (S_{ij})$ is PSD. This property may be lost when there are missing values. An example will suffice:

We have three points with four quantitative features:

Feature	1	2	3	4
Point 1	1	2	3	1
Point 2	1	3	3	*
Point 3	1	3	3	5

In this table $*$ denotes a missing value and all feature values lie in $[1, 5]$, that is, $R_k = 4$. In this case,

$$S = \begin{pmatrix} 1 & \frac{11}{12} & \frac{11}{16} \\ \frac{11}{12} & 1 & 1 \\ \frac{11}{16} & 1 & 1 \end{pmatrix}$$

then

$$\det(S) = -\frac{121}{2304}$$

and S is therefore not PSD. But if we replace $*$ by *any* value in $[1, 5]$, then the matrix S is certainly PSD.

It is not difficult to realize that this is equivalent to the replacement of the missing similarities by the average of the non-missing ones. Therefore, the conjecture is that the missing values, if known, would not change the overall similarity.

2. The second proposal is even simpler: to replace the missing partial similarity measures by a constant quantity, namely the midpoint of the similarity codomain

I_s. For instance, when $I_s = [0, 1]$, this constant would be $\frac{1}{2}$. Doing this we are assuming that the missing similarities, if known, would make the example as similar to any other example as the average similarity.

Both methods look very naive and indeed they are; on the other hand, they are very intuitive and computationally simple. The appealing trait of these two approaches is that they do not try to estimate the missing information (a delicate and risky undertaking) but to estimate the *overall* similarity between two observations, given that some of the partial similarities could not be computed. We argue that this second task is easier and, after all, is what we really are interested in: the *similarity value*. This is the reason why we consider missing value treatment together with the construction of the overall similarity value.

2.5 Extension to Neural Networks

This section deals with the extension of the previously defined similarity representations to a neural network architecture. Section 2.5.1 briery describes artificial neural networks in general. Section 2.5.2 extends the idea of a similarity based neuron and section 2.5.3 extends this idea to a neural network architecture.

2.5.1 Artificial Neural Networks

Artificial Neural Networks (ANN) [37] are information processing structures evolved as an abstraction of known or assumed principles of how the brain might work. The network is said to *learn* when, as a result of exposure to examples, the parameters of the units are adapted to represent the information present in the examples in an optimal sense to be precised. The network relies upon the neuron model representation capacity as the cornerstone for a good approximation.

The strong points of ANNs are their appealing capacity to learn from examples, their distributed computation —which helps them tolerate partial failures to a certain extent— and the possibility, so often exploited, to use them as black-box models. This last characteristic is paradoxically one of the major weaknesses, given that in practice this autonomy of functioning involves no transfer of knowledge from or to the designer. With the exception of very specific architectures, the networks are forced to learn from scratch most of the times. The network *works* (in the sense that solves a problem to a certain satisfaction), but the weights convey information as high-dimensional real vectors whose meaning about the solution can be intricate. A significant part of the task is devoted to find structure in the data and transform it to a new *hidden* space (or space spanned by the hidden units) in such a way that the problem becomes easier (almost linearly separable in the case of the output layer). The internal workings of a neuron are thus obscure, because the weights have not been set to shape a previously defined (and considered adequate) similarity measure, but rather to adapt a general *physical measure* (like scalar product or Euclidean distance) to the problem at hand.

In practice the network has to discover the relations in the structure induced by the chosen coding scheme and find ways to *accommodate* the underlying similarity relationship (inherent in the training examples) to a fixed similarity computation. During the learning process, patterns seen as physically similar may have to be told apart and vice versa. In consequence, several layers may be needed for complex transformations, or a large amount of neurons per layer if we restrict the number of hidden layers to one or two, as is common proceeding. An increase in neurons leads to a corresponding growth in the number of free parameters, and these are less likely to be properly constrained by a limited size data set [37].

Besides all that, real-world data come from many different sources (continuous or discrete numerical processes, symbolic information, etc.) and have their own peculiarities (vagueness, imprecision, incompleteness), and thus may require different treatments. For example, in the well-known UCI repository [9] over half of the problems contain *explicitly declared* nominal attributes, let alone other discrete types or fuzzy information, usually unreported. This heterogeneity is traditionally coped with by *preparing* the data using a number of methods. This preprocessing is not part of the original task and may involve an abrupt change in input dimension and distribution. The new high-dimensional patterns entail a lower data density, requiring stronger constraints on the problem solution [38].

2.5.2 The S-Neuron Model

Consider $s : X^n \times X^n \to I_s$ a similarity function in $X^n = X^{(1)} \times \ldots \times X^{(n)}$, the cartesian product of an arbitrary number n of *source sets*, where $I_s = [0, 1]$ for simplicity. This function is formed by combination of n partial similarities $s_k : X^{(k)} \times X^{(k)} \to I_s$, each $X^{(k)}$ being the domain of the predictive variable k.

The s_k are normalized to a common real interval ($I_s = [0, 1]$ in this case) and computed according to different formulas for different variables (possibly but not necessarily determined by variable type alone). A *neuron model* can be devised that is both similarity-based and handles data heterogeneity and missing values, as follows. Let $\Sigma_i(x)$ the function computed by the i-th neuron, where $x \in \hat{\mathcal{H}}^n$ having a weight vector $\mu_i \in \hat{\mathcal{H}}^n$ and smoothing parameter p_i, defined as:

$$\Sigma_i(x) = f(s(x, \mu_i), p_i), \text{ with } s(x, \mu_i) = \frac{1}{n} \sum_{k=1}^{n} s_k(x_k, \mu_{ik}) \tag{4}$$

This S-neuron adds a non-linear *activation* function to the linearly aggregated similarities. Such function could be any sigmoid-like automorphism (a monotonic bijection) in $[0, 1]$. In particular, we consider the simple family of functions:

$$f(x,p) = \begin{cases} \frac{-p}{(x-0.5)-a(p)} - a(p) & \text{if } x \leq 0.5 \\ \frac{-p}{(x-0.5)+a(p)} + a(p) + 1 & \text{if } x \geq 0.5 \end{cases}$$
$$a(p) = \frac{-0.5 + \sqrt{0.5^2 + 4p}}{2} \tag{5}$$

where $p > 0$ is a parameter controlling the shape of the function (Fig. 2). The function fulfills $\forall p \in \mathbb{R}^+,\ f(0,p) = 0,\ f(1,p) = 1,\ \lim_{p\to\infty} f(x,p) = x$ and $f(x,0) = H(x-0.5)$, being H the Heaviside function.

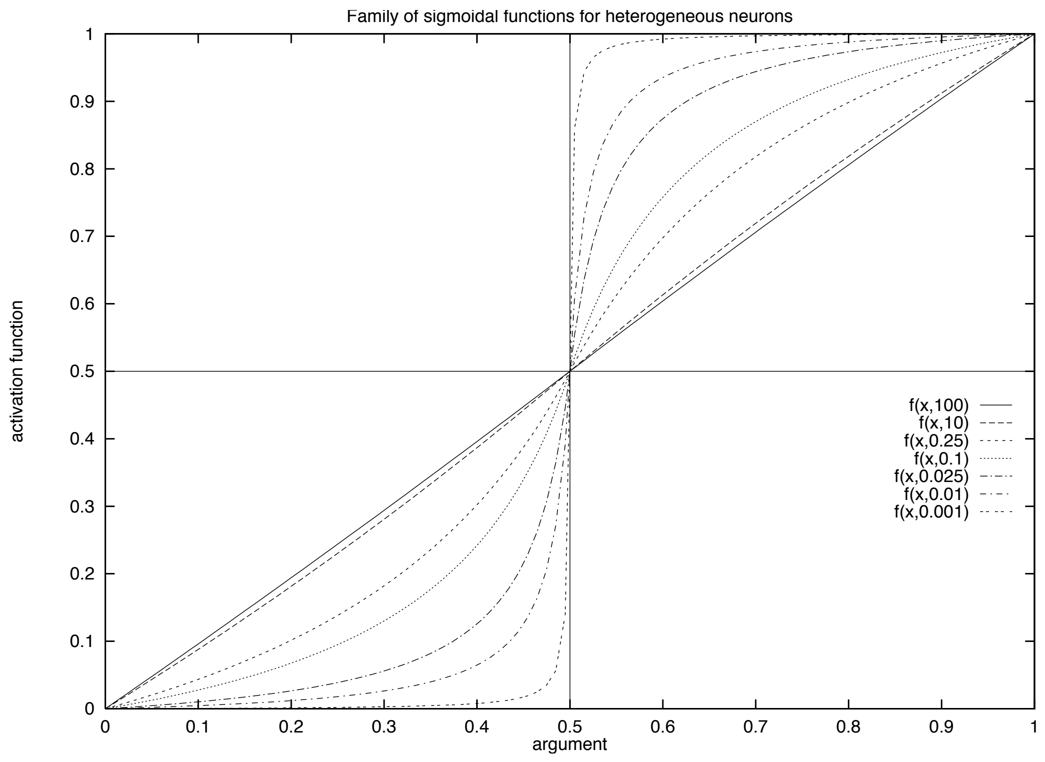


Figure 2: Family of sigmoidal functions $f(\cdot, p)$ for different values of p.

2.5.3 Similarity Neural Networks

Similarity neural networks (SNN) are neural architectures built out of the previously defined S-neurons, thus allowing for heterogeneous or missing inputs. A feed-forward architecture, with a hidden layer composed of heterogeneous neurons and a linear output layer is a straightforward choice, thus conforming a *hybrid* structure. The k-th output neuron of the SNN computes the function:

$$f_k(x) = \sum_{i=1}^{h} w_{ki}\Sigma_i(x) + w_{k0},\ k = 1, \dots, m$$

where $h > 0$ is the number of hidden S-neurons, m is the number of outputs and $\{w_{ki}\}$ is the set of mixing coefficients. The SNN thus keeps linearity in the output layer and interpretability in the hidden layer. It can be naturally seen as a generalization of the RBF. This is so because the response of hidden neurons is localized: centered at a given object (the neuron weight, where response is maximum), falling down as the input is less and less similar to this center.

2.6 Similarity representation

Another approach along these lines could be to incorporate the similarity neural network as a pre-processing step to project the data into a new space called the similarity space [26, 27]. Here the new set of features are similarities to a set of chosen prototypes. Any learner can now be used in the new space for the required task. Figure 3 shows this framework of pre-processing.

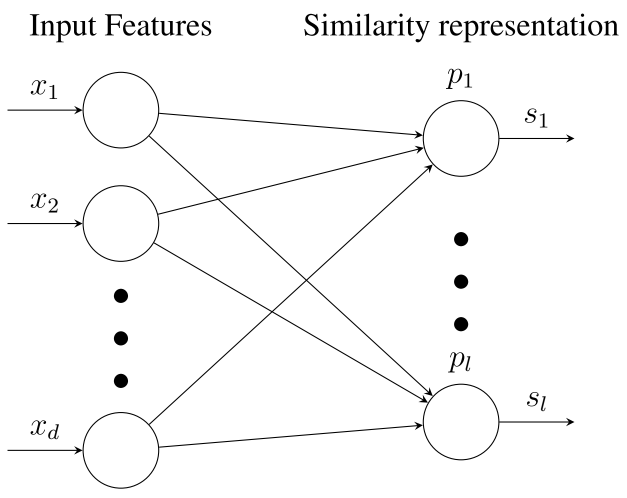


Figure 3: Similarity representation used as pre-processing

The first layer represents the d input features for a single instance in the original data space. The l chosen prototypes selected from the original space are as $p_1 ... p_l$ and $s_1 ... s_l$ represent features in the new space

3 Data

This section deals with the datasets used for study in this thesis. Section 3.1 introduces the 7 datasets used for study and section 3.2 describes the handling of data for the different methodologies tested in this thesis.

3.1 Datasets

Some challenging problems have been selected as characteristic of modern modeling datasets because of the diversity in data heterogeneity and the presence of missing values. The problem descriptions and the datasets are taken from the UCI repository [9]. The available documentation has been analyzed for an assessment on the more appropriate treatment. Missing information is also properly identified – see Table 1. The Horse Colic dataset has been investigated with two different targets (variables #23 and #24, resp.).

Name	#Obs	Def.	Missing	In→Out	Data
Pima Diabetes	768 (500,268)	65.1%	10.6%	8 → 2	8R, 0N, 0D
Horse Colic 23	363 (295,68)	61.4%	25.6%	22 → 3	7R, 7N,8D
Horse Colic 24	364 (296,68)	63.5%	25.6%	22 → 2	7R, 7N,8D
Credit Approval	690 (400,290)	55.5%	0.65%	15 → 2	6R, 9N, 0D
Audiology	226 (200,26)	66.3%	2.1%	31 → 4	0R, 24N, 7D
Heart	270 (200,70)	55.56%	0%	13 → 2	6R, 6N, 1D
Hepatitis	155 (100,55)	79.35%	5.6%	19 → 2	6R, 13N, 0D

Table 1: Basic characteristics of the datasets

#Obs (learning, test). Def. (default accuracy), Missing (percentage of missing values). In→Out (no. of inputs features and output classes). The last column shows variable types: (R)eal, (N)ominal, or(D)inal.

3.1.1 Pima Diabetes

This is a much studied dataset, in which a population of Pima Indian women living near Phoenix, Arizona was tested for diabetes according to World Health Organization criteria. In this dataset, most of the variables show impossible zero values (e.g, Diastolic blood pressure), which are actually missing values [39]. Upon careful analysis, it turns out that only 392 out of the 768 observations are unaffected.

3.1.2 Horse Colic

This dataset makes an excellent case study, because of the diversity in data heterogeneity and the presence of missing values; it has been used as paradigmatic example in some textbooks [19]. Each observation is the clinical record of a horse and the variables are specially well documented[1].

3.1.3 Audiology Database

This problem is interesting for many reasons: it is multiclass, has a low number of observations and all variables are categorical (with different modalities, some of them ordered)[2]. We have reduced the original 24 classes to 4 by a meaningful grouping. We have also eliminated non-informative variables.

3.1.4 Credit Approval

This is a binary classification problem concerning credit card applications. This dataset is interesting because there is a good mix of attributes – continuous, nominal with small numbers of values, and nominal with larger numbers of values. There are also a few missing values.

3.1.5 Hepatitis

This dataset [3] relates to predicting whether the patients live or die after being diagnosed with Hepatits. It contains real and nominal values as features.

[1]This dataset is made available thanks to M. McLeish and M. Cecile (Computer Science Dept., Univ. of Guelph, Ontario, Canada).

[2]Original owner: Professor Jergen at Baylor College of Medicine.

[3]Original donor: G.Gong (Carnegie-Mellon University) via Bojan Cestnik

3.1.6 Heart

This is another binary classification that deals with data collected from patients aiming to predict the presence or absence of heart disease. It contains no missing values but provides a good mix of different data types. The classes are evenly distributed with accuracy close to 50%.

3.2 Data Handling

The datasets are handled in the two way discussed

1. ***raw :-*** There is no effort in identifying variable types (all information is considered numerical, and scaled); missing values are either not identified or left as they come (for example, treated as zeros).
2. ***heterog :-*** The data is retained in the original format. The missing values are identified. This mechanism is used for the Gower's similarity formulation.

4 Simple KNN and Gower KNN

The K nearest neighbour technique discussed earlier is now compared for the different datasets using the traditional Euclidean distance and the Gower's similarity measure.

