Summer 2017

Bringing interpretability and visualization with artificial neural networks

Andrey Gritsenko
University of Iowa

BRINGING INTERPRETABILITY AND VISUALIZATION WITH ARTIFICIAL NEURAL NETWORKS

by

Andrey Gritsenko

A thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Industrial Engineering
in the Graduate College of
The University of Iowa

August 2017

Thesis Supervisor: Associate Professor Amaury Lendasse

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

PH.D. THESIS

This is to certify that the Ph.D. thesis of

Andrey Gritsenko

has been approved by the Examining Committee for the thesis requirement for the Doctor of Philosophy degree in Industrial Engineering at the August 2017 graduation.

Thesis Committee: ______________________________
Amaury Lendasse, Thesis Supervisor

____
Stephen Baek

____
Andrew Kusiak

____
Albert Ratner

____
Edward Ratner

ACKNOWLEDGEMENTS

I would like to acknowledge my academic supervisor Assoc. Prof. Amaury Lendasse for his guidance and support during my studies, for sharing his knowledge and experience and also for keeping me in shape, both mentally and emotionally. It was my highest pleasure to work with this bright person, whom I can not only call my teacher, but a friend.

I would like to express my gratitude to all great researchers and scientists whom I had an honor to meet and work with: Anton Akusok, Edward Ratner, Stephen Baek, Emil Eirola, Du**v**san Sovilj, Yoan Miche, Guang-Bin Huang. Their example inspire me and serve as an incentive to improve myself to become a better scholar and human being.

I would like to separately thank the faculty and staff of the Department of Mechanical and Industrial Engineering, in particular Tara Hoadley for her assistance in all administrative processes and small talks we had over a cup of coffee, and Prof. Albert Ratner for his support and with whom it was always nice and fun to chat in my native language.

Last but not least, I would like to acknowledge my family and friends for all the love and support they provided during these years, helping me to bridge over the difficulties and keep moving towards my goal and my dream.

ABSTRACT

Extreme Learning Machine (ELM) is a training algorithm for Single-Layer Feed-forward Neural Network (SLFN). The difference in theory of ELM from other training algorithms is in the existence of explicitly-given solution due to the immutability of initialed weights. In practice, ELMs achieve performance similar to that of other state-of-the-art training techniques, while taking much less time to train a model. Experiments show that the speedup of training ELM is up to the 5 orders of magnitude comparing to standard Error Back-propagation algorithm.

ELM is a recently discovered technique that has proved its efficiency in classic regression and classification tasks, including multi-class cases. In this thesis, extensions of ELMs for non-typical for Artificial Neural Networks (ANNs) problems are presented. The first extension, described in the third chapter, allows to use ELMs to get probabilistic outputs for multi-class classification problems. The standard way of solving this type of problems is based 'majority vote' of classifier's raw outputs. This approach can rise issues if the penalty for misclassification is different for different classes. In this case, having probability outputs would be more useful. In the scope of this extension, two methods are proposed. Additionally, an alternative way of interpreting probabilistic outputs is proposed.

ELM method prove useful for non-linear dimensionality reduction and visualization, based on repetitive re-training and re-evaluation of model. The forth chapter introduces adaptations of ELM-based visualization for classification and regression tasks. A set of experiments has been conducted to prove that these adaptations provide better visualization results that can then be used for perform classification or regression on previously unseen samples.

Shape registration of 3D models with non-isometric distortion is an open problem in 3D Computer Graphics and Computational Geometry. The fifth chapter discusses a novel approach for solving this problem by introducing a similarity metric for spectral descriptors. Practically, this approach has been implemented in two methods. The first one utilizes Siamese Neural Network to embed original spectral descriptors into a lower dimensional metric space, for which the Euclidean distance provides a good measure of similarity. The second method uses Extreme Learning Machines to learn similarity metric directly for original spectral descriptors. Over a set of experiments, the consistency of the proposed approach for solving deformable registration problem has been proven.

PUBLIC ABSTRACT

The modern society is rightly called *information society* – the amount of information and data grows exponentially over the several past decades. And with development of computational technologies we become able not only to properly analyse and use that data, but teach computers to perform human-like tasks, e.g. face or speech recognition. Machine Learning is the field of study that made this possible by *"giving computers the ability to learn without being explicitly programmed"* (Arthur Samuel, 1959).

Algorithmically, for Machine Learning methods it doesn't matter for what task they are applied for, either it face recognition or stock market prediction: they represent all data as a set of *samples* that are described in terms of specific properties – *features*. Some of these features are considered as *targets* depending on the goal of the task, e.g. for face recognition task the target would be a person identification, while for speech recognition a word would be a target. The goal of any Machine Learning techniques, regardless of the human-level task, is to build a function of relationship between *features* and *targets*, learning from a set of *samples*. Different Machine Learning methods use different approaches to reach this goal. One of the method family is Artificial Neural Networks that copies a simplified model of the brain.

This thesis is devoted to the Extreme Learning Machines – a fast novel model in the family of Artificial Neural Networks, and its applications for several tasks that are still not well developed in the field of Machine Learning. One of applications is based on the method that allows to visualize data with big amount of features on a 2-D plane and proposes a way to improve visualization results by using additional target information.

The second extension consists in the obtaining of probabilistic estimations as a result of classification task. The idea of this application is derived from human recognition process: if we hear a word that we cannot recognize at the moment, we give assumptions, with certain level of probability, on what that word might be. Computers, on the other hand, will try to recognize it just by considering the highest similarity with known words. Two methods described in this thesis allow to perform Machine Learning classification task in more human-like way.

The third application bridges two fields of study – Machine Learning and Computer Graphics, more precisely, 3D Computer Graphics. This application is devoted to advance the accuracy of such tasks in Computer Graphics as, transfer attributes of one 3D model to another, reconstruct a 3D object from a set of different time-varying frames, interpolate shape of a 3D object, and all other tasks in one or another way related to the problem of shape registration.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF SYMBOLS

N	Number of data samples
d	Number of input features
c	Number of output features or classes
L	Number of hidden neurons
L^1	$\|\|x\|\|_1$ norm
L^2	$\|\|x\|\|_2$ norm
$\mathbf{X}_{[N \times d]}$	Input data in matrix form
$\mathbf{T}_{[N \times c]}$	Target (true) outputs in matrix form
$\hat{\mathbf{T}}_{[N \times c]}$	Predicted outputs in matrix form
$\mathbf{W}_{[d \times L]}$	Input-to-hidden layer projection matrix in ELM
$\mathbf{b}_{[1 \times L]}$	Bias vector in ELM
$\mathbf{H}_{[N \times L]}$	Hidden layer output of ELM in matrix form
$\boldsymbol{\beta}_{[L \times c]}$	Hidden-to-output layer projection matrix in ELM
$\mathbf{e}$	Vector of ones of a corresponding shape
$\phi()$	Activation function of a hidden neuron