4.1 Experimental Setup

The analysis is done in the two ways discussed below

1. Using the *'raw'* version of the data (as described in section 3.2). The distance/similarity measure used here is the Euclidean distance.
2. Using the *'heterog'* version of the data (as described in section 3.2). The distance/similarity measure used here is the Gower's similarity measure 2.4.

The data is split to test and training sets as described in table 1. The number of nearest neighbours considered for KNN is tuned between 1 and 15. This is done with K-fold cross validation with 5 folds on the training set. This process is repeated for 20 different combinations of training and test sets and the average classification error is studied.

4.2 Results and Inference

Figure 4 shows the average test error for both versions of KNN. As can be seen there is not much difference between the two methodologies. More sophisticated methods have to be implemented in order to get significantly better results using the Gower's similarity measure. Table 2 shows the average test error over all the seven datasets being considered.

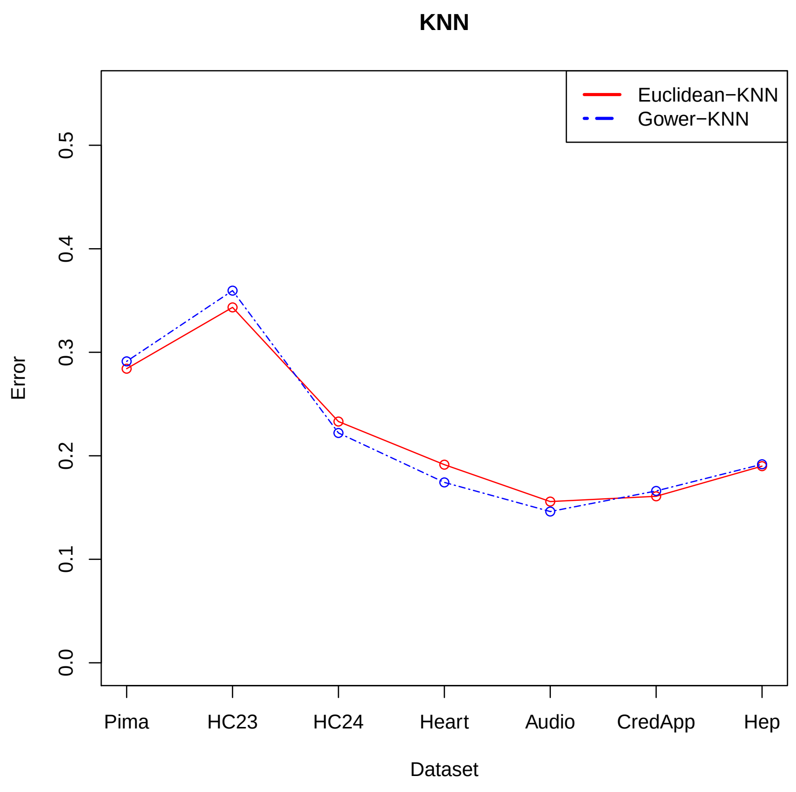


Figure 4: K-Nearest Neighbour results

Average test error values for all datasets considered for simple KNN method shown in red and KNN with Gower's measure in blue

Method	**Average error(%)**
Euclidean-KNN	22.2
Gower-KNN	22.1

Table 2: Average Error over all Datasets

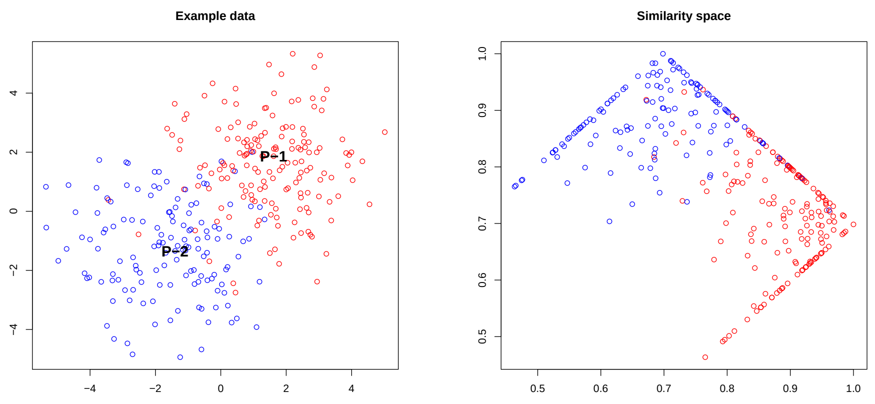
5 Prototype Selection

The main idea with the advent of similarity (or dis-similarity) based methods is to create a more efficient method than nearest neighbour methods. The problem with nearest neighbour methods is that similarities between all the instances in the entire training dataset and the current instance to be classified, need to be measured. This can be very computationally intensive for large datasets. Better techniques use clustering to reduce the dataset to a certain number of centroids and use nearest neighbour on the centroids. But these methods suffer from rejecting all instances other than the chosen centroids.

Recent techniques developed by [27, 26] use the similarity representation as data to be learned using a typical learner. In this methodology, the similarity between each instance in the training data and the set of centroids(called prototypes) is calculated. Hence each instance now has instead of its original features, features which are similarities to each of the chosen prototypes. Data in this new 'similarity' feature space can now be learned by a typical learner. The advantage here over, typical nearest neighbour methods is that, the entire dataset and not just the prototypes contribute to the results, while still having to calculate only the similarity between the new instance to be solved, with the set of prototypes.

For example, consider a dataset with originally N instances and d features. Clustering is done to choose d_1 prototypes from the data space. The new similarity space is represented by an $N \times d_1$ matrix i.e. with N instances and d_1 features corresponding to the similarity between each instance and the chosen prototypes. A typical learner is now used for the required task of classification or regression .

This methodology is illustrated in the following example shown in figure 5. This shows a simple 2 dimensional classification problem. Prototypes are chosen using the partitioning around medoids (PAM) algorithm discussed in algorithm 1.



(a) *2-D example* (b) *Similarity representation*

Figure 5: Similarity Representation - Example

Figure 5(a) shows a simple 2-D problem with two classes in red and blue. The prototype chosen are marked as 'P-1' and 'P-2'. Figure 5(b) shows the similarut representation using the two prototypes.

All studies done so far on such similarity based methodologies use clustering in the original data space(to choose prototypes) to reduce the final similarity matrix. Another possibility that is not yet explored, is using feature selection methods. Here the similarity data space created from the similarity of each instance with all the other points in the space. Hence for a dataset with originally N instances and d features, the similarity data space is now an $N \times N$ matrix i.e. with N instances and N features corresponding to the similarity between each instance and every other instance. Feature Selection methods could now be used on this dataset and may lead to better results.

The subsequent sections will describe the methodology of the theory, the experimental setup and results obtained.

5.1 Methodology

The following approaches have been followed to compare the two approaches for prototype selection i.e. Clustering and Feature Selection. Each of these approaches results in the similarity space representation of an $N \times d_1$ matrix i.e. with N instances and d_1 features. This is treated as separate dataset and is trained with a typical learner. Finally section 5.1.3 discussed the different resampling methods used for experimentation.

5.1.1 Clustering

Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups (clusters). The notion of similarity can be different leading to different algorithms. It is the task of dividing the data space into definite zones and the set of instances contained in each zone is projected onto a representative instance called the centroid. An example is shown in the figure 6

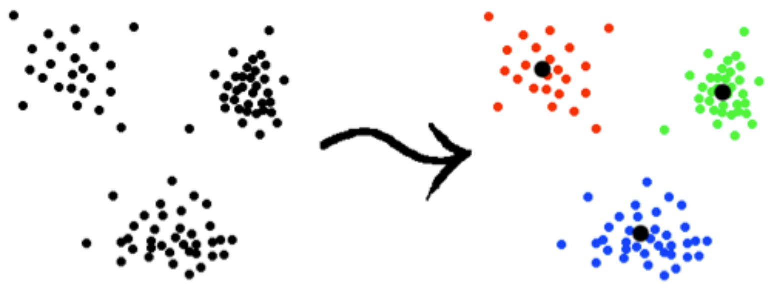


Figure 6: Clustering Example

The data space on the left is divided into 3 zones/clusters shown as different colours on the right. Each cluster is represented by an instance called as a centroid.

PAM for Clustering in the original data space :- Partitioning around medoids (PAM) [41] algorithm has the advantage of choosing medoids(centroids) that are part of the dataset. A brief description of the PAM algorithm is shown below in 1. It is also called as the k-medoids algorithm. The total cost is the sum of distance/dis-similarity between each instance and its closest the medoid (prototype). The Manhattan distance is used as a measure of distance/dis-similarity here. The number of medoids d_1 has to be optimised.

Algorithm 1: Partition around Medoids

```
Data: $X_{N \times d}$ - Input data of $N$ instances and $d$ features
Result: $P_{d_1 \times d}$ - $d_1$ instances chosen as medoids
1  Initialize :- $P_{d_1 \times d}$ randomly from $X_{N \times d}$ ;
2  while $P_{d_1 \times d}$ changes do
3      Associate each instance to the closest medoid forming clusters;
4      for $i$ := all medoids do
5          for $k$ := all instances do
6              Swap $i$ and $k$ and re-compute total cost of the configuration;
7          end
8      end
9      Choose the set of medoids that corresponds to minimum total cost
10 end
```

This is the typical approach, where clustering is is done on the original data space to choose the prototypes. PAM offers reliable and relatively stable results.

PAM for Clustering in the similarity space :- The $N \times N$ matrix that represents the data in the similarity space is symmetric. This means that another approach that could be explored other than the two discussed above could be explored. Clustering could be done in the similarity space to choose prototypes. Since this $N \times N$ matrix is symmetric, the indices of the prototypes chosen can be used to choose the relevant features.

It will be interesting to observe the results of this approach. Although it is true that the matrix is symmetric, the methodology of clustering and feature selection are completely different. Clustering tries to best choose few instances in the space that are the best approximation of the entire dataset. Feature selection on the other hand is the problem of finding the set of features that add the most information to the task of learning.

5.1.2 The Feature Selection algorithm

Feature selection can be seen as a search problem, where each state in the search space corresponds to a subset of the features. In the ML literature, a wide family of suboptimal algorithms depart from an initial solution and iteratively add or delete features by locally optimizing the error function. In *forward selection*, features are progressively incorporated into larger subsets; in *backward selection* (or elimination) one starts with the full set of features and progressively eliminates elements from it.

Wrappers are often criticized because they are computationally very expensive. Moreover, feature selection is badly affected by small sample sizes, producing overly optimistic results and introducing an excess of variance in the readings. This is aggravated in the presence of very sophisticated search algorithms [44]. On the other hand, greedy search strategies seem to be particularly computationally advantageous and may alleviate the problem of overfitting [45]. Nevertheless, traditional pure forward selection and backward elimination search algorithms are ill-advised in that they cannot rectify their decisions and may end up delivering poor solutions both in terms of quality and size.

ReliefF Algorithm for Feature selection :- The Relief algorithm [42] is a feature selection algorithm designed for a binary classification problem. It is one of the simplest feature selection algorithms. The algorithm returns a set of weights corresponding to the relevance of each feature for the problem of classification. The features with the highest weights are chosen. The basic algorithm of Relief is shown in 2. The nearest hit H_i and nearest miss M_i of a particular instance x_i refer to the closest instances of the same class and of the opposite class respectively.

Algorithm 2: Relief

```
Data: $X_{N \times d}$ - Data of $N$ instances and $d$ features
Result: $w$ - array of weights for each feature
1 $w[:]$ := 0;
2 for i := 1 to $d$ do
3     $x_i$ := random instance ;
4     $H_i$ := nearest-hit of $x_i$;
5     $M_i$ := nearest-miss of $x_i$;
6     for k := 1 to $N$ do
7         $w_k := w_k + \frac{|M_{ik} - X_{ik}|}{N} - \frac{|H_{ik} - X_{ik}|}{N}$
8     end
9 end
```

This algorithm is used for feature section in the $N \times N$ similarity matrix, as discussed above. An improvement of the Relief algorithm called the ReliefF algorithm is used here. As before, the number of features chosen, d_1, has to be optimised.

CFS for Feature selection :- The CFS [43] is a correlation based feature selector. It is based on finding the subset of features S that maximises the correlation between the features and the target and minimises the correlation between the features. This is based on the concept that good features are those which have a high linear relationship to the target but at unrelated to each other. For a dataset of d features, CFS finds the subset of k features C_k that maximises the value shown in 6. $f_1 ... f_k$ are the k chosen features, t is the target and r denotes a measure of correlation.

$$\underset{S_k}{\text{maximize}} \quad \frac{r_{tf_1} + r_{tf_2} ... + r_{tf_k}}{\sqrt{k + 2(r_{f_1 f_2} + ... + r_{f_i f_j} ... + r_{f_k f_1})}} \tag{6}$$

For this framework, the CFS algorithm is implemented in the $N \times N$ similarity matrix to extract the relevant prototypes. There is no need of tuning the model, as in the case of ReliefF, as CFS directly returns the optimal feature set.

5.1.3 Resampling methods

Model selection is concerned with the process of finding the optimal model for a set of samples among a set of candidate models. Resampling methods aim at making a better use of the available data. These methods are very useful for assessing how a predictive model that can be the result of a complex modeling process will perform in practice.

The generic goal of cross-validation (CV) is to estimate the expected error of a model in a data set that is independent of the data that were used to train the model. One round of k-fold CV (or k-CV) involves partitioning the sample into k complementary subsets, systematically performing the modeling on the union of $k-1$ such subsets and checking the obtained model on the remaining subset (acting as a validation set). The result of k-CV is an estimation of the error if only a fraction $(k-1)/k$ of the available data is used. This error is expected to be conservative (larger than the error obtained if the entire sample was used). To reduce variability, multiple rounds can be performed using different partitions, and the results averaged over the rounds.

5.2 Experimental Setup

The *'heterog'* version of the datasets are used i.e. containing the nominal, ordinal and missing values in the original form. The data is split to test and training as described in table 1. Let N_{tr} be the number of instances in the training set and N_{te} be the number of instances in the test set. The $N_{tr} \times N_{tr}$ similarity matrix is calculated using the Gower's similarity measure. d_1 features from this matrix are selected by using the above mentioned methods of clustering or feature selection. For the methods such as PAM and RELIEFF, the number of features/prototypes chosen must be tuned. This is done using K-fold cross validation, for each learner(discussed below) individually, by varying the number of features/prototypes chosen, d_1, from 2% to 10% of N_{tr}. The learner is then finally trained using d_1 corresponding to the minimum classification error in cross validation.

For testing the Gower similarity between each of the test instances and the chosen prototypes/features is calculated to form a $N_{te} \times d_1$ dataset and fed to the trained learner. This process is repeated for 20 different combinations of training and test sets and the average classification error is studied.

This approach is initially compared by applying KNN to the new similarity data-space. Other learners chosen for the analysis in this study are

1. Logistic regression
2. Linear Discriminant Analysis
3. Support Vector Machines
4. Multinomial regression

The following will briefly describe each of these algorithms

5.2.1 Logistic regression

Logistic regression is a binary classification algorithm. It seeks to solve the equation 7, where Y_i is the target,X_i the ith instance of the data, w a set of weights for each feature and function $g()$ is called the *logit* function.

$$g(\mathbb{E}\{Y_i|X_i\}) = w^T X_i \tag{7}$$

It is an extension of the generalised linear model for a Bernoulli distribution. Here the *logit* function is defined as $g(z) = ln(\frac{z}{1-z})$. Hence equation 7 can be rewritten as equation 8 .

$$\mathbb{E}\{Y_i|X_i\} = \frac{1}{1 + e^{-w^T X_i}} \tag{8}$$

The final probability distribution of Y_i belonging to a particular class can be written as equation 9. The class with the highest probability is assigned the ith instance.

$$P(Y_i = y_i|X_i) = (\frac{1}{1+e^{w^T X_i}})^{y_i}(1 - \frac{1}{1+e^{w^T X_i}})^{1-y_i} \tag{9}$$

5.2.2 Linear Discriminant Analysis

Linear and quadratic discriminant analyses or LDA/QDA (Duda et al. 2001) are widely used parametric methods which assume that the class distributions are multivariate Gaussians. With LDA, all classes are assumed to have the same covariance matrix. QDA does not need such an assumption; however, the number of parameters to be estimated from the data available for each class is much higher, entailing lower statistical significance.

In both methods, classification is achieved by assigning an example to the class for which the posterior probability $P(\omega_k|\mathbf{x})$ is greater, or equivalently for which $\ln\{P(\omega_k)p(\mathbf{x}|\omega_k)\}$ is greater.

These methods are attractive because they need no parameter tuning, and their limited complexity (quadratic at most) may be a solid guard against overfitting the data. Moreover, for LDA fast updating procedures exist for the computation of certain forms of the cross-validation error [47]. The *discriminant function* for class ω_k is expressed as:

$$\begin{aligned} g_k(\mathbf{x}) &= \ln\{P(\omega_k)p(\mathbf{x}|\omega_k)\} \\ &= \ln\, P(\omega_k) - \ln\{(2\pi)^{\frac{n}{2}}|\Sigma_k|^{\frac{1}{2}}\} - \frac{1}{2}(\mathbf{x}-\mu_k)^t\Sigma_k^{-1}(\mathbf{x}-\mu_k) \end{aligned}$$

which simplifies to:

$$g_k(\mathbf{x}) = \ln\, P(\omega_k) - \frac{1}{2}\left(\ln\,|\Sigma_k| + (\mathbf{x}-\mu_k)^t\Sigma_k^{-1}(\mathbf{x}-\mu_k)\right)$$

If we assume that all class-conditional distributions $p(\mathbf{x}|\omega_k)$ have the *same* covariance matrix Σ, we get:

$$g_k(\mathbf{x}) = \ln\ P(\omega_k) + \mu_k^t \Sigma^{-1}\mathbf{x} - \frac{1}{2}\mu_k^t \Sigma^{-1}\mu_k$$

These are **linear discriminant functions** (linear in $\mathbf{x}$) and the **decision boundaries** $g_i(\mathbf{x}) = g_j(\mathbf{x})$ are hyperplanes in n-dimensional space.

In practical situations, only an i.i.d data sample S is available. When means, covariances and priors for every class are not available, maximum-likelihood estimates on S can be used, although in this case the Bayesian optimality properties are no longer valid. Let $S_k \subset S$ be the subset of samples known to belong to class ω_k. Then $S_1, \ldots, S_c$ is a partition of S. Unbiased estimates for the vector means and for the class priors can be obtained as:

$$\mu_k \approx \hat{\mu}_k = \frac{1}{|S_k|}\sum_{\mathbf{x}\in S_k}\mathbf{x}; \qquad P(\omega_k) \approx \hat{P}(\omega_k) = \frac{|S_k|}{|S|}$$

The following *pooled* covariance matrix is then used:

$$\Sigma \approx \hat{\Sigma}_{\text{pooled}} = \frac{1}{|S|-c}\sum_{k=1}^{c}(|S_k|-1)\hat{\Sigma}_k$$

where

$$\hat{\Sigma}_k = \frac{1}{|S_k|-1}\sum_{\mathbf{x}\in S_k}(\mathbf{x}-\hat{\mu}_k)(\mathbf{x}-\hat{\mu}_k)^t$$

5.2.3 Support Vector Machines

The support vector machine (SVM) is a machine learning method solidly based on statistical learning theory [46]. Intuitively, given a set of examples labeled into one of two classes, the linear SVM finds their optimal linear separation: this is the hyperplane that maximizes the minimum orthogonal distance to a point of either class (this distance is called *margin* of the separation).

Consider again an i.i.d data sample $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ of training patterns (in n dimensions), labelled into two classes ω_1, ω_2 by $z_1, \ldots, z_N$, with $z_i = +1$ if $\mathbf{x}_i \in \omega_1$ and $z_i = -1$ if $\mathbf{x}_i \in \omega_2$. If we set up an affine function $g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$, then we have a linear discriminant as $sgn(g(\mathbf{x}))$, for which we would like:

$$\begin{aligned} &\langle \mathbf{w}, \mathbf{x}_i \rangle + b > 0 \qquad \mathbf{x}_i \in \omega_1 \, (z_i = +1) \\ &\langle \mathbf{w}, \mathbf{x}_i \rangle + b < 0 \qquad \mathbf{x}_i \in \omega_2 \, (z_i = -1) \end{aligned}$$

In short, $z_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) > 0$, or $z_i \, g(\mathbf{x}_i) > 0$, for all $1 \leq i \leq N$. Given the hyperplane $\pi : g(\mathbf{x}) = 0$, the perpendicular distance from $\mathbf{x}$ to π is $d(\mathbf{x}, \pi) = \frac{|g(\mathbf{x})|}{\|\mathbf{w}\|}$. The *support vectors* are those $\mathbf{x}$ closest to the hyperplane. Rescaling $\mathbf{w}, b$ such that $| \langle \mathbf{w}, \mathbf{x} \rangle + b| = 1$ for these closest points, one obtains $| \langle \mathbf{w}, \mathbf{x} \rangle + b| \geq 1$. The *support vectors* are now those $\{\mathbf{x}_i \, / \, | \langle \mathbf{w}, \mathbf{x}_i \rangle + b| = 1\}$.

The *margin* $m(\pi)$ of a plane π can now be written as twice its distance of any support vector: $m(\pi) = 2 \, d(\mathbf{x}_{\mathrm{SV}}, \pi) = \frac{2}{\|\mathbf{w}\|}$, where $|g(\mathbf{x}_{\mathrm{SV}})| = 1$. To maximize the margin, we should minimize $\|\mathbf{w}\|$ subject to $z_i \, (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$, for all $1 \leq i \leq N$.

In the case where an hyperplane does not exist that can separate correctly the points in the data sample, a set of non-negative *slack* variables are introduced to allow for small *margin violations*, leading to a *soft margin*:

$$z_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) + \xi_i \geq 1 \qquad i = 1, ..., n \tag{10}$$

where $\xi_i \geq 0$. For an error to occur, the corresponding ξ_i must exceed unity, and so $\sum_i \xi_i$ is an upper bound on the number of *training* errors. The optimal separating hyperplane can be found as the solution of the 1-norm Quadratic Programming problem:

$$\min_{\mathbf{w},\xi} \qquad \tfrac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\xi_i$$

$$s.t. \quad z_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \; i = 1, \ldots, N$$

The solution to this optimization problem corresponds to the saddle point of its associated Lagrangian:

$$\frac{||\mathbf{w}||^2}{2} - \sum_{i=1}^{N} \alpha_i(z_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 + \xi_i) + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\mu_i\xi_i$$

where $\alpha_i, \mu_i \geq 0$ for $i = 1, ..., n.$

Once this QP problem is solved, the solution vector $\mathbf{w}^*$ can be expressed as a linear expansion over the support vectors:

$$\mathbf{w}^* = \sum_{i=1}^{N} \alpha_i^* z_i \mathbf{x}_i \tag{11}$$

The support vectors are precisely those $\mathbf{x}_i \in S$ for which $\alpha_i^* > 0$.

5.2.4 Multinomial

Multinomial regression is a simple neural network with no hidden layer. Non linearities are created by using a sigmoid function at the neurons of the output layer. This is illustrated in figure 7. For multi-class problems more than one output neurone is used for each class. The weights are optimised with the help of a typical decent algorithm.

5.3 Results and Inference

Figure 8 shows the average test error for the different datasets using KNN as a learner in the similarity representation model discussed earlier. The different prototype

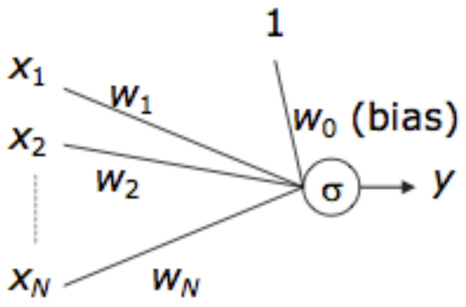


Figure 7: Multinomial

Here the weights are applied to each feature before passing it through a sigmoid function at the neuron

selection methods are compared. Table 3 shows the average test error values over all the datasets. Apart from the Audiology dataset, there is not much difference in the test error. The ReliefF method for prototype selection offers the worst results amoung the methodologies compared. It is the simplest method amoung the different methods being considered, and does not do a very good job at prototype selection.

Method	**Average error(%)**
Euclidean-KNN	22.2
Gower-KNN	22.1
Sim-PAMnonClass_KNN	23.3
Sim-PAMClassical-KNN	23.9
Sim-relief_KNN	25.9
Sim-CFS_KNN	23.3

Table 3: Comparing Prototype selection methods

Figure 9 shows the average test errors over all datasets with the similarity representation methodology for different prototype selection methods and more sophisticated learners than KNN i.e. Logistic regression, Multinomial regression, LDA and SVM. It can be seen that there is not much difference in performance comparing

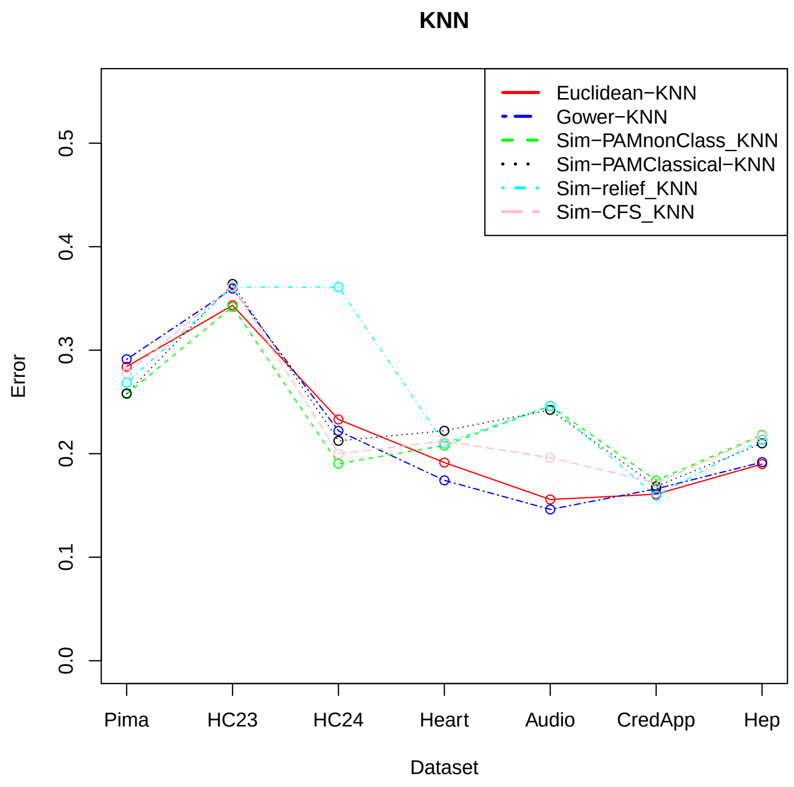


Figure 8: K-Nearest Neighbour results

Average test error values for all datasets considered for simple KNN method shown in red, KNN with Gower's measure in blue and Similarity based representation with KNN as learner using prototyping as PAM in original space shown in black, PAM on similarity space in green, ReliefF in cyan and CFS in pink

clustering and feature selection for the task of prototype selection. ReliefF again performs the worst of all the prototype selection methods. CFS, another feature selection method, on the other hand performs quiet well. As described earlier CFS is just based on correlation or linear relationships between the features and the targets. It is possible that a more sophisticated, non-linear, feature selection method would produce better results. Interestingly PAM used on the $N \times N$ similarity matrix works as well as that used on the original dataset.

In general most methods (other than ReliefF) work better than Euclidean KNN. SVM when used as a learner after the similarity representation doesn't work well, but this could be due to the poor tuning of its parameters. The rest of the learners being compared work almost equally well. Logistic regression in particular works well when applicable and consistently shows better prformance then Euclidean KNN. Section 18 shows the plots for each dataset. Except for *Audiology*, the datasets perform better with this similarity mechanism than with the simple Euclidean KNN. As seen in figure 8 simple Gower-KNN works as well as Euclidean KNN. Hence, the problem then is most likely with the selection of prototypes.

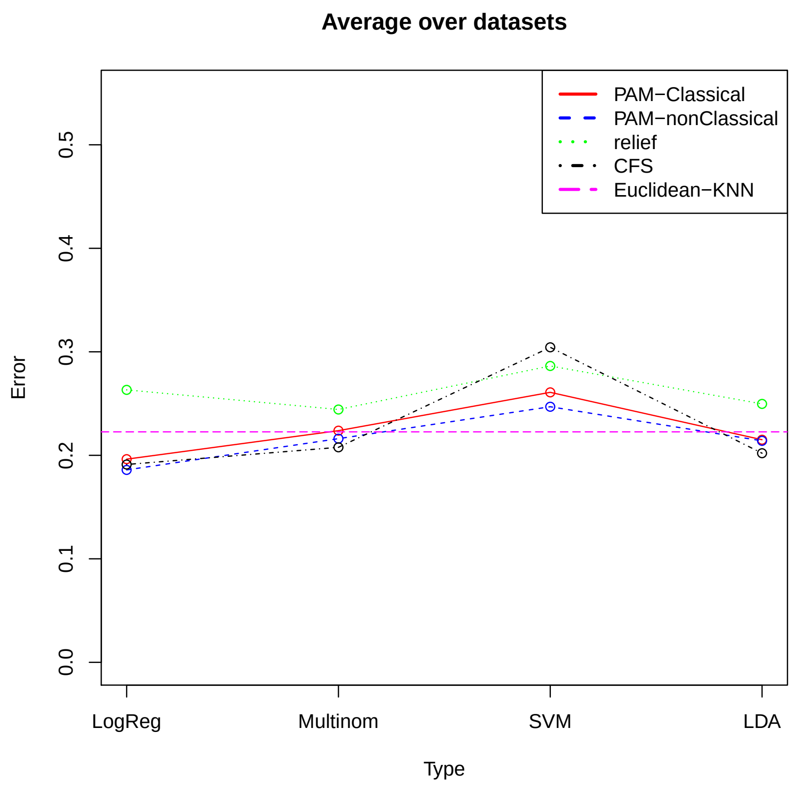


Figure 9: Comparison of different prototype selection methods and learners

The figure shows average results for all datasets for different prototype selection methods (as different plots) and different learners (on x-axis). The horizontal line shows the test error of simple Euclidean KNN for comaprison

This methodology offers the advantage over simple KNN that only the similarity to a certain number of prototypes have to be calculated and not the whole training dataset. Hence, although not providing significantly better results in terms of classification error, this methodology provides an reduction in computation.