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
ELM	Extreme Learning Machine
ELMVIS	Extreme Learning Machine-based Visualization
ELMVIS+C	Extreme Learning Machine-based Visualization for Classification data
ELMVIS+R	Extreme Learning Machine-based Visualization for Regression data
GMM	Gaussian Mixture Model
GPS	Global Point Signature spectral descriptor
HKS	Heat Kernel Signature spectral descriptor
LOO	Leave-One-Out cross-validation method
MDS	Multidimensional Scaling
MLP	Multi-Layer Perceptron
MNIST	Mixed National Institute of Standards and Technology dataset – popular benchmark dataset of handwritten digits
MSE	Mean Squared Error
PCA	Principal Component Analysis
PRESS	Predicted Residual Sum of Squares
RBF	Radial Basis Function
RMSE	Root Mean Squared Error
SLFN	Single Layer Feed-forward Neural network
SOM	Self-Organizing Maps
SVM	Support Vector Machine
UCI	A popular online source of benchmark data
WKS	Wave Kernel Signature spectral descriptor

CHAPTER 1
INTRODUCTION

1.1 Aims and Scope

The purpose of this thesis is to advance the state-of-the-art knowledge in Extreme Learning Machines (ELM) [CHK$^+$13], by developing new technologies based on this method. The ELM is a recently discovered way of training Single Layer Feed-forward Neural networks (SLFN) [Hay98], based on random initialization [HZS04]. The main advantage of the method is that in general it can achieve accuracy similar to that of a Multi-Layer Perceptron (MLP), using Error Back-Propagation for training, in a much smaller training time – up to the 5 orders of magnitude.

The second chapter of the thesis provides an introduction to the field of ELMs. It includes the description of standard algorithm, discussing basic methods to optimize the number of neurons in hidden layer and essential tips to counter the negative effects of random initialization.

The third chapter focuses on the non-typical task in the field of Artificial Neural Networks, and ELMs in particular – obtaining probability outputs for classification problems. The chapter consists of two sections describing two approaches developed by the author: one is based on the combination of ELMs and Gaussian Mixture Model (GMM), in the second approach probability outputs are calculated separately for each class, using the ratio between misclassified and correctly classified samples for corresponding class. The chapter also provides a brief discussion on alternative ways to interpret probabilistic outputs.

The fourth chapter of the thesis presents an extension of ELM for non-linear visualization – ELMVIS, and introduces two applications of this extension, described in corresponding sections. In these applications ELMVIS, which is unsupervised visualization method, is employed for classification and regression tasks data, using weighted supervised component to improve visualization results. The results of visualization are then used to perform corresponding tasks. Experiments presented in the chapter show the increase of accuracy.

1.2 Author's Contributions

Author made the following contributions to the three journal articles, four conference papers, one patent application and the thesis itself:

The second chapter summarizes the algorithm, solution and notations of an original ELM model, which are used throughout the thesis. All of the sections of the second chapter are based on prior works, but cannot be omitted due to their importance in the field of ELMs.

The third chapter of the thesis presents four original author's works. The paper [EGA+15] describes the use of combination of ELMs and GMM for probability outputs in multi-class classification problems and serves as a base for Section 3.1. Section 3.2 is based on two original works [GES+16, GES+17] presenting a novel model of classifier with probabilistic outputs, designed for multi-class problems. The result of author's contribution also include the participation in patent application [SRG15] for the mentioned method.

The fourth chapter of the thesis is based on two papers [GAM$^+$16, GAB$^+$17]. Both papers present applications of ELMs' extension for non-linear-based visualization for classification and regression problems. Papers provide experimental results proving the positive impact of supervised component on visulization results.

Additionally, the author appears as a co-author of one journal and one conference papers not included in this thesis [AEM$^+$17, AGM$^+$17].

1.3 List of Publications

Included in the thesis (journal papers)

[GAB$^+$17] Andrey Gritsenko, Anton Akusok, Stephen Baek, Yoan Miche, and Amaury Lendasse, *ELMVIS++R – Mastering Visualization with Target Variables*, Cognitive Computation (upcoming 2017), 1 – 22.

Included in the thesis (conference papers)

[EGA$^+$15] Emil Eirola, Andrey Gritsenko, Anton Akusok, Kaj-Mikael Björk, Yoan Miche, Duvsan Sovilj, Rui Nian, Bo He, and Amaury Lendasse, *Extreme Learning Machines for Multiclass Classification: Refining Predictions with Gaussian Mixture Models*, Advances in Computational Intelligence, Lecture Notes in Computer Science, vol. 9095, Springer International Publishing, 2015, pp. 153–164.

[GAM$^+$16] Andrey Gritsenko, Anton Akusok, Yoan Miche, Kaj-Mikael Björk, Stephen Baek, and Amaury Lendasse, *Combined Nonlinear Visualization and Classification: ELMVIS++C*, The 2016 International Joint Conference on Neural Networks (IJCNN), IEEE, July 2016, pp. 2617–2624.

[GES$^+$16] Andrey Gritsenko, Emil Eirola, Daniel Schupp, Edward Ratner, and Amaury Lendasse, *Probabilistic Methods for Multiclass Classification Problems*, Proceedings of ELM-2015 Volume 2, Proceedings in Adaptation, Learning and Optimization, vol. 7, Springer International Publishing, 2016, pp. 375–397.

[GES$^+$17] ______, *Solve Classification Tasks with Probabilities. Statistically-Modeled Outputs*, Hybrid Artificial Intelligent Systems (HAIS 2017), Lecture Notes in Artificial Intelligence, Springer International Publishing AG, June 2017, pp. 293–305.

[SGBL17] Zhiyu Sun, Andrey Gritsenko, Stephen Baek, and Amaury Lendasse, *Deep Spectral Descriptors: Learning the Point-Wise Correspondence Metric via Siamese Deep Neural Networks*, The 25th Pacific Conference on Computer Graphics and Applications (Pacific Graphics 2017), upcoming 2017.

Included in the thesis (patents)

[SRG15] Daniel Schupp, Edward Ratner, and Andrey Gritsenko, *U.S. Provisional Patent Application No. 7062320: Object categorization using statistically-modeled classifier outputs*, August 2015.

Not included in the thesis

[AEM$^+$17] Anton Akusok, Emil Eirola, Yoan Miche, Andrey Gritsenko, and Amaury Lendasse, *Advanced Query Strategies for Active Learning with Extreme Learning Machines*, The European Symposium on Artificial Neural Networks (ESANN), upcoming 2017.

[AGM$^+$17] Anton Akusok, Andrey Gritsenko, Yoan Miche, Kaj-Mikael Björk, Rui Nian, Paula Lauren, and Amaury Lendasse, *Adding Reliability to ELM Forecasts by Confidence Intervals*, Neurocomputing (upcoming 2017), 232–241.

CHAPTER 2
THEORY OF EXTREME LEARNING MACHINES

Extreme Learning Machines [HZS04, HZS06, HZDZ12, CHK$^+$13] (ELM) is a recently discovered Machine Learning technique that was successfully approved not only for Single-Layer Feed-forward Neural networks (SLFNs) [HZS04, HZDZ12, HCS06, Hua15, Hua14], but also for Multi-Layer Perceptrons (MLPs) [Hua15]. In contrast to the traditional learning algorithms and theories, e.g. learning through Back-Propagation [Hay98], ELM model doesn't use iterative approach for updating weights between the layers. Instead, the weights for the input layer and values of hidden neurons are assigned and calculated only once, accordingly, but nevertheless ELMs were proved to have universal approximation and classification properties [HCS06, Hua15, Hua14]. Though some attempts to build SLFNs with randomly generated parts have been taken before [Whi89, Whi06, PPS94], they lacked universal approximation property, and ELM is the first method with randomly assigned and not trained input weights prove universal approximation of any continuous function with almost any nonlinear and piecewise continuous hidden neurons [Hua15, Hua14]. The universal approximation property implies that an ELM can solve any regression problem with a desired accuracy, given that it has enough hidden neurons and training data to learn the parameters for all the hidden neurons.