6 Improved Similarity Measure

The evaluation of the Gower similarity metric can involve a series of weights for each feature, as shown equation (12). Here the S_{ij} refers to the similarity measure, between the ith and jth instances of the data. s_{ijk} is the similarity measure between the the kth feature of the ith and jth instances, w_k to the weights assigned to the kth feature. The quantity δ_{ijk} represents the possibility to make a comparison, and is usually 1 when feature k can be compared for instances i and j and 0 otherwise.

$$S_{ij} = \frac{\sum_{k=1}^{d} s_{ijk} w_k}{\sum_{k=1}^{d} \delta_{ijk} w_k} \tag{12}$$

By default the weights are set equal, meaning an equal importance is given to all the features. Assigning weights can increase or decrease the relative influence of the features on the similarity measure. This could possibly lead to better separation of classes in the new similarity space. The difficulty is in choosing the weights.

The subsequent sections will describe the methodology of the theory, the experimental setup and results obtained.

6.1 Methodology

This thesis proposes the following methods to tackle the problem of choosing the weights in an attempt to create a better similarity representation.

6.1.1 Relief Algorithm

A simple way to assign weights to the features is the use of the RELIEFF algorithm (as described in section 5.1.2). This algorithm assigns weights to each of the features corresponding to its relevance for the problem of classification.

6.1.2 Least squares

In this approach the task of assigning weights is treated as the equivalent of solving the following least squares problem as illustrated in equation (13). Here the target, t_{ij}, is 1 if the ith and jth instances belong to the same class and is 0 otherwise. We assume that the weights w_k are positive and sum to 1, so that the denominator of the similarity measure, from equation (12), can be eliminated from the optimisation problem. d is the number of features. As the original similarity matrix S is symmetric, only the upper triangle values of the matrix need to be optimised.

$$\begin{aligned} \underset{w_k}{\text{minimize}} \quad & \sum_{i=1}^{N}\sum_{j>i} |\sum_{k=1}^{d} s_{ijk}w_k - t_{ij}|^2 \\ \text{subject to} \quad & \sum_{k=1}^{d} wk = 1 \\ & w_k \geq 0,\ k = 1, \ldots, d. \end{aligned} \tag{13}$$

Intuitively, this means that the similarity measure S_{ij} should be as close to 1 as possible when the ith and jth instances are of the same class and 0 otherwise. This approach tries to separate instances of different class and bring closer those of the same class. There are a few drawbacks of this procedure.

1. Firstly, this problem has ${}^{N}C_2$ i.e. $N(N-1)/2$ equations where N is the number of instances. This can cause the problem to become very computationally intensive for large datasets. One possible way to counter this is to do feature selection via clustering on the original data space before the optimisation. This means that the similarity matrix to be optimised now is reduced from $N \times N$ to $N \times d_1$, where d_1 is the number of prototypes chosen by the clustering algorithm in the original data space. This reduces the number of equations to $N.d_1$.

2. Another drawback is in the way this problem is designed. As explained earlier, this approach tries to bring closer all instances with the same class and vice versa. This might be unnatural in some cases. Let us consider the example as

shown in Figure 10. Here all the instances of the blue class are not similar to each other i.e. they are separated by they instances of the red class. If the previous idea for optimisation is applied here, the similarity measure would try to force all the instances of the blue class to be similar which is unnatural and may lead to worse results. Instead what is desired is that cluster instances of the blue class in the left and right must be individually separated from the instances of the red class. The flaw in the previous optimisation procedure which is causing this problem is that the similarities are optimised globally, whereas it might more appropriate if done locally. One approach to solve this problem is to optimise the weights considering instances of each cluster individually. For example if the dataset was separated to 3 clusters, the instances (i and j in equation 13) chosen to optimise the weights are all possible combinations of within each cluster.

If both these strategies are combined, the number of equations in the objective function reduces to N.

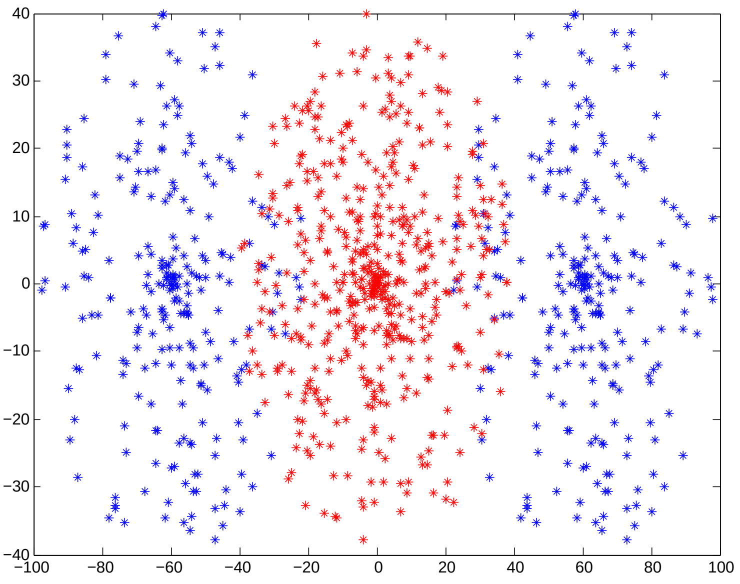


Figure 10: Example

Here the blue class is naturally separated and hence all instances of this class will not be similar to each other

Equation (13) can be rewritten in matrix notation as shown below in (14). Matrix $\widetilde{S}$ is $L \times d$, where L is the number of equations in the objective function. Each row of $\widetilde{S}$ corresponds to to a particular combination of ith and jth instances. $\mathbf{w}$ is a d dimensional vector of the weights to be optimised and $\mathbf{t}$ is a vector containing the L target values corresponding to t_{ij} from equation (13)

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & ||\widetilde{S}\mathbf{w} - \mathbf{t}||^2 \\ \text{subject to} \quad & \mathbf{1}\mathbf{w} = 1 \\ & \mathbf{w} \geq \mathbf{0} \end{aligned} \tag{14}$$

6.2 Experimental Setup

The *'heterog'* version of the datasets are used i.e. containing the nominal, ordinal and missing values in the original form. The data is split to test and training as described in table 1. Let N_{tr} be the number of instances in the training set and N_{te} be the number of instances in the test set. The two methodologies discussed for generating weights for features i.e. ReliefF and the least squares are implemented on the training set to obtain a vector of weights. This is used to obtain he $N_{tr} \times N_{tr}$ similarity matrix is calculated using the Gower's similarity measure. The rest of the procedure is repeated similar to the previous as described in section 5.2

6.3 Results and Inference

Figure 16 shows the average test errors over all datasets with the improved similarity representation using the RELIEFF algorithm discussed earlier. It is compared, as earlier, for the different prototype selection methods and learners i.e. Logistic regression, Multinomial regression, LDA and SVM. The change varies depending on the dataset as shown in section 19. This is especially evident for the RELIEFF prototype selection method. This indicates that a more sophisticated methodology for including weights could potentially greatly improve the results.The disadvantage with the chosen datasets is that the features are all relevant for the problem of classification i.e. feature selection has already been done. The results could be much better for general problems where some features may be irrelevant.

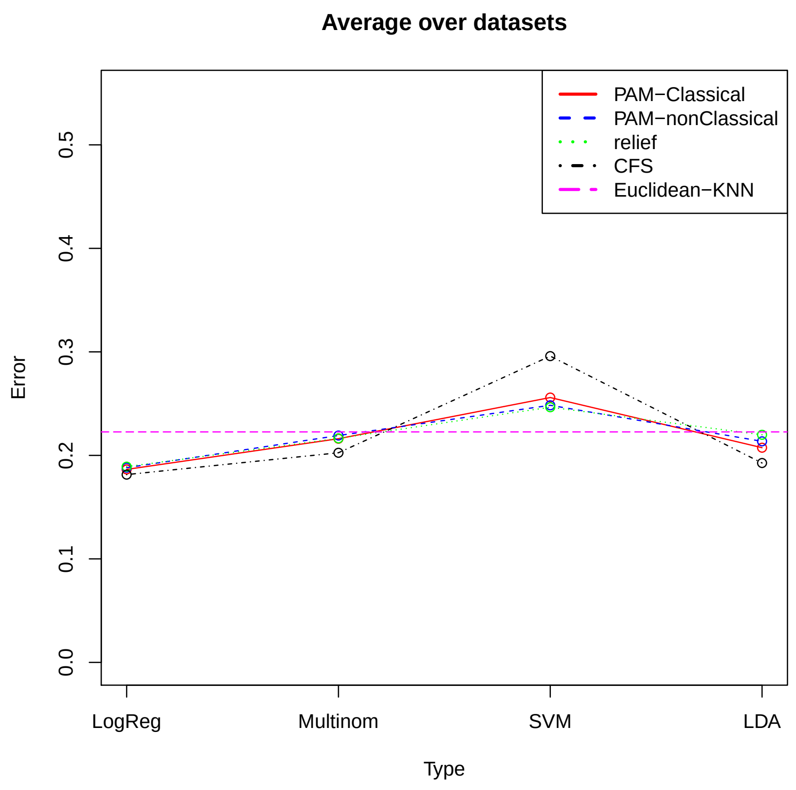


Figure 11: Comparison using improved similarity measure with RELIEFF

The figure shows average results for all datasets for different prototype selection methods (as different plots) and different learners (on x-axis). The horizontal line shows the test error of simple Euclidean KNN for comaprison

Figure 12 shows average test errors over all datasets with the improved similarity representation using the least squares problem. Only the methodology optimising the similarities within a cluster with respect to the corresponding prototype was explored. Section 20 shows the results for each dataset. The results are considerably worse than the previously discussed methods. This is because the weights are not chosen well. The major problems with this could be

- This methodology is very susceptible to when the prototypes chosen are outliers with respect to class labels.

- A more robust objective function than the one used here needs to be developed

.

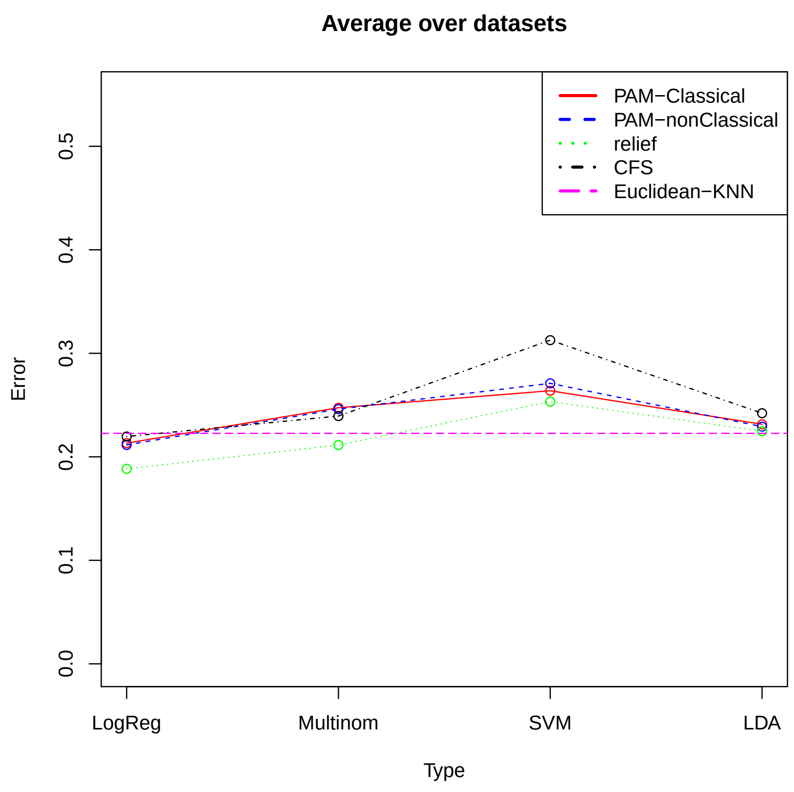


Figure 12: Comparison using improved similarity measure with lease squares

The figure shows average results for all datasets for different prototype selection methods (as different plots) and different learners (on x-axis). The horizontal line shows the test error of simple Euclidean KNN for comaprison

7 Higher Level Similarities

In similarity based learning data from the original space of, for example, N instances and d features i.e. an $N \times d$ matrix is transformed to a space of N instances and d_1 features i.e. an $N \times d_!$ matrix. Each instance in the new representation corresponds to its similarity between a few chosen instances (called prototypes). This representation is defined by not just the values of certain features but the topological location of instances with respect to a few prototypes.

In a way, we moving from a local representation to a global representation. This intuition can be extended to a representation where each instance contains information not just about its similarity to other instances but about the global topological information. This representation, which we refer to as higher level similarities, can potentially be ricer and more informative.

Listed below are some of the motivations of switching to a higher level similarity representation

- As we move from the original representation to a more global one, where each instance has information about the entire topology, the influence of outliers will reduce.
- Intuitively this could mean that simpler models, such as linear/quadratic be good enough to learn the data.

This thesis explores three different approaches of calculating higher level similarities. They are elaborated in sections 7.1 and 7.2

7.1 Similarity of Similarity

In this approach, higher level similarities are extracted by iterating the methodology of transforming the original data space to the similarity data space. The process of calculating the initial similarity representation, denoted here by, S^1 (i.e. the $N \times d_1$

matrix) is repeated. S^1 matrix is considered as a new dataset and is subject to clustering to choose d_2 prototypes. The similarity between all the N instances and these d_2 prototypes is calculated from S^1 to form the new set of features. This $N \times d_2$ matrix representation can be called as the second level of similarities denoted by S^2. Here, each instance now corresponds to the similarity between the first level of similarities. This process can be iterated for a number of loops.

This can be visualized as a deep neural network to extend the ideas presented in section 2.6. Each hidden layer corresponds to one iteration of clustering and similarity calculation. The number of hidden layers corresponds to the number of iterations. Also, the number of neurons in each hidden layer the neurons represents the prototypes chosen for that particular iteration. For example, if there are m iterations and $d_1, d_2, ..., d_m$ are the number of prototypes chosen for each loop, then the neural network representation will have m hidden layers with each $d_1, d_2, ..., d_m$ neurons respectively. This is illustrated in the figure 13, shown below below. The figure shows an m-layer deep similarity neural network with the prototypes being represented as neurons.

In the new *'high level'* similarity representation each instance has global/higher level information of the entire topology. This means that instances close to each other in the original data space are very close in the new representation. As the number of iterations increases, the instances are expected to get more and more similar to each other. Hence the similarities will get closer and closer to one and eventually it will be impossible to get any information from the representation. There should be, however an optimal number of iterations for which the separation between the classes is maximum.

7.1.1 Experimental Setup

As previously done, the dataset is split into training and test sets of approximately 70% and 30% respectively. The *'heterog'* version of the data is used. Let N_{tr} be the number of instances in the training set and N_{te} be the number of instances in the test set. Only PAM clustering on the original data space is considered for experiments in this case.

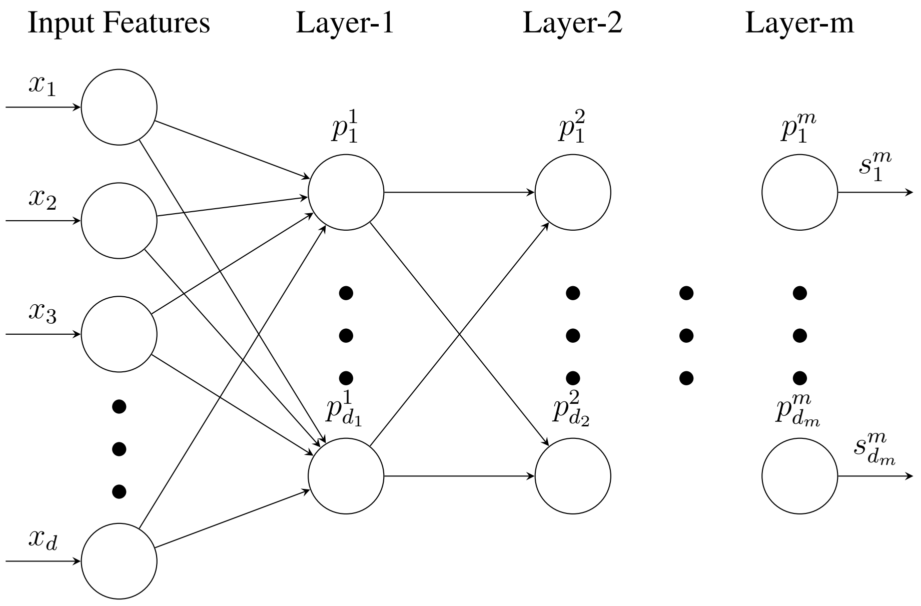


Figure 13: Deep Similarity Neural Network

The first layer represents the d input features for a single instance in the original data space. Layer-1 shows the d_1 prototypes selected from S^1 as $p_1^1 ... p_{d_1}^1$. Prototypes of layer-2 are chosen from S^2 and are denoted by $p_1^2 ... p_{d_2}^2$ and so on. There are m layers in total and the final feature space are instances in the S^m similarity data space. Features in this space are denoted by $s_1^m ... s_{d_m}^m$

The prototypes chosen at each level of the iterative process described earlier are saved for testing. The similarity matrix at each iteration is calculated using the Gower's similarity measure.

Let there are m iterations and $d_1, d_2, ..., d_m$ are the number of prototypes chosen for each loop. PAM clustering is used on the original data space i.e. the $N_{tr} \times d$ matrix. The layer-1 similarity representation, S^1, is calculated. This process is repeated for m layers. The final representation i.e. the $N_{tr} \times d_m$ similarity matrix trained with the learners described in section 5.2 i.e. LDA, SVM, Multinom and Logistic regression. For testing, the N_{te} instances are passed through the iterative process with the prototypes $d_1, d_2, ..., d_m$ chosen during training. The final matrix $N_{te} \times d_m$ is used to cacluate the prediction of the trained learner.

Cross validating the number of prototypes at each iteration can be computationally very intensive. Hence for the sake of this study two approaches are explored as discussed below

1. ***Number of prototypes decreasing over the different layers*** :- Here the number of prototypes is lowered as the number of the iteration is increased. For example in in a 2 layer similarity measure, the number of prototypes double be 10 for the first layer and 5 for the second. One intuitive reason is as we move higher in the similarity loops, each instance already contains a lot of information about the topology and hence we may not need as many features(prototypes) to represent the dataset in this similarity space. Experimentally is done such that the final number of neurons(d_m) is fixed, and reduces after each loop at a constant rate. d_m values of 3% and 6% of N_{tr} are used.

2. ***Number of prototypes constant over the different iterations*** :- The option of having the number of prototypes constant over the different levels is also tested i.e. $d_1 = d_2 = ... = d_m$. The values chosen for the number of prototypes are of 5% and 10% of N_{tr}.

7.1.2 Results and Inference

The plots show the Credit Approval dataset for the different methods of reducing the number of prototypes at each layer. There is not much difference in the different variations of choosing the number of prototypes for each layer. The tests results as expected increase with the number of layers as the instances get more and more similar to each other. But the performance does not improve in the higher level representation.

The plots for the rest of the datasets are shown in figures 21, 22, 23 and 24. The overall results for different datasets are inconsistent. For most of the datasets the test error eventually increases as the number of iterations/layers increases. The test errors of the high level representation reduce only for a few datasets and only by a very small margin. It is possible that better prototype selection techniques may improve the results. Also optimisation of the similarity matrix with weights should help here. More work needs to be done to try and improve this methodology.

7.2 Powers of Similarity

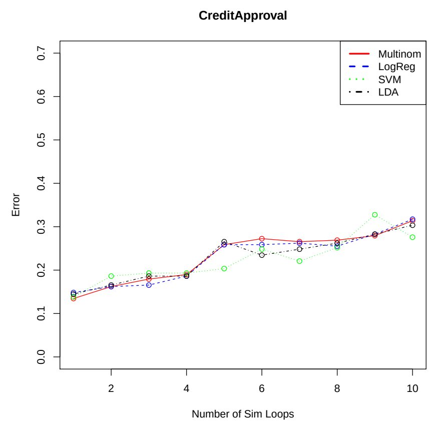
Another possibility explored in this thesis to extract higher levels of similarity is to take powers of the $N \times N$ similarity matrix S i.e. similarity of each instance to every other instance in the dataset. Each term of S is the similarity evaluated between instances in the original space. The closer the two instances are in the original space the closer the measure is to 1. And conversely, if they are dis-similar in the original space the measure will be closer to 0. As discussed earlier, this methodology only considers the instances individually and not the global characteristics.

Powers of S retain global characteristics. For instance the second power of the similarity matrix (S^2) is given by

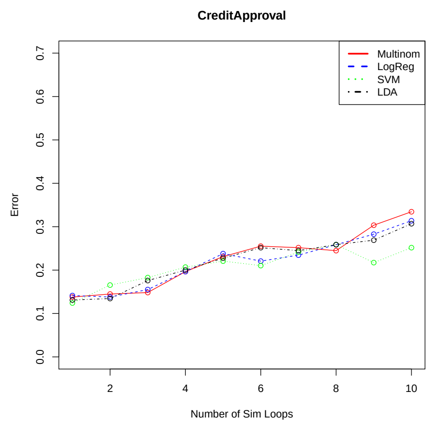
$$S^2_{ij} = \frac{1}{N}\sum_{k=1}^{N}\sqrt{S_{ik}}\sqrt{S_{kj}} \tag{15}$$

The (i, j)th term in the matrix S^2 represents the product of the similarity between the ith instance and an instance k and the similarity between k and the jth instance. This value is averaged over all the instances k in the dataset. This can be visualised as the average distance/similarity of all possible paths from instances i and j separated by an single intermediate instance k. This is illustrated in the figure 15(b). Now we have a new feature which contains global information about the entire topology of the space and not just the similarity between two points. The element-wise square root is taken before taking powers of the matrix in order to prevent the values from concentrating to zero.

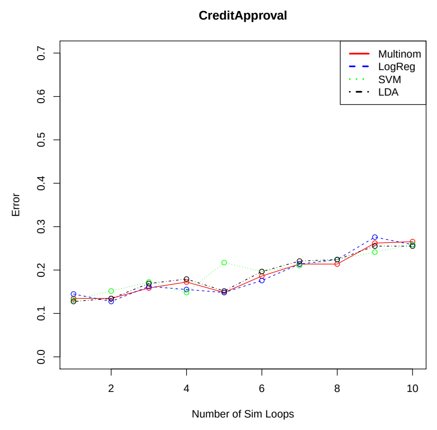
This methodology can be extended to higher powers. For example let us consider the



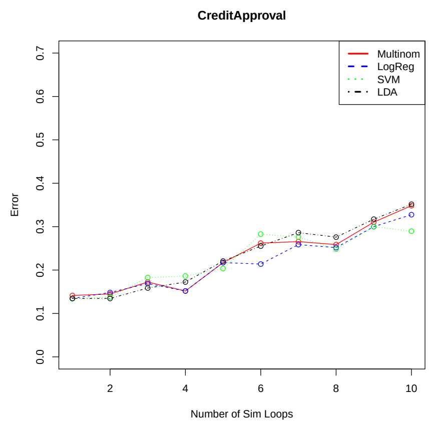
(a) *Reducing with 3% final*



(b) *Reducing with 6% final*



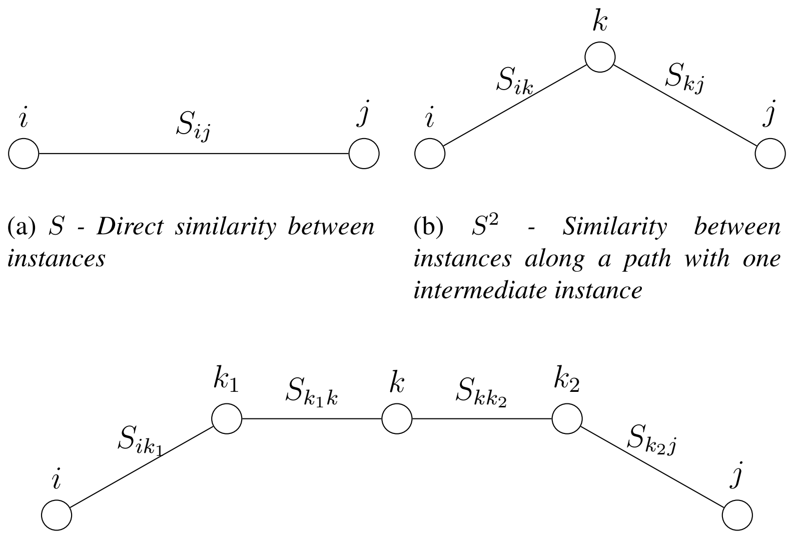
(c) *Constant with 5%*



(d) *Constant with 10%*

Figure 14: Higher Level using Similarity of similarity:- CreditApproval dataset

The plots show the test errors for the Credit Approval dataset with the similarity of similarity framework. The different learners are shown as separate lines and the x-axis shows increasing number of iterations. (a) shows the methodology with the number of prototypes for each iteration reducing at a constant rate to a final of 3% of N_{tr} and similarly (b) for 6%. (c) shows the methodology with the number of prototypes for each iteration constant at 5% of N_{tr} and similarly (a) for 10%



(a) S *- Direct similarity between instances*

(b) S^2 *- Similarity between instances along a path with one intermediate instance*

(c) S^4 *- Similarity between instances along a path with three intermediate instances*

Figure 15: Interpretation of Powers of the Similarity Matrix

(a) shows the calculation of the usual direct similarity between two instances i and j, (b) shows the calculation of the similarity using S^2, where the similarity is calculated over a path with one intermediate instance(averaged over all instances), and (c) shows the calculation of the similarity using S^4 i.e. over a path with three intermediate instance(averaged over all instances)

4th power of S. This is given by

$$\begin{aligned} S^4_{ij} &= \frac{1}{N}\sum_{k=1}^{N}(\frac{1}{N}\sum_{k_1=1}^{N}\sqrt{S_{ik_1}}\sqrt{S_{k_1k}})(\frac{1}{N}\sum_{k_2=1}^{N}\sqrt{S_{kk_2}}\sqrt{S_{k_2j}}) \\ &= \frac{1}{N^3}\sum_{k=1}^{N}\sum_{k_1=1}^{N}\sum_{k_2=1}^{N}\sqrt{S_{ik_1}}\sqrt{S_{k_1k}}\sqrt{S_{kk_2}}\sqrt{S_{k_2j}} \end{aligned} \tag{16}$$

Again this can be visualised as the average distance/similarity of all possible paths from instances i and j separated now by three intermediate instance k, k_1 and k_2. This is illustrated in the figure 15(c).

This computation of the higher powers of the S matrix can be made very efficient using its singular value decomposition (SVD). Gower [17] proves that the S matrix is positive semidefinite, if there are no missing vales in the data. If this is the case the SVD is as follows in equation 17, where Q is the matrix of the eigen vectors and D is a diagonal matrix containing the eigen vectors along the diagonal.

$$S = QDQ^* \tag{17}$$

The powers of S can now be simply calculated as shown in eqution 18 since Q is orthogonal.

$$\begin{aligned} S^n &= (QDQ^*)(QDQ^*)....(QDQ^*) \\ &= QD^nQ^* \end{aligned} \tag{18}$$

7.2.1 Experimental Setup

In this case, the problem of learning is considered to be semi-supervised. This means that the test set is also used for the purpose of learning. So the whole set of N instances of the data are used to calculate the $N \times N$ similarity matrix, S. The powers of S are now calculated and PAM clustering is applied to choose the prototypes.

7.2.2 Results and Inference

The errors for this methodology are quite poor. The test errors consistently increase for higher powers of the similarity matrix. They tend to become constant after a few iterations. Modifications in the methodology are needed to get better results.

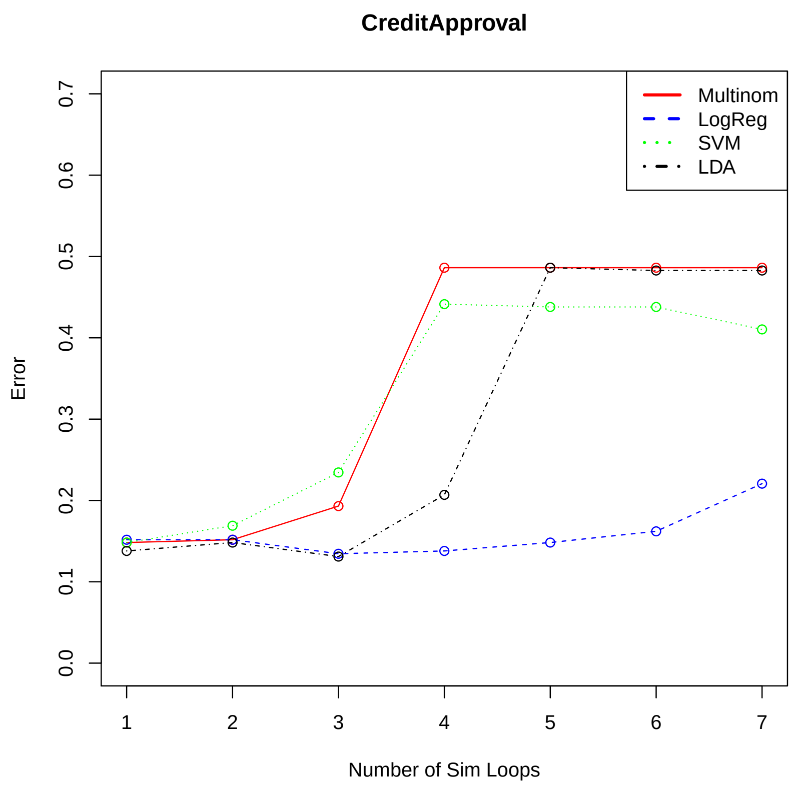


Figure 16: Higher level similarity using Powers of S:-Credit Approval dataset

The figure shows average results for all datasets for different prototype selection methods (as different plots) and different learners (on x-axis). The horizontal line shows the test error of simple Euclidean KNN for comaprison

8 Conclusion and Future Work

The thesis aimed to extend the analysis of similarity based representations laid out by [27, 26] . In particular the thesis dealt with the analysis of heterogenous data with a significant amount of missing information. Gower's similarity measure was used for the analysis. The three directions of study included

- The analysis and comparison of clustering and feature selection methods for the task of prototype selection
- The improvement of Gower's similarity measure with the incorporation of weights for features
- The analysis of possible methods to extract higher level features from the similarity representation

These methodologies were compared with simple K-nearest neighbour techniques. They produce results that are comparable (and in some cases better) than nearest neighbour techniques. The advantage here is that, during testing, only the similarity with a few prototypes needs to be measured rather than with the whole dataset, hence contributing to reduced computational load. This is especially true for large datasets, which are very common in the field of online search, biomedical datasets, etc.