The main difference between ELM and traditional training methods is the existence of explicit solution for the output weights. The existence of such non-iterative and, furthermore, linear solution becomes possible due to the independence of input and output weights. This property allows ELMs to be trained in up to 5 orders of magnitude less time compared to MLPs [Ros58], and up to 6 orders of magnitude less time compared to Support Vector Machines [CV95] (SVMs) [MvHB$^+$11, MSB$^+$10, MSL08].

Originally developed as a regression method [HZS04, HMZ$^+$06], ELM can be easily adapted to solve classification problems [HZDZ12], including cases of multiclass classification (the number of output nodes correspond to the number of classes, and the index of the output node with the highest output value indicates the predicted class of input).

To decrease possible negative impact of random initialization, regularization (L^1 [MBJ$^+$08, MSB$^+$10] and L^2 [MvB$^+$11]) and model structure selection [YME$^+$13, ZHC13, MSB$^+$10] techniques are successfully applied to Extreme Learning Machines.

The recent advance in the field allowed to use Extreme Learning Machines for Big Data problems, successfully employing both CPU and GPU acceleration [vML$^+$09, vMOL11, AGLM13, ZOT14, AMH$^+$14, AMK$^+$15, HBKV15, ABML15].

2.1 Basic Extreme Learning Machines

The ELM algorithm, originally proposed by Guang-Bin Huang *et al.* in [HZS04, HMZ+06, HCS06, HZS06], uses the Single Layer Feedforward Neural Network (SLFN) structure. The main concept behind the ELM lies in the random initialization of the SLFN input weights and biases. Then, under certain conditions, the synaptic input weights and biases do not need to be adjusted (classically through an iterative updates such as Error Back-Propagation) and it is possible to calculate both the hidden layer output matrix and, hence, the output weights implicitly. The complete network structure (weights and biases) is thus obtained with very few steps and very low computational cost comparing to iterative methods determine the weights.

The traditional ELM model is depicted in Figure 2.1. Bias element is conveniently included as an additional constant +1 input. Hidden layer weights $\mathbf{W}$ are fixed, and only output layer weights $\boldsymbol{\beta}$ are calculated.

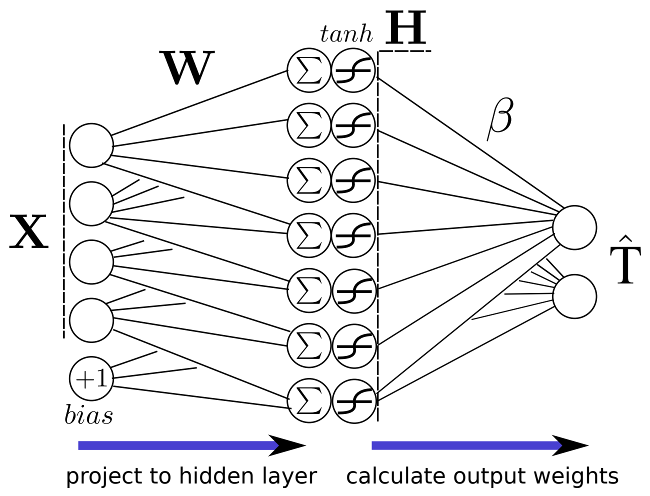


Figure 2.1: Extreme learning machine with multiple outputs

The method is proven to be an universal approximator given an enough number of hidden neurons [Hua14, Hua15]. It works as follows: Consider a set of N distinct samples $(\mathbf{x}_i, \mathbf{t}_i)$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $\mathbf{t}_i \in \mathbb{R}^c$. In case the ELM would perfectly approximate the data, it can be modeled as:

$$\sum_{i=1}^{L} \beta_i \phi(\mathbf{w}_i \mathbf{x}_j + b_i) = \mathbf{t}_j,\ j \in [\![1, N]\!]. \tag{2.1}$$

with ϕ being the activation function, $\mathbf{w}_i$ the input weights, b_i the biases and β_i the output weights.

In the case where the SLFN would perfectly approximate the data, the errors between the estimated outputs $\hat{\mathbf{y}}_i$ and the actual outputs $\mathbf{y}_i$ are zero and the relation between inputs, weights and outputs is then

$$\sum_{i=1}^{N} \beta_i \phi(\mathbf{w}_i \mathbf{x}_j + b_i) = \mathbf{y}_j, j \in [1, M], \tag{2.2}$$

which could be concisely written as $\mathbf{H}\beta = \mathbf{T}$, where

$$\mathbf{H} = \begin{pmatrix} \phi(\mathbf{w}_1 \mathbf{x}_1 + b_1) & \cdots & \phi(\mathbf{w}_L \mathbf{x}_1 + b_L) \\ \vdots & \ddots & \vdots \\ \phi(\mathbf{w}_1 \mathbf{x}_N + b_1) & \cdots & \phi(\mathbf{w}_L \mathbf{x}_N + b_L) \end{pmatrix}, \tag{2.3}$$

and $\beta = (\beta_1^T \ldots \beta_N^T)^T$ and $\mathbf{Y} = (\mathbf{y}_1^T \ldots \mathbf{y}_M^T)^T$.

Solving the output weights β from the hidden layer output matrix $\mathbf{H}$ and target values $\mathbf{T}$ is a linear regression problem. In the general case of $N \neq d$, the exact solution is achieved by the Moore-Penrose generalized inverse of the matrix $\mathbf{H}$ denoted as $\mathbf{H}^\dagger = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T$ [RM72, RM71].

Being originally developed as regression model, ELM can be easily adapted for classification tasks. In the following, a multi-class classification problem, as a general case, is analyzed. The data is a set of N distinct samples $\{\mathbf{x}_i, y_i\}$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{1, \ldots, c\}$ where c is the number of distinct classes. Encode classification targets as one binary variable for each class (one-hot encoding). $\mathbf{T}$ is the matrix of targets such that $T_{ij} = 1$ if and only if $y_i = j$, i.e., sample i belongs to class j. Otherwise, $T_{ij} = 0$. In the case of two classes, a single output variable is sufficient.

$$\mathbf{T}_{ij} = \begin{cases} 1 & \iff y_i = j \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

Matrix Form of ELMs

A single (hidden) layer feedforward neural network (SLFN) with d input nodes, c output nodes, and M neurons in the hidden layer can be written as

$$f(\mathbf{x}) = \sum_{k=1}^{M} \boldsymbol{\beta}_k h\left(\mathbf{w}_k \cdot \mathbf{x}\right), \tag{2.5}$$

where $\mathbf{w}_k$ are randomly assigned d-dimensional weight vectors, the output layer weights $\boldsymbol{\beta}_k$ are c-dimensional vectors, and $h(\cdot)$ an appropriate nonlinear activation function, e.g., the sigmoid function. The output of f is a c-dimensional vector, and class assignment is determined by which component is the largest.

In terms of matrices, the training of the network can be re-written as finding the least-squares solution to the matrix equation.

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T}, \quad \text{where} \quad H_{ik} = h\big(\mathbf{w}_k \cdot \mathbf{x}_i\big). \tag{2.6}$$

Constant bias terms are commonly included by appending a 1 to each $\mathbf{x}_i$ and concatenating a column of 1s to $\mathbf{H}$.