Only very simple feature selection techniques like ReliefF and CFS (i.e. correlation based) were studied here. Overall, these methodologies did not produce significantly better results than PAM (Partitioning Around Medoids) based clustering techniques. More advanced non-linear techniques could possibly lead to better results. Also interestingly PAM(Partitioning Around Medoids) based clustering works as well for prototype selection when used on the $N \times N$ similarity matrix, as it does when used on the original feature space

The similarity representation shows possibility of improvement with the inclusion of weights for the features in the dataset. This is illustrated in the results of the methodology of inclusion of weights using the very simple ReliefF algorithm. An obvious step forward would be to use one of the vast array of sophisticated feature selection mechanisms that assign weights to features such as the Random Forest Algorithm etc. A novel mechanism was also explored to directly obtain weights by trying to minimise a cost function geared at separating the instances of different classes within a cluster. The current thesis only examines a simplified version of this where the similarities are compared with respect to the chosen prototype. This can lead to erroneous results when the prototype chosen is an outlier. This could be one of the main reasons for the ineffectiveness of this method in this study with respect to the method that uses ReliefF or with equal weights.

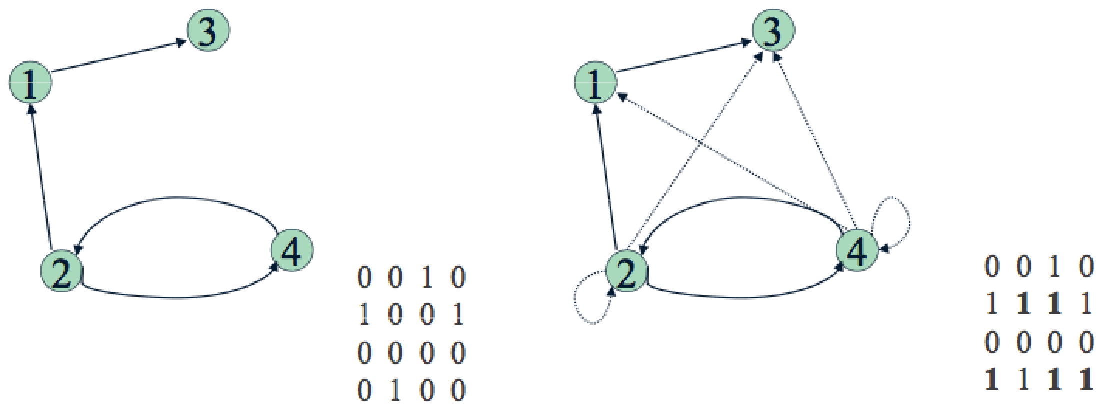
A possible way to avoid this, is to use the similarities of all instances within a cluster, with each other rather than with just the prototype. Also, the current framework (cost function to be minimised) is designed to increase the intra-cluster class separability. It does not take into account the similarities of inter-cluster instances. It is possible that instances of different classes in different clusters are pushed closed together, thus making the problem more difficult. A different cost function needs to be developed that takes into account both these effects. Another for reason the moderate effectiveness of this attempt, is that the datasets chosen for this thesis already have a set of features that are very relevant for the problem of classification. General detests which may contain irrelevant information could potentially be greatly improved by such techniques.

Two approaches to extract deep/high level features from the similarity representation were explored. This included iterating the process of similarity calculation and exploring the powers of the similarity matrix. Initial attempts showed inconsistent results for different datasets. For the former (i.e. the similarity of similarity mechanism), the results as expected worsen as the number of layers increases as all instances get similar to each other. Results improve only for a few cases in the higher level similarity space, for a small number of layers. The improved similarity measure using weighted features could improve results for this methodology, as all the prototypes chosen in the high level similarity space may not be equally important.

For the latter methodology using powers of S, the results are unexpectedly poor. Error values increase for the second power and worsen for further powers. The idea here was to reduce the effect of outliers by calculating similarities along paths averaged over all instances, as shown in figure 15. The performance could improve by averaging over only the local instances. Both these methodologies are only a preliminary attempt to extend the notion of similarity representation to deep learning. They have to be explored in further detail to create an effective model. Another approach along this line could be the use of transitivity to transform the similarity matrix as discussed in section 8.1.

8.1 Transitive Closure

A relation R on a set X is transitive if, for all x, y, z in X, whenever xRy and yRz then xRz. The transitive closure of a binary relation R on a set X is the transitive relation R^+ on set X such that R^+ contains R and R^+ is minimal (Lidl and Pilz 1998:337). This intuition can be extended by visualising a binary directed graph. A binary directed graph is transitive if two nodes are connected by a path via other nodes, then those two nodes are also directly connected (i.e. adjacent). This is illustrated in the figure 17 shown below.



$$\begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix} \qquad \begin{matrix} 0 & 0 & 1 & 0 \\ 1 & \mathbf{1} & \mathbf{1} & 1 \\ 0 & 0 & 0 & 0 \\ \mathbf{1} & 1 & \mathbf{1} & \mathbf{1} \end{matrix}$$

Figure 17: Transitivity in a directed Graph

The figure on the left is a directed graph that does not have the property of transitivity. The adjacency matrix of the graph is shown next to it. The figure on the right shows the same after transitive closure

Applying a transitive closure in the adjacency matrix of the binary directed graph on the left gives us the graph on the right of the figure 17. For example node 1 and node 4 are connected via node 2 in the first graph, but are not directly connected. Hence after the operation of transitive closure nodes 1 and 4 are now adjacent. The dashed lines show the newly added edges. This notion of binary transitivity has been extended to fuzzy relations as studied in [40] using the Floyd-Warshall algorithm. This could be used on the similarity matrix S create a better representation in the similarity space.

Finally, the disadvantage in the current study lies in working with small datasets that for the most part are relatively simple and yield good results with very simple methods. They are most likely already at the Bayesian limit, and could be impossible to improve greatly. Also, the test errors over different datasets are used as the primary evaluation of performance. But, datasets suffer from inherent randomness and are not ideal to monitor progress in every step of research.

The analysis of the methodologies in this thesis should therefore be extended with more complicated heterogenous datasets. Ideal datasets could include online search based datasets, datasets with media based content such as speech/video. In this vein, the developed framework could also be extended to include more sophisticated similarity (or dissimilarity) measures, beyond simple univariate ones. Different techniques of visualising the similarity representation directly also need to be developed. One such method could be the analysis of inter-class and intra-class similarities before and after the change of representation.

To conclude it may be inappropriate to quote the father of machine learning and artificial intelligence Alan Turing - 'I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted.'

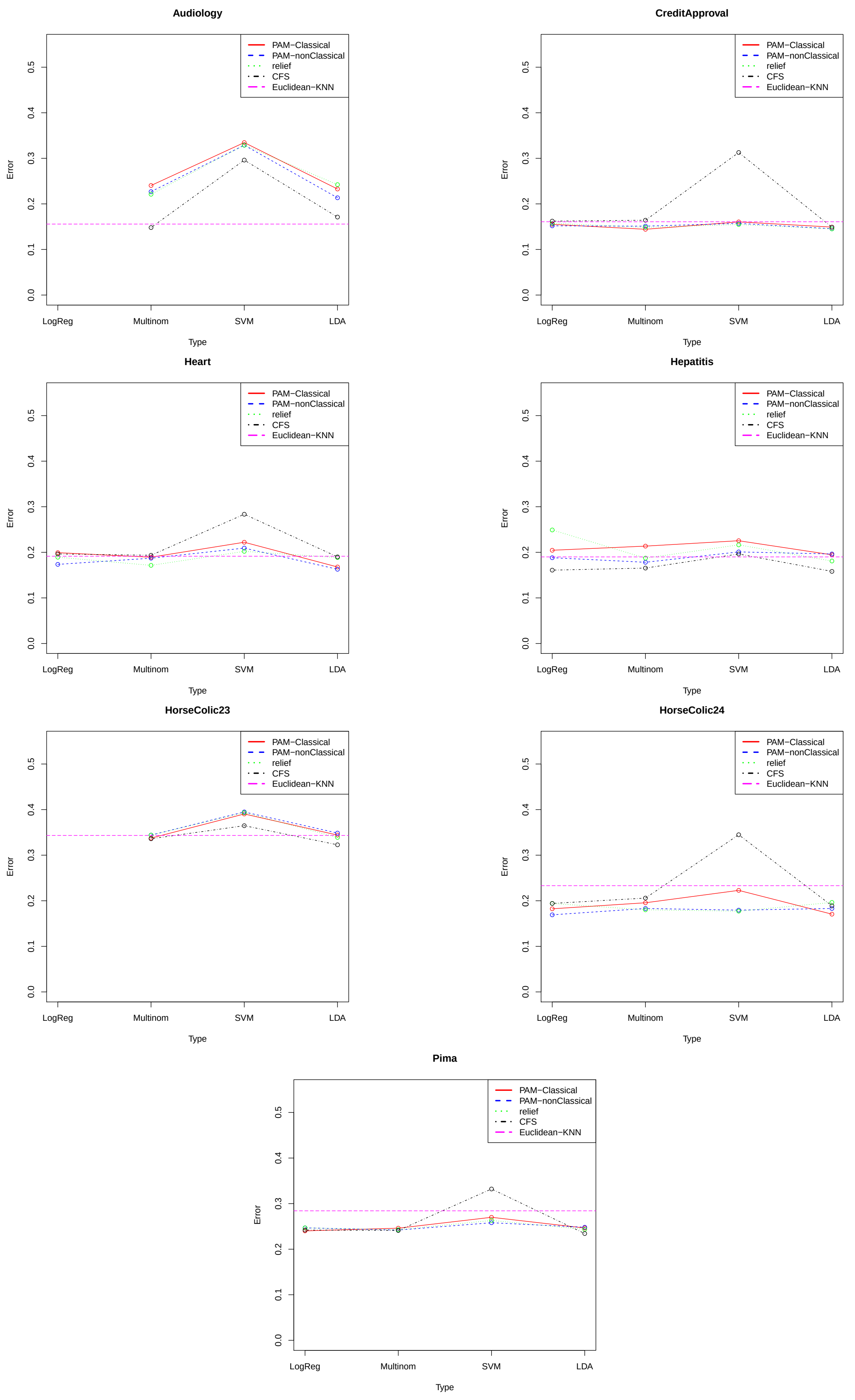
9 Appendix

The appendix includes detailed plots from experiments described in the thesis in section 9.1.

9.1 Results

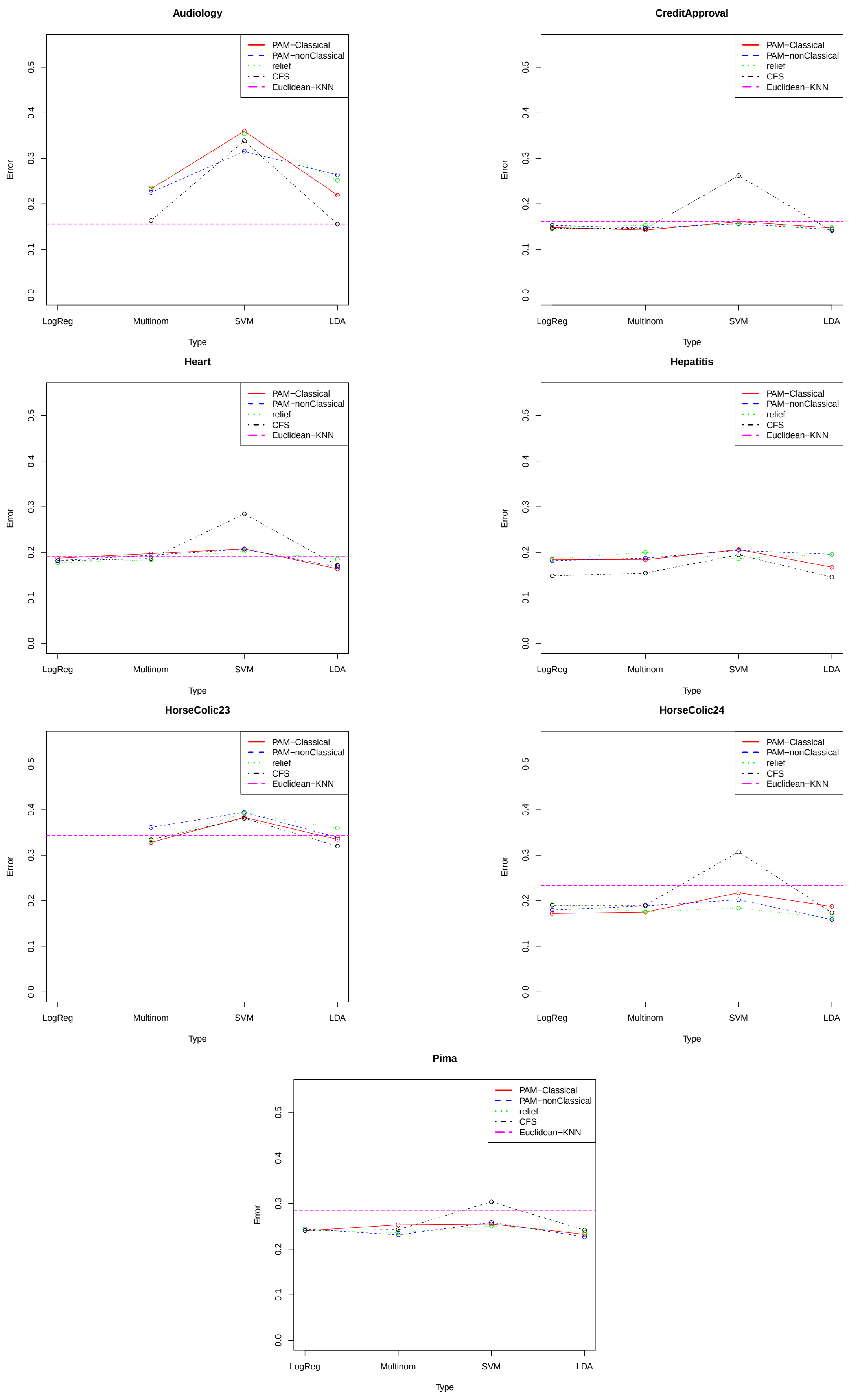
The following includes detailed results of experiments described in previous sections. Figure 18 describes the results for individual datasets for different prototype selection technoques discussed in section 5.2. Figure 19 describes the results for all datasets of improvement of the similarity measure with the ReliefF algorithm and figure20 for the same with the least squares algorithm as described in section 6.2. Figures 21, 22, 23 and 24 show the results of the high level similarity using the '*similarity of similarity*' mechanism described in section 7.1.2. Finally figure 7.2 shows results of the Powers of S mechanism shown in section 7.2.1.

Figure 18: **Prototype selection**



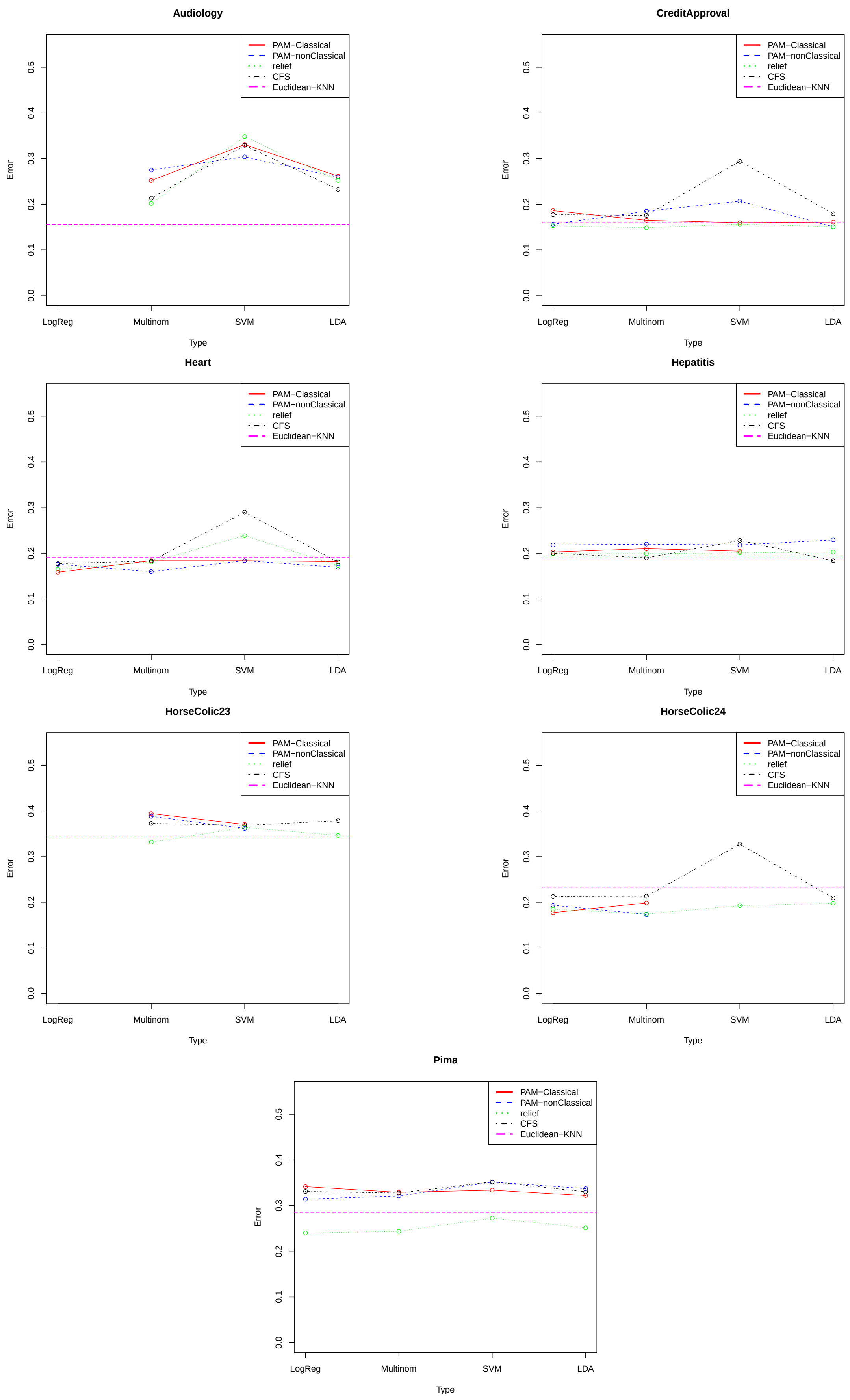
Average test errors of the prototype selection methods for each dataset. The horizontal line shows the test error for simple Euclidean KNN for comparison

Figure 19: **Improved Similarity measure - RELIEFF**



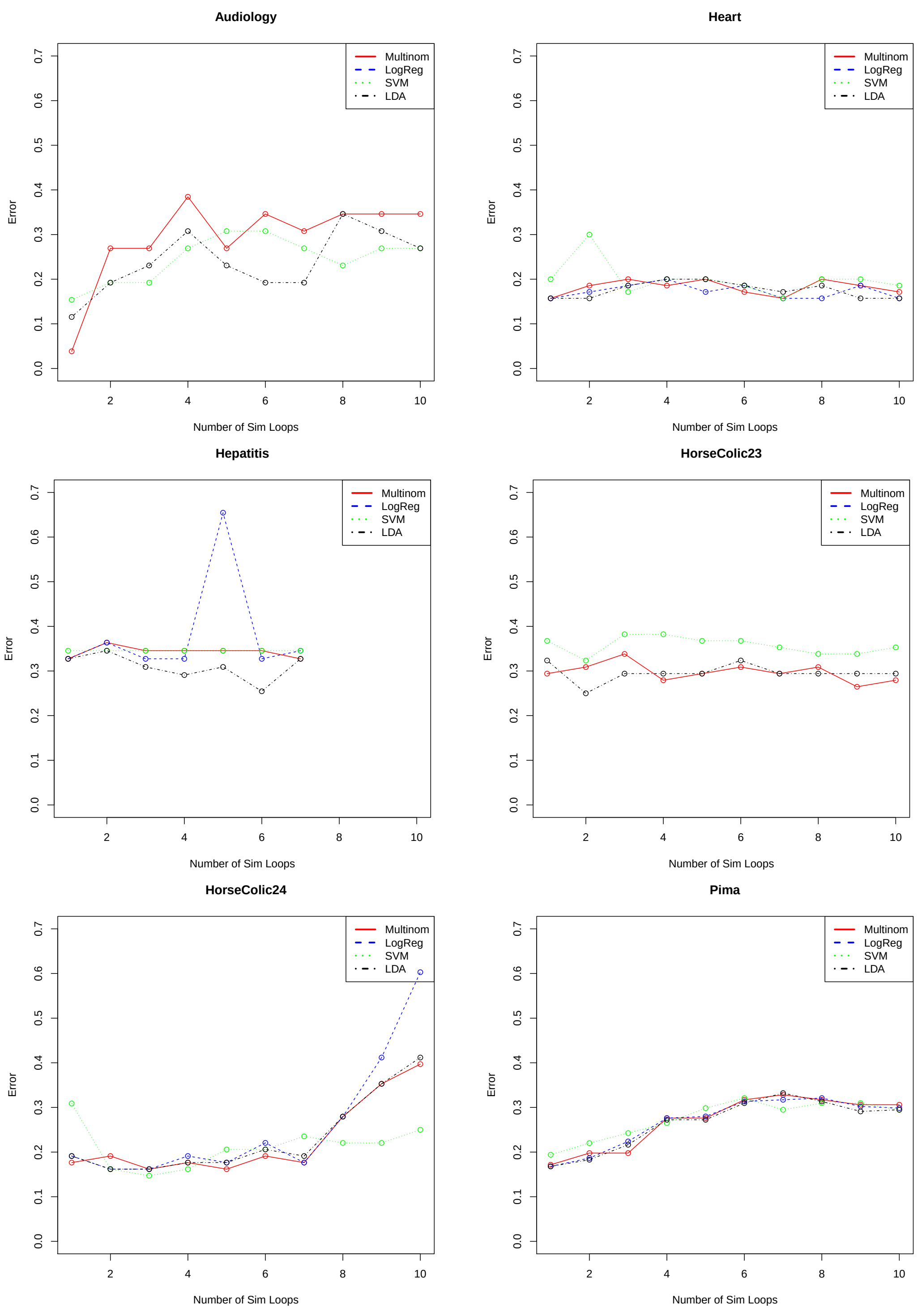
Average test errors of the prototype selection methods for each dataset. The horizontal line shows the test error for simple Euclidean KNN for comparison

Figure 20: **Improved Similarity measure - Least Squares**



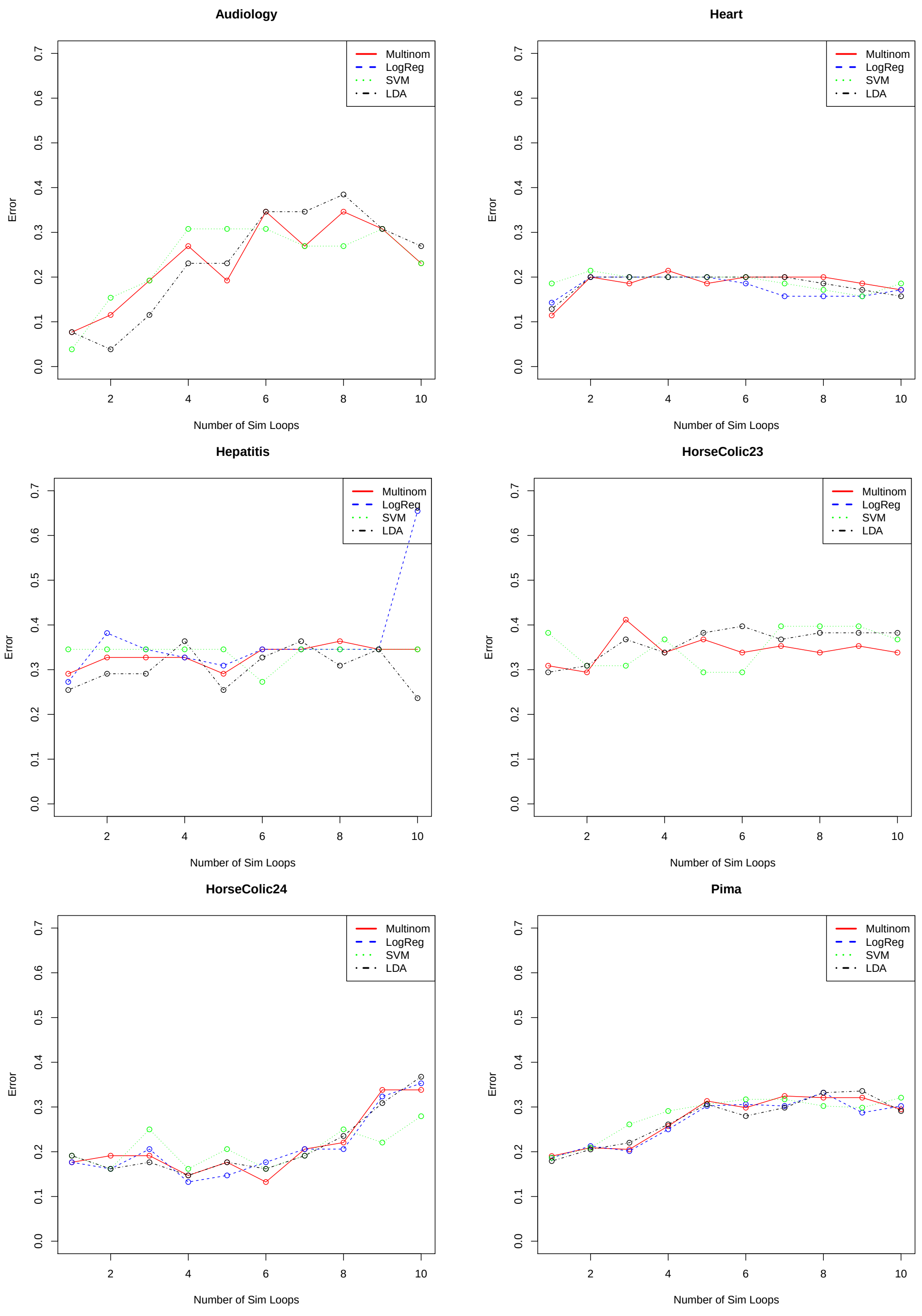
Average test errors of the prototype selection methods for each dataset. The horizontal line shows the test error for simple Euclidean KNN for comparison

Figure 21: **Higher level similarities - Similarity of similarity(3% reducing)**



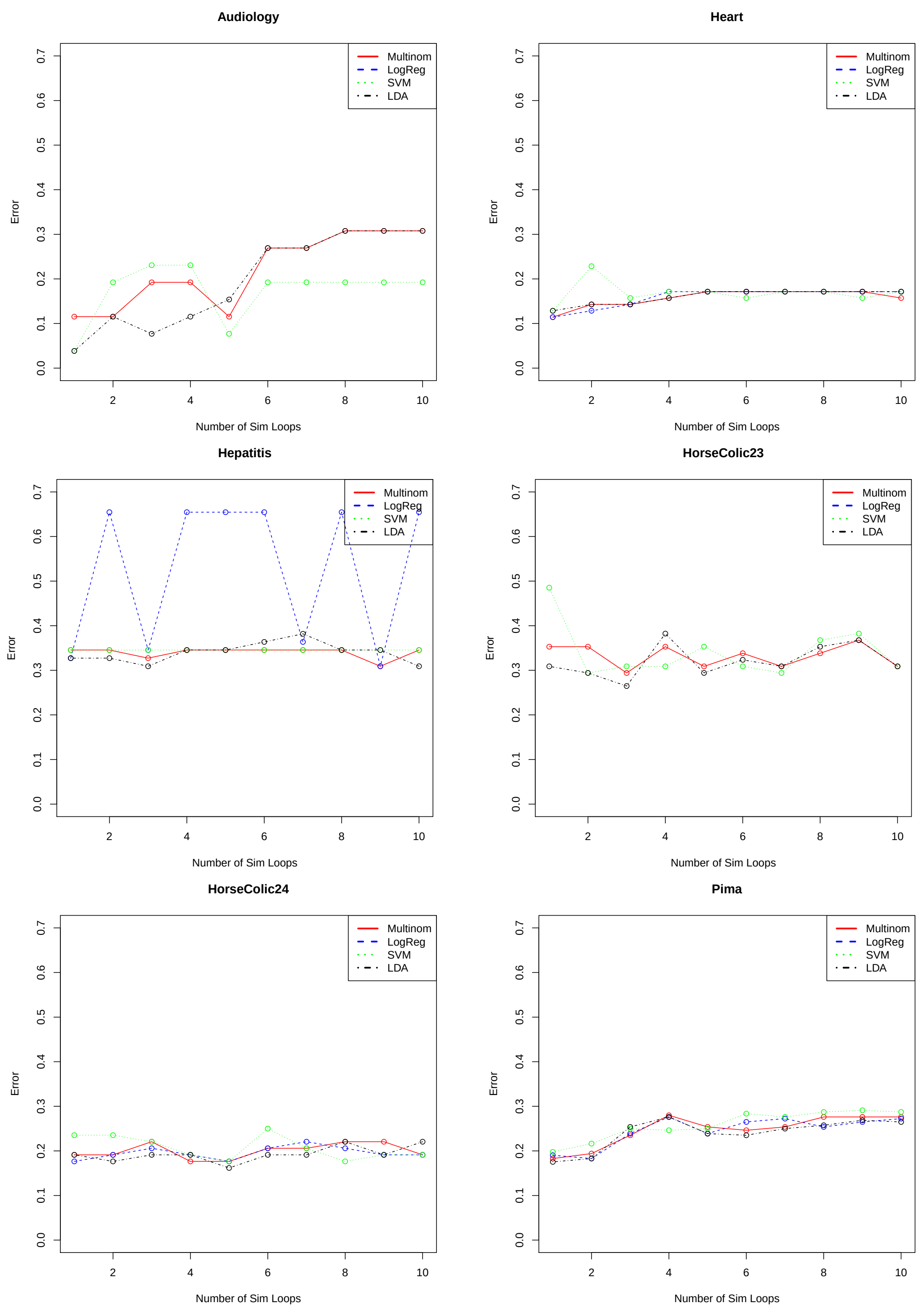
Test errors over increasing number of layers using PAM clustering and different learners. These plots are for the methodology of reducing the number of neurons in each layer with the number of neurons in the last layer as 3% of N_{tr}