2.2 Optimizing Number of Neurons

The number of hidden neurons is the only tunable hyperparameter in an ELM model. It is selected using a Leave-One-Out (LOO) Cross-Validation error. The LOO method is usually a costly approach to optimize a parameter since it requires to train the model on the whole dataset but one sample, and evaluate on this sample repeatedly for all the samples of the dataset. However, the output layer is linear for the ELM model, and the LOO error has a closed form given by Allen's Prediction Sum of Squares (PRESS) [All74]. This closed form allows for fast computation of the LOO Mean Square Error, which gives an estimate of the generalization error of ELM. The optimal number of hidden neurons is found as the minimum of that Meas Squared Error.

The Allen's PRESS formula written with the multi-output notations of the paper is

$$\mathrm{MSE}_{\mathrm{LOO}}^{\mathrm{PRESS}} = \frac{1}{Nc} \sum_{n=1}^{N} \sum_{k=1}^{c} \left(\frac{\mathbf{T} - \mathbf{H}\mathbf{H}^{\dagger}\mathbf{T}}{\left[\mathbf{1}_N - \mathrm{diag}\left(\mathbf{H}\mathbf{H}^{\dagger}\right)\right] \mathbf{1}_c^T} \right)_{ik}^{2} , \tag{2.7}$$

where $\mathbf{H}^{\dagger}$ denotes the Moore-Penrose pseudo-inverse [RM71] of $\mathbf{H}$, and the division and square operations are applied element-wise. Additionally, for the particular implementation of PRESS optimization Tikhonov regularization [Tik95] was also used in the paper when computing the pseudo-inverse matrix of $\mathbf{H}$

$$\mathbf{H}^{\dagger} = \left(\mathbf{H}^T\mathbf{H} + \alpha\mathbf{I}\right)^{-1} \mathbf{H}^T , \tag{2.8}$$

where $\alpha = 10^{-5}$ is a chosen regularization parameter and $\mathbf{I}$ is an identity matrix. Tikhonov regularization is a well-known regularization technique for ill-posed problems [Bel78], which allows to avoid possible numerical and computational issues when finding the solution of the system described by Equation (2.6).

CHAPTER 3
EXTREME LEARNING MACHINES FOR PROBABILITY OUTPUTS

The Extreme Learning Machines (ELM) [HZS06, HCS06, ABML15, CHK$^+$13, MSB$^+$10, YME$^+$13, WSEL15, WEH15, WEH14, WHDE14] and Neural Networks in general, as well as other classification methods, have a successful history of being used to solve multi-class classification problems. The standard procedure is to convert the class labels into numerical 0/1 binary variables (or equivalently, $+1/-1$), effectively transforming the situation into a regression task. When a new sample is fed through the network to produce a result, the class is assigned based on which numerical value is the highest. While this leads to good performance in terms of classification accuracy and precision, the network outputs as such are not always meaningful.

Most classifiers based on neural networks provide results which can not directly be interpreted as probabilities. Probabilities are useful for understanding the confidence in classification, and evaluating the possibility of misclassification. In a multiclass problem, for instance, certain misclassification results may be considerably more harmful or expensive than others.

One example is in website filtering based on user-defined categories, where Artificial Neural Networks are used to classify previously uncategorized sites [QD09, PP12]. More reliable estimates of the risks involved are necessary for cloud security service provider to make informed (but automated) filtering decisions. Other such cases where the penalty for choosing the wrong class may vary greatly, include detecting malicious software activity [DSDY13, RTWH11, MAHN14], bankruptcy prediction [AVB$^+$14] and nuclear accident prediction [STL09].

It is true that the optimal least-squares estimator is equivalent to the conditional probability:

$$\hat{\mathbf{t}}(x) = \mathrm{E}[\mathbf{T} \mid x] = p(\mathbf{T}{=}1 \mid x). \tag{3.1}$$

In practice, however, the results can be outside the range 0–1, and this interpretation is not very reliable or directly useful.

Previous Works

Regardless of great benefit that could be obtained with probabilistic classification, only early work has been done. The most popular method is called Naive Bayes Classifier (NBC) [RSTK03, BMLR12, HY01, Zha04] and is based on Bayes' Theorem to compute conditional probabilities

$$p\left(C_k|x\right) = \frac{p\left(C_k\right) p\left(x|C_k\right)}{p\left(x\right)} \ , \tag{3.2}$$

where x is a sample vector of features, $p\left(C_k\right)$ is the *prior* probability of class C_k, $p\left(x|C_k\right)$ is the likelihood of x for a given class C_k, and $p\left(x\right) = \sum_k p\left(C_k\right) p\left(x|C_k\right)$ is the marginal probability of x. Though this theorem works well in probability theory, NBC cannot be considered as a reliable probability classifier as it assumes input variables are conditionally independent given the class label.

There are many measures to estimate the accuracy of non-probabilistic learning algorithms ([WCW74, HK06, LC98, AC92, MG63, OWN97]), while the lack of interest in probabilistic classification methods has resulted in a lack of accepted methods to evaluate probabilistic outputs. The most well known Mean Square Error estimation [BC01] is usually used to evaluate accuracy of probabilistic methods, though this approach may discard the meaning of probability output [EGA+15]. The primary method for assessing the accuracy of probability estimates is Brier score [Bri50].

Here for each example x, the square error (SE) is defined as

$$\mathrm{SE} = \sum_{i} \left(T\left(c_i \mid x\right) - P\left(c_i \mid x\right)\right)^2 \ , \tag{3.3}$$

where $P(c_i|x)$ is the probability estimated for example x and class c_i and $T(c_i|x)$ is defined to be 1 if the actual label of x is c and 0 otherwise. The drawback of this method, which is true for all MSE-based estimators, is that they heavily weight outliers. First of all, that means that large errors have a bigger impact on the estimation than small errors. Secondly, but what is more important in case of probabilities, is that a lot of small errors would influence the evaluation and not let us to estimate properly how far our prediction from the correct class.

Another approach to obtain probability estimates (also multiclass probability estimates) is to map classifier outputs into some mapping function. The main idea is to learn the mapping function using naive Bayes, based on the fact that Bayesian function ranks examples well: if $\hat{y}(x_1) < \hat{y}(x_2)$, then $P(c|x_1) < P(c|x_2)$, where $\hat{y}(x)$ – output of the classifier for a given sample x, $P(c|x)$ – probability for a sample x to belong to a class (in case of binary-class problem) [ZE01]. To learn a mapping function a subset of a training dataset is usually used. There exist three strategies to learn a mapping function: parametric method (learn parameters of the assumed function) [Pla99, Ben00], non-parametric – binning [ZE02], and isotonic regression [RWD88, ABE+55]. Originally proposed for binary classification, these strategies can be applied to multiclass classification if a multiclass problem is decomposed in several binary problems that are then solved separately. The decomposition from multiclass problem to a set of binary classification problems can be done by any appropriate method [HT98, ASSK00].

The following main drawbacks can be seen for this approach: parametric method requires knowledge of a mapping function to use, that is not always possible [Ben00, Pla99]; to choose the optimal number and boundaries of bins in non-parametric method cross-validation step is required, still, even applying cross-validation, for small and unbalanced datasets this method often fails to produce accurate probability estimates [ZE02]; in isotonic regression, for each interval classifier outputs are mapped into a different isotonic function, chosen from a set of non-decreasing functions.

In related work, the Sparse Bayesian Extreme Learning Machine [JCMPK14] presents another approach to use an ELM and obtain estimates of the posterior probability for each class. In the SBELM, the parameters of the ELM and the Bayesian inference mechanism are linked, and must be learned together through an iterative optimization scheme. This contrasts the currently proposed method, where the ELM and GMM layers are entirely decoupled, and can be trained separately.

Description of the Problem

Consider the following problem of multiclass classification of an object depicted on an image, where each object could belong to any of three classes: class 1 – ’Dog’,

Table 3.1: Possible outputs for multiclass classification problem

Sample	Class 1 (Dog)	Class 2 (Cat)	Class 3 (Bird)
Sample 1	99	1	0
Sample 2	49	51	0
Sample 3	33.3	33.3	33.4

class 2 – 'Cat', Class 3 – 'Bird'. Assume that for three different samples, the following output values have been received (see Table 3.1). Suppose that for each sample, output value $\hat{t_i}$ for the corresponding class i belongs to the interval $[0; 1]$ and $\sum_{i=1}^{3} \hat{t_i} = 1$. The discussion of what classification algorithm has been used to obtain these output values is out of the scope of this example.

When using standard classification estimators (e.g., mean square error estimator [MG63]) the class with the highest output value should be picked as a prediction. Though, in terms of the given example, the mentioned approach is meaningful only for the first data sample. For samples 2 and 3 it is more intuitively to use the terms of probability when discussing and estimating the corresponding outputs. With this it can be stated, that sample 2 most probably belongs to either class 1 or 2 with slightly better hand of class 2 and definitely does not belong to class 3; and sample 3 has a more or less equal probability of belonging to any of three classes, that can indicate, for example, that the sample belongs to a class of unseen during the training step data, or the sample possess features of all three classes.

Though it is more intuitively to comprehend and interpret probabilistic outputs, additional methods are required to convert raw outputs into probabilistic ones (discussed in Sections 3.1 and 3.2). Moreover, in order to evaluate probabilistic outputs of probabilistic classification methods new estimators should be proposed (Section 3.2.3). The advantage of a probability-based approach for classification is that even for a wrong prediction one can estimate *how far* the prediction was from the correct class (see Sections 3.1.2 and 3.2.4).

The next two sections contain the description of two methods designed to obtain probability outputs for classification tasks. The first method presents the combination of Extreme Learning Machines and Mixture of Gaussian Models. The second method is based on information on correct hits and misses during the training step to calculate the probability of belonging to a certain class, and can be used with any classifier, e.g. Extreme Learning Machines, Support Vector Machines, etc.

3.1 ELM-GMM

Mixtures of Gaussians can be used for a variety of applications by estimating the density of data samples [Bis06, ELVB14]. A Gaussian Mixture Model can approximate any distribution by fitting a number of components, each representing a multivariate normal distribution. See Figure 3.1 as an example.

The model is defined by its parameters, which consist of the mixing coefficients π_k, the means $\boldsymbol{\mu}_k$, and covariance matrices $\boldsymbol{\Sigma}_k$ for each component k $(1 \leq k \leq K)$ in a mixture of K components. The combination of parameters is represented as $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$.

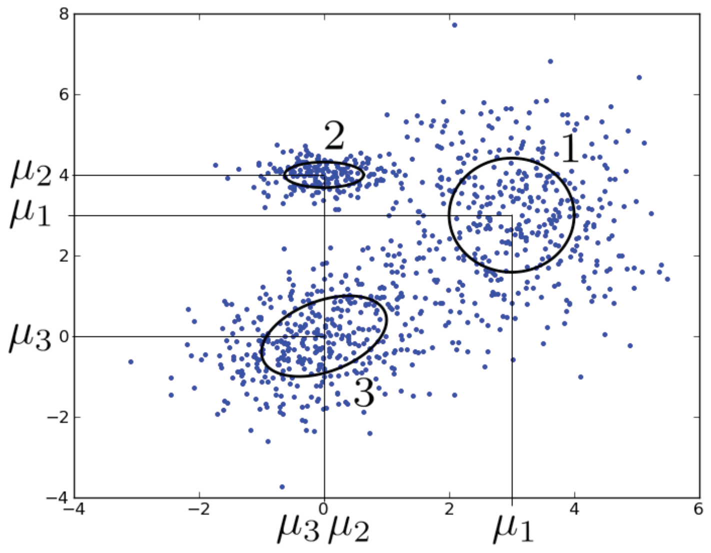


Figure 3.1: An example of 2D data with 3 Gaussian components after the convergence of GMM

The model specifies a distribution in $\mathbb{R}^d$, given by the probability density function

$$p(\mathbf{x} \mid \boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{3.4}$$

where $\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the probability density function of the multivariate normal distribution

$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right). \tag{3.5}$$

The standard procedure for fitting a Gaussian Mixture Model to a dataset is maximum likelihood estimation by the Expectation-Maximisation (EM) algorithm [DLR77, MK97, ELVB14]. The E-step and M-step are alternated until convergence is observed in the log-likelihood. Initialisation before the first E-step is arbitrary, but a common choice is to use the clustering algorithm K-means to find a reasonable initialisation [Bis06].

The only parameter to tune select is the number of components K. This can be done by separately fitting several models with different values for K, and using the BIC criterion [Sch78] to select the best model. In the proposed methodology, we are using the BIC criterion to select the value of K. Several further criteria are discussed in [MP00, Ch. 6].

Gaussian Mixture Models can be used to transform the values in the output layer to more interpretable probabilities. Specifically, this is accomplished by fitting the model to the training data and using it to calculate the probability of a sample belonging to a class, conditional on the output of the ELM. This procedure of refining the classification result of the ELM also leads to better classification accuracy and precision in some cases, as illustrated in the Section 3.1.2).

3.1.1 Methodology

The main idea of the proposed method is to first train a standard ELM for classification, and then apply a GMM to refine the results into more interpretable probabilities. This is accomplished by building a separate GMM for each class, on the ELM outputs of the samples from that class. If Y is the output of the ELM, the GMM is a model for the conditional distribution $p(Y \mid C)$ for each class. This leads to c separate GMMs. As the number of classes in data is less then number of features/variables, ELM could be intuitively considered as a dimensionality reduction technique that decreases number of training parameters N_{param} of GMM:

$$N_{param} = N_{comp}N_{dim} + \frac{N_{comp}N_{dim}\left(N_{dim}+1\right)}{2} + N_{comp} - 1\ , \tag{3.6}$$

where N_{dim} is the dimension of input data for GMM and N_{comp} is number of mixture components.

Given a new sample, prediction is conducted as follows: calculate the ELM output Y, and apply Bayes' theorem to find the posterior probability of each class C:

$$p(C \mid Y) = p(Y \mid C)\frac{p(C)}{p(Y)}.$$

Specifically:

$$p(C \mid Y) \propto p(Y \mid C)p(C),$$

where the proportionality constant is determined by the condition of adding up to 1.

The class priors $p(C)$ are given by the proportions in the training set (i.e., the maximum likelihood estimate).

The end result is now interpretable as a probability. A summary of the training and testing parts of the algorithm is presented in Algorithm 3.1.