Figure 22: **Higher level similarities - Similarity of similarity(6% reducing)**



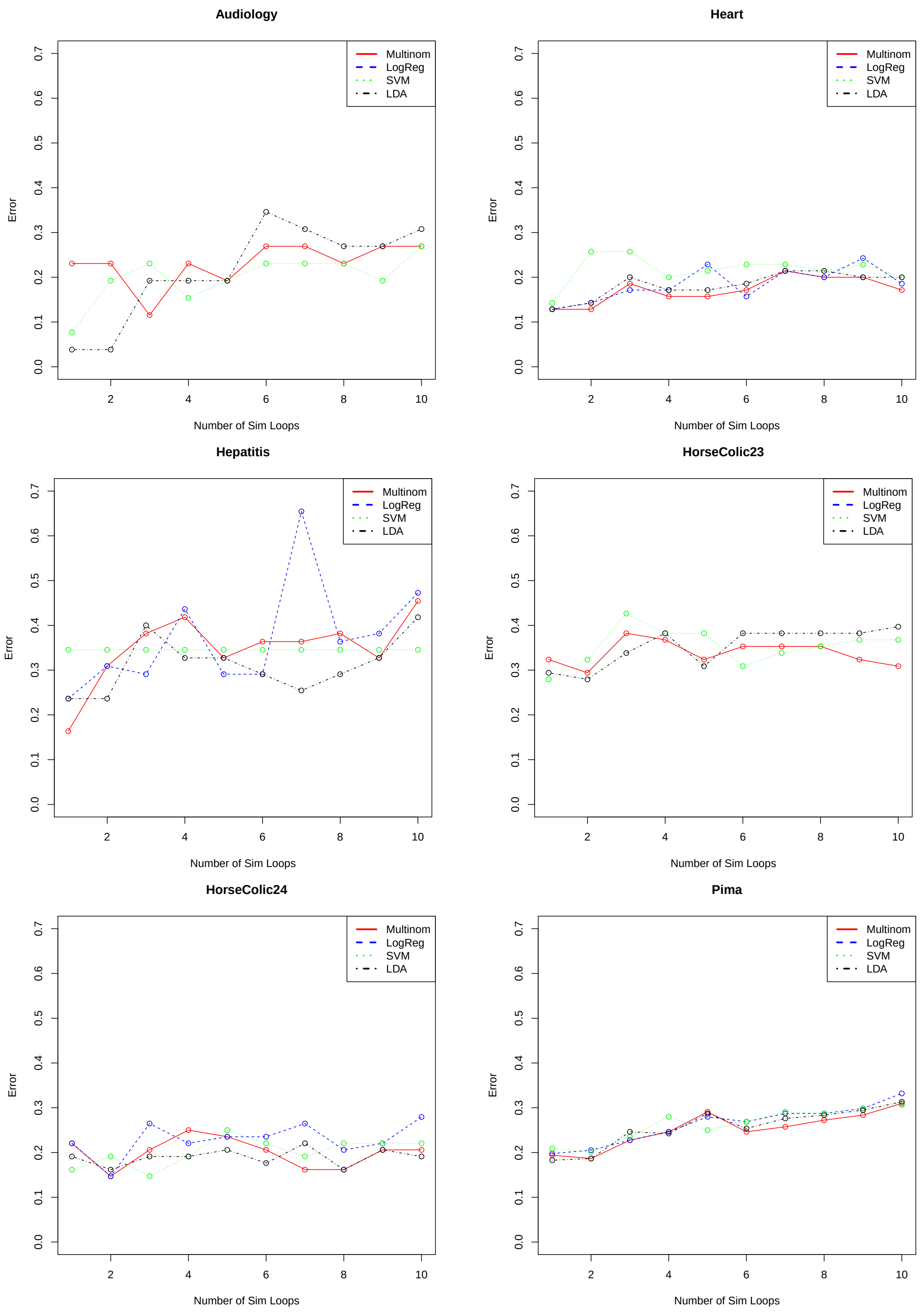
Test errors over increasing number of layers using PAM clustering and different learners. These plots are for the methodology of reducing the number of neurons in each layer with the number of neurons in the last layer as 6% of N_{tr}

Figure 23: **Higher level similarities - Similarity of similarity(5% constant)**



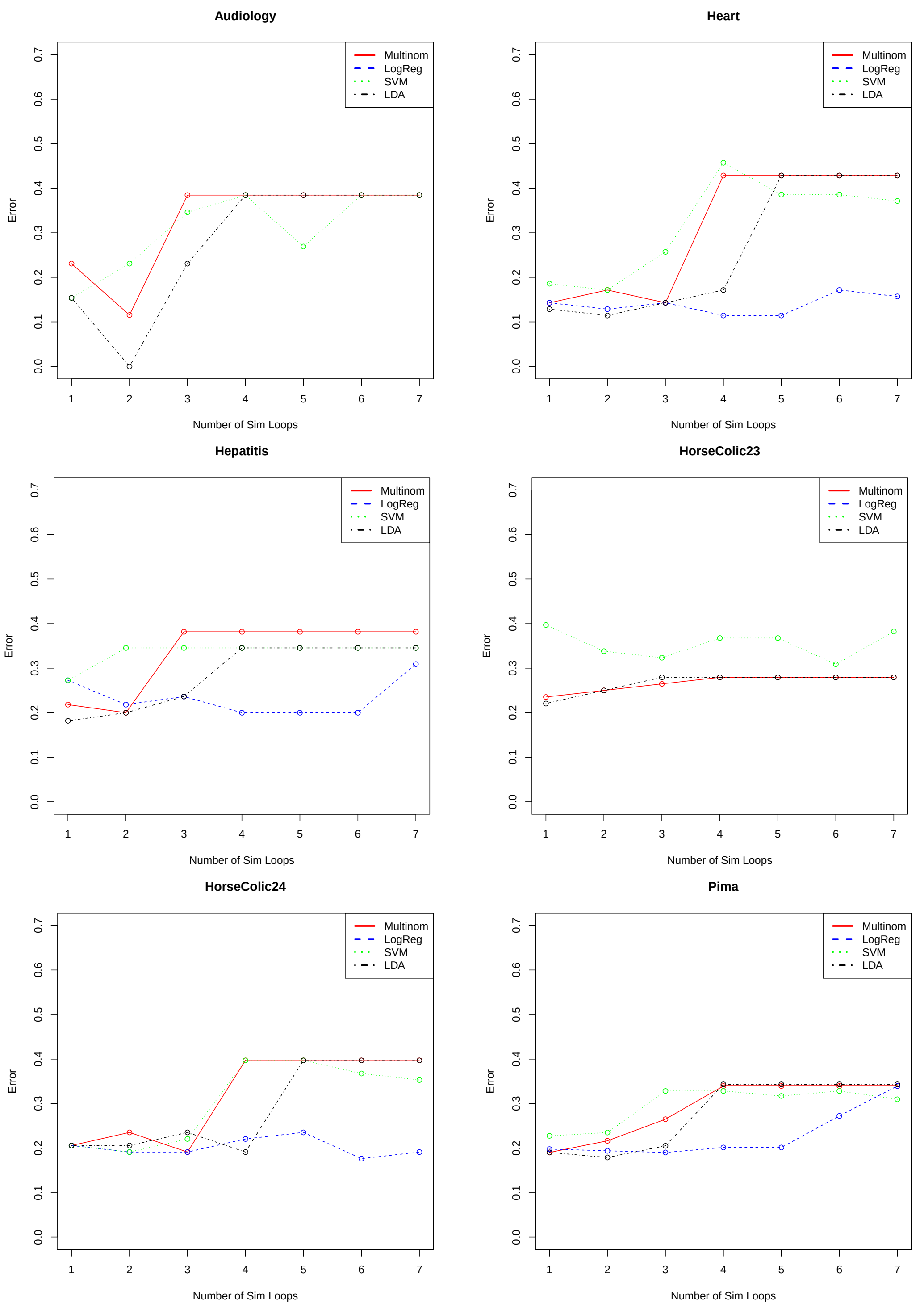
Test errors over increasing number of layers using PAM clustering and different learners. These plots are for the methodology of having the number of neurons constant in each layer as 5% of N_{tr}

Figure 24: **Higher level similarities - Similarity of similarity(10% constant)**



Test errors over increasing number of layers using PAM clustering and different learners. These plots are for the methodology of having the number of neurons constant in each layer as 10% of N_{tr}

Figure 25: **Higher level similarities - Powers of S**



Test errors over increasing number of layers using PAM clustering and different learners.

References

[1] H. Osborne, D. Bridge: Models of similarity for case-based reasoning. In: Interdisciplinary Workshop on Similarity and Categorization, pp. 173–179 (1997)

[2] R. Baeza-Yates, B. Ribeiro-Neto: Modern information Retrieval, ACM Press, New York (1999)

[3] R.C. Veltkamp, M. Hagedoorn: Shape similarity measures, properties and constructions. In VISUAL 2000: Advances in Visual Information Systems, LNCS, vol. 1929, Springer, Heidelberg (2000)

[4] A. Tversky: Features of similarity. Psychological Review 84(4), 327–352 (1977)

[5] S. Santini, R. Jain: Similarity measures. IEEE Transactions on Pattern Analysis and Machine Intelligence (1999)

[6] J. Orozco: Similarity and dissimilarity concepts in machine learning. Technical Report LSI-04-9-R, Universitat Politcnica de Catalunya, Barcelona, Spain (2004)

[7] G. Klir, B. Yuan: Fuzzy Sets and Fuzzy Logic: Theory and Applications. Pearson Education (1995)

[8] D. Lin: An information-theoretic definition of similarity. In: International Conference on Machine Learning, Madison, Wisconsin, USA (1998)

[9] Frank, A., Asuncion, A. UCI Machine Learning Repository `http://archive.ics.uci.edu/ml`. Irvine, CA: University of California, School of Information and Computer Science (2010)

[10] Kaburlassos, V., Petridis, V. Learning and decision-making in the framework of fuzzy lattices. New learning paradigms in soft computing, Physica-Verlag (2002)

[11] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford (1995)

[12] Omohundro, S. M. Geometric Learning Algorithms. Technical Report 89-041. Univ. of Berkeley, CA (1989)

[13] L. Prechelt. Proben1-A set of Neural Network Benchmark Problems and Benchmarking Rules. Fac. für Informatik. U. Karlsruhe T.R. 21/94 (1994)

[14] Everitt, B. Cluster Analysis. Heinemann Books (1977)

[15] J. L. Chandon, S. Pinson. *Analyse Typologique*. Masson (1981)

[16] V. Tresp, S. Ahmad, R. Neuneier. Training Neural Networks with Deficient Data. In NIPS 6, Morgan Kaufmann (1994)

[17] J.C. Gower. A General Coefficient of Similarity and Some of Its Properties. Biometrika, 27(4), pp. 857–871 (1971)

[18] J. Hartigan. Clustering Algorithms. Wiley (1975)

[19] R. Xu, D. Wunsch. Clustering. IEEE Press Series on Computational Intelligence. Wiley (2008)

[20] S. Haykin. Neural Networks, a Comprehensive Foundation. Prentice Hall (1999)

[21] T. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. Neural Computation 10(7), pp. 1895–1923 (1998)

[22] D. Wettschereck and T. Dietterich. Improving the Performance of Radial Basis Function Networks by Learning Center Locations. In Advances in Neural Information Processing Systems 4, pp. 1133–1140 (1992)

[23] E. Alpaydin. Combined 5$\times$2 cv F Test for Comparing Supervised Classification Learning Algorithms. Neural Computation 11(8), pp. 1885–1892 (1999)

[24] Zadeh, L. Fuzzy Sets as a basis for a theory of possibility. Fuzzy Sets and Systems, 1: 3-28 (1978)

[25] M. Orr. Introduction to Radial Basis Function Networks. Technical Report of the Institute for Adaptive and Neural Computation. Available at `http://www.anc.ed.ac.uk/rbf/papers/intro.ps.gz` (1996)

[26] L. Belanche, Learning in Networks of Similarity Processing Neurons. Workshop New Challenges in Neural Computation 2013.

[27] E. Pe kalska, Dissimilarity representations in pattern recognition. Concepts, theory and applications, Ph.D. thesis, Delft University of Technology, ASCI Dissertation Series, 109, The Netherlands, January 2005.

[28] R.P.W. Duin, E. Pe kalska, P. Paclik, D.M.J. Tax, The dissimilarity representation, a basis for domain based pattern recognition? Pattern representation and the future of pattern recognition, ICPR 2004 Workshop Proceedings, Cambridge, United Kingdom, 2004, pp. 4356.

[29] R.P.W. Duin, D. de Ridder, D.M.J. Tax, Featureless pattern classification, Kybernetika 34 (4) (1998) 399404.

[30] Edelman S and Shahbazi R (2012) Renewing the respect for similarity. Front. Comput. Neurosci. 6:45. doi: 10.3389/fncom.2012.00045

[31] S. Edelman, Representation and Recognition in Vision, MIT Press, Cambridge, 1999.

[32] Bache, K. and Lichman, M. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml. Irvine, CA: University of California (2013)

[33] Aamodt, A., Plaza, E. Case-based reasoning: Foundational issues, methodological variations and system approaches. *AI Communications*, 7(1):39-59, 1994.

[34] Hebb, D. *The organization of behaviour*, John Wiley, 1949.

[35] D.R. Wilson, T.R. Martinez. Heterogeneous Radial Basis Function Networks. In Procs. of the 6th Intl. Conf. on Neural Networks (ICANN'96), vol. 2, pp. 1263-1267, 1996.

[36] C. Stanfill, D. Waltz. Toward memory-based reasoning. *Communications of the ACM* 29, 1986.

[37] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford, 1995.

[38] Cherkassky, V., Mulier, F. Learning from data:concepts, theory and methods. Wiley, 1998.

[39] Ripley, B. Pattern Recognition and Neural Networks. Cambridge Univ. Press, (1996)

[40] De Baets, Bernard, and Hans De Meyer. ”Transitive approximation of fuzzy relations by alternating closures and openings.” Soft Computing 7.4 (2003): 210-219.

[41] Reynolds, A., Richards, G., de la Iglesia, B. and Rayward-Smith, V. (1992) Clustering rules: A comparison of partitioning and hierarchical clustering algorithms; Journal of Mathematical Modelling and Algorithms 5, 475504

[42] Robnik-ikonja, Marko, and Igor Kononenko. ”Theoretical and empirical analysis of ReliefF and RReliefF.” Machine learning 53.1-2 (2003): 23-69.

[43] Hall, Mark A. Correlation-based feature selection for machine learning. Diss. The University of Waikato, 1999.

[44] Reunanen, Juha. ”Overfitting in making comparisons between variable selection methods.” The Journal of Machine Learning Research 3 (2003): 1371-1382.

[45] Guyon, Isabelle, and Andr Elisseeff. ”An introduction to variable and feature selection.” The Journal of Machine Learning Research 3 (2003): 1157-1182.

[46] Vapnik, Vladimir. ”Statistical learning theory. 1998.” (1998).

[47] Ripley, B. D. ”Pattern recognition and neural networks, 1996.” Cambridge Uni. Press, Cambridge.