The main requirement to fit a representative mixture model is that the number of observations should be greater than the number of parameters. That means, that aside from the fact that using ELM as data preprocessing reduces computational complexity of GMM algorithm, it also increases the possibility to basically building a GMM.

In the current situation, GMM is applied in conjunction with Bayes' theorem to find estimates for the posterior probabilities of each class for a sample. Specifically, it is used to estimate the term $p\left(x|C_k\right)$ in Equation (3.2). This requires that a separate GMM is built for each class.

To evaluate performances of this model for each sample, we consider the class with the highest conditional probability as the result of classification. A second criterion is presented and used in Section 3.1.2 in order to evaluate the quality and the applicability of the predicted probabilities.

Refine the Training for GMM

It is obvious that GMM built of the ELM outputs would inherit the error of the ELM model. To avoid this error accumulation, we are proposing to build GMM using only the correct classifications of the ELM. This training approach will be denoted by

Algorithm 3.1 Training the model and finding the conditional class probabilities for unseen data

```
▷ Training step
Require: Input data $\mathbf{X}$, targets $\mathbf{T}$
1: Randomly assign input vectors $\mathbf{w}_k$ and form $\mathbf{H}$
2: Calculate $\boldsymbol{\beta}$ as the least squares solution to eq. (2.6)
3: Calculate outputs on training data: $\mathbf{Y} = \mathbf{H}\boldsymbol{\beta}$
4: For each class $C$ do
5:     Fit a GMM$_C$ to the rows of $\mathbf{Y}$ corresponding to the class $C$
6: End for
7: Calculate $p(C)$ based on proportions of each class
8: Return $\mathbf{w}_k$, $\boldsymbol{\beta}$, GMM$_C$, $p(C)$

▷ Testing step
Require: Test data $\mathbf{X}_t$, weights $\mathbf{w}_k$, $\boldsymbol{\beta}$, GMM$_C$ and $p(C)$ for each class $C$
1: Form $\mathbf{H}_t$ by using the weights $\mathbf{w}_k$
2: Calculate outputs: $\mathbf{Y}_t = \mathbf{H}_t\boldsymbol{\beta}$
3: For each class $C$ do
4:     Use GMM$_C$ to calculate $p(Y_t \,|\, C)$ for each sample
5: End for
6: Calculate $p(C \,|\, Y_t) \propto p(Y_t \,|\, C)p(C)$ for each sample
7: Return Conditional probabilities $p(C \,|\, Y_t)$ for each class for each sample
```

suffix ‘r’ added to the corresponding GMMs. Compared to the algorithm presented in Algorithm 3.1, the only change is an additional step between steps 3 and 4 of the training phase: delete the rows of $\mathbf{Y}$ corresponding to misclassified samples. In Section 3.1.2, it is shown that this second approach is especially relevant when the original multiclass classification task is challenging.

Therefore, the training part of the algorithm could be summarized as the following algorithm:

Algorithm 3.2 Training the model on correctly classified samples

```
▷ Training step
Require: Input data $\mathbf{X}$, targets $\mathbf{T}$
1: Same as for original ELM-GMM model
2: Same as for original ELM-GMM model
3: Calculate correct outputs on training data: $\mathbf{Y}^k$
4: For each class $C$ do
5:     Fit a GMM$_C$ to the rows of $\mathbf{Y}^k$ corresponding to the class $C$
6: End for
7: Calculate $p(C)$ based on proportions of each class
8: Return $\mathbf{w}_k$, $\boldsymbol{\beta}$, GMM$_C$, $p(C)$
```

3.1.2 Experiments

Datasets

Six different datasets (three small datasets and three large ones) have been chosen for the approbation of the proposed combination of ELM and GMM: ELM-GMM and ELM-GMMr, comparing it with the original ELM. Datasets are collected from the University of California at Irvine (UCI) Machine Learning Repository [Lic13]. Table 3.2 summarizes the different attributes for the six datasets. All datasets have been preprocessed in the same way. At first, each dataset was subjected to random permutation. The result of permutation was then split in two parts under the ratio of 2:1, with two thirds of the points used as the training set and the remaining third used as the test set. For all datasets, training sets were normalized to zero mean and unit variance, and test sets were also normalized, using the same means and variances calculated and used for corresponding training sets. Proportions of classes in subsets have been kept balanced: each class in test set is represented in the same proportion, in which it is included in training set.

Each experiment has been repeated 1000 times, and the average performance is reported in Tables 3.3 and 3.4. Optimal value of hidden neurons in each repetition has been calculated using the PRESS Leave-One-Out Cross-Validation technique [All74, Mye90], with a maximum number of neurons equal to 300 based on the performance results obtained in [MvHB+11].

Table 3.2: Information about the selected datasets

			Samples	
Dataset	Variables	Classes	Train	Test
Wisconsin Breast Cancer	30	2	379	190
Pima Indians Diabetes	8	2	512	256
Wine	13	3	118	60
Image Segmentation	18	7	1540	770
First-Order Theorem Proving	51	6	4078	2040
Cardiotocography	21	10	1417	709

Results

Table 3.3 shows the test results for the three models and six datasets. Here "Wisc. B.C." stands for Wisconsin Breast Cancer dataset, "Pima I.D." – for Pima Indians Diabetes dataset, "Image Seg." – for Image Segmentation dataset, "F.-O. T.P." – for First-Order Theorem Proving dataset, and "Card." – for Cardiotocography dataset.

Comparing the accuracies of ELMs to the ones of the GMM variants, some datasets (Wine, Cardiotocography) are showing that the GMM is providing a clear improvement. In the other cases, the results are not notably different, but never statistically worse. The First-Order Theorem Proving dataset is the only situation where ELM-GMM performs clearly worse, but ELM-GMMr is again better than the original ELM. For all datasets, ELM-GMMr provides similar or better results than ELM-GMM.

Table 3.3: Performance of 3 different methods in terms of correct classification rates (with standard deviation in brackets)

	Wisc. B.C.	Pima I.D.	Wine	Image Seg.	F.-O. T.P.	Card.
ELM	95.05 (1.49)	70.90 (1.69)	93.00 (2.92)	93.67 (0.66)	52.34 (0.77)	73.63 (1.34)
ELM-GMM	95.00 (1.84)	70.40 (2.22)	94.00 (3.16)	93.96 (0.68)	50.42 (0.90)	76.62 (1.27)
ELM-GMMr	95.00 (1.49)	70.98 (1.51)	96.67 (2.81)	93.92 (0.64)	52.45 (0.75)	76.59 (1.29)

Reevaluate Performance of Probability Classification Methods

When calculating the performance of a probability-based classification method by just picking the class with the highest probability and treating it as a result of classification, we lose the advantage of the probability itself.

There are several possible solutions to take into account the predicted probabilities. One of the most simple solutions is to consider a classification to be correct if one of the two highest probabilities is for the correct class. If the predicted probabilities were not meaningful, the increase of performance measured by this second criterion would be limited. For example, in the Cardiotocography dataset with a total of 10 classes the improvement is close to 15%. The standard deviation is decreased. The correct class is nearly certainly one of the two most probable predicted classes. Eight classes are then certainly discarded. Similar considerations can be made for the First-Order Theorem Proving dataset, for which the improvement is even more significant, and the other examples.

Table 3.4 shows the resulting improvement in accuracy for multi-class datasets.

Table 3.4: Increase of the classification accuracy when considering top 2 labels

	Wine	Image Seg.	F.-O. T.P.	Card.
ELM-GMM	94.00 (3.16)	93.96 (0.68)	50.42 (0.90)	76.62 (1.27)
ELM-GMM 2	99.50 (0.81)	97.89 (0.43)	68.48 (0.85)	91.42 (0.86)
ELM-GMMr	96.67 (2.81)	93.92 (0.64)	52.45 (0.75)	76.59 (1.29)
ELM-GMMr 2	99.50 (0.81)	97.83 (0.39)	68.79 (0.89)	91.03 (0.91)

Conclusions

Including GMM as post-processing preserves the qualities of ELMs. Based on the results obtained on six datasets, it has been shown that the provided predicted probabilities are accurate, useful and robust.

The drawback of the given methodology is an increase of the over-fitting risk based on the fact that both ELM and GMM are trained on the same training sets. Furthermore, the optimal number of neurons for the original ELM is probably not optimal when the GMMs are added. In the future, selecting the optimal number of neuron for the proposed global methodology will be rigorously investigated.

As it can be clearly seen from the results presented in Section 3.1.2, reevaluating performance is imperfect, and there are needs to develop a better criterion to evaluate the quality and the advantages of dealing with probability outputs.

3.2 Histogram Probability Method

3.2.1 Methodology

In broad terms, the histogram method is a simple characterization of the continuous result from a classification algorithm. For example, in the case of an SVM, the distance to the separating hyperplane can be considered to be the projection of the data onto a decision plane. By binning a set of test data, a series of histograms can be constructed that allows for more robust analysis of the solution space [SRG15]. The physical interpretation for the SVM is straightforward, thanks to the definition of the separating hyperplane:

$$w = \sum_{i=1}^{l} y_i a_i \phi(x_i) \ . \tag{3.7}$$

In short, the distance is defined as the dot product between the normal to the hyperplane and the transformed feature vector.

The physical interpretation is less obvious in the case of the ELM. It is still not entirely unintuitive, because the ELM output, soft bounded to the set [0:1] and obeying a relationship described by activation function of neurons (e.g., *tanh* function), could be expected to also exhibit a distribution-like behavior. This was empirically verified in the experiments to follow.

In short, the proposed approach is to take advantage of a continuous decision function (as opposed to a binary decision maker). The proposed approach then bins the resulting decision space, and uses training data to build a model of the class distributions [SRG15]. Bayesian theory can then be applied for robust decision making.

At first, some classification algorithm should be trained in order to obtain outputs (for the sake of unambiguousness, called raw outputs) that would serve as a basis for probabilistic outputs for a certain multi-class problem. In order to build histograms, the range of raw training set outputs is computed as $\left[\min \hat{\mathbf{T}}_{train}; \max \hat{\mathbf{T}}_{train}\right]$. The range is then divided into bins, and a certain number of bins can differ for different classification problems with respect to their complexity. It should be noted that too small number of bins would result in loosing features of distribution, while too large number could be a reason for sparse histograms.

Then, for each class two types of histograms are built. At the first stage, samples in training set are split up into different sets according to the corresponding correct class. Then for each sample i in separated sets the raw output value of the correct class is added to the set of IN-class values, output values for other classes are added to the set of OUT-class values.

$$\hat{T}_{ij} \mapsto \begin{cases} IN_{C_k} & \text{if } j = k \\ \\ OUT_{C_k} & \text{if } j \neq k \end{cases}, \ k = correct\,class\,(i) \ , \tag{3.8}$$

where $\hat{T}_{ij}$ – is the raw output value of sample i for the corresponding class C_j, k identifies correct class for sample i, IN_{C_k} and OUT_{C_k} are respectively sets of IN-class and OUT-class values for class C_k.

After all, samples are processed, histograms of IN-class and OUT-class values are constructed on the selected bins. Obviously, number of values used to build OUT-class histogram is bigger than number of raw output values used to build IN-class histogram for the same class. Moreover, for real-world multi-class classification problems, number of samples in training set for different classes could be different.

In order to compensate these differences, histograms are normalized in the following way

$$\overline{\mathrm{IN}}_{C_i} = \frac{\mathrm{IN}_{C_i}}{size\left(C_i\right)} \tag{3.9}$$

and

$$\overline{\mathrm{OUT}}_{C_i} = \frac{\mathrm{OUT}_{C_i}}{\sum_{j=1}^{N_{classes}} size\left(C_j\right)}\ , j \neq i\ , \tag{3.10}$$

where IN_{C_i} and OUT_{C_i} are respectively IN-class and OUT-class histograms built for class C_i, $size\left(C_j\right)$ represents the number of samples in the training set, whose correct class is class C_j, $N_{classes}$ is number of total classes in a certain multi-class classification problem. For each class, resulting histograms can be considered as histograms of probability density functions that for a certain output value define the probability of either correct (in case of IN-class histogram) or wrong (in case of OUT-class histogram) classification, if that output value would be picked as the resulting value (the highest output value) for a sample.

With the respect to the used classification algorithm, probabilistic output for a given sample i, which has a raw output value $\hat{T}_{ij}$ for j class respectively, is computed as follows

$$p\left(C_j \mid \hat{T}_{ij}\right) = \frac{\overline{\mathrm{IN}}_{C_j}\left(\hat{T}_{ij}\right)}{\overline{\mathrm{IN}}_{C_j}\left(\hat{T}_{ij}\right) + \overline{\mathrm{OUT}}_{C_j}\left(\hat{T}_{ij}\right)}\ . \tag{3.11}$$

Equation (3.11) does not guarantee that for each sample probabilities would sum up to 1. The main reason is that for each class these probabilities are calculated regardless of information about other classes. In order to treat results of Equation (3.11) as probabilities for each sample they should be normalized.

3.2.2 Implementation details by example of the Iris dataset

A detailed explanation of the described probabilistic classification method is presented using as an example the *Iris* dataset from the University of California at Irvine (UCI) Machine Learning Repository [Lic13]. Assume, for a particular ELM implementation a certain output values were obtained. All of the output values are in the interval $[-0.3177; 1.3322]$. Again, this interval solely depends on the accuracy of the model and can differ for different experiments, even when the same input data and the same number of neurons in the hidden layer are used, e.g. in the current example the accuracy of ELM algorithm was 94%, which can be noticed by well-separated IN- and OUT-class histograms. Also, SVM algorithm was implemented, whose output was used to build histograms as well. Output values for the current SVM method implementation that represent the distance from data samples to a hyperplane are disposed in the range of $[-4.4169; 6.0817]$.

These intervals were divided in 20 bins each to build histograms of IN-class and OUT-class values for each class (Figure 3.2). Hereafter, IN-class histograms are represented in blue color and OUT-class histograms are represented in red color. The number of bins was chosen empirically so that on one hand, the features of distribution would retain, and on the other hand, the histograms itself would not be sparse. The same number of bins was used for other datasets mentioned in this paper.

According to the methodology, value of a particular bin for both IN-class and OUT-class histograms can be zero, if for all samples there was no one output in a specified range. This usually happens only for well separated classes, while in practice

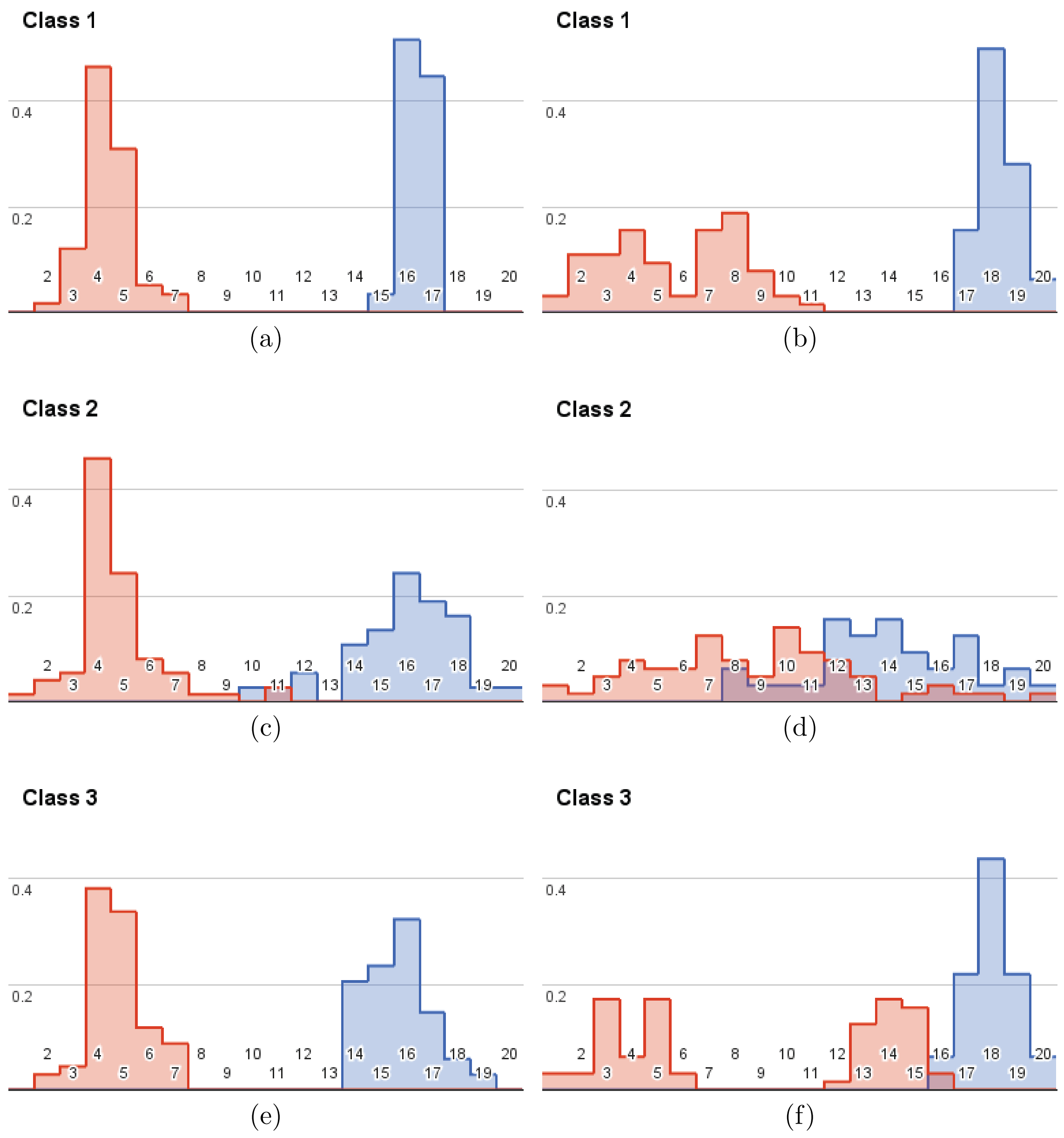
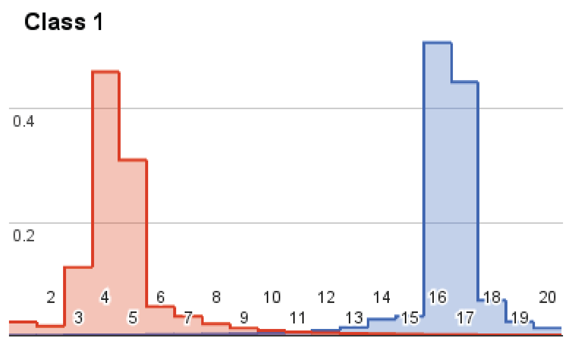
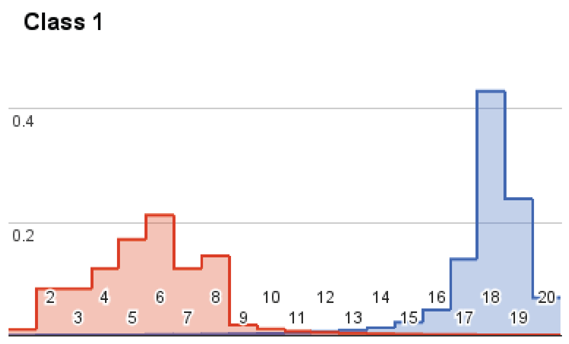


Figure 3.2: IN-class and OUT-class histograms for each class, built based on ELM outputs (Figures 3.2a, 3.2c and 3.2e) and SVM outputs (Figures 3.2b, 3.2d and 3.2f)

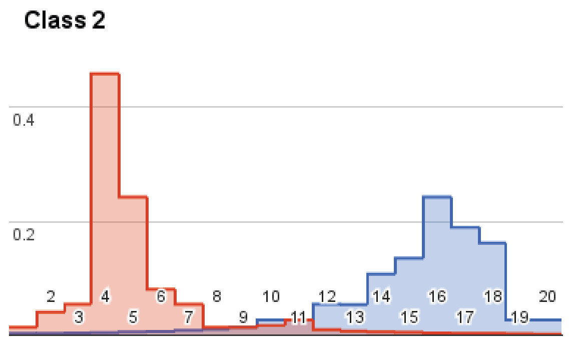
another situation is more frequent (see Figures 3.2b to 3.2e), when either IT-class or OUT-class values equal to 0.



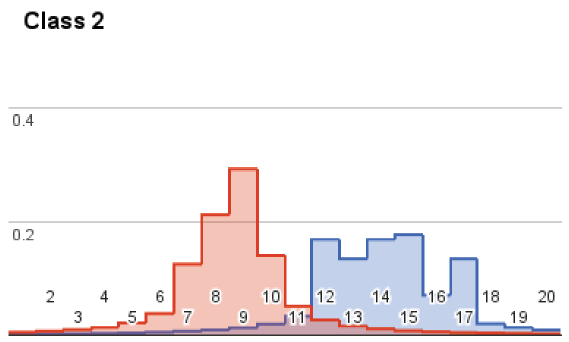
(a)



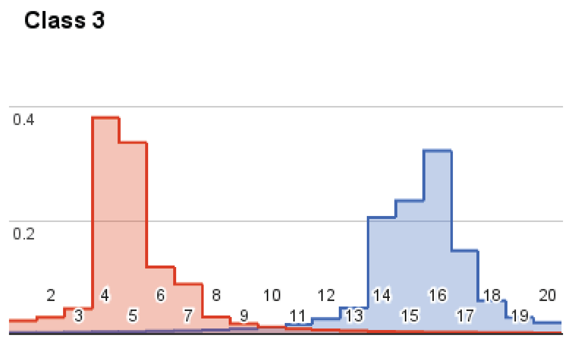
(b)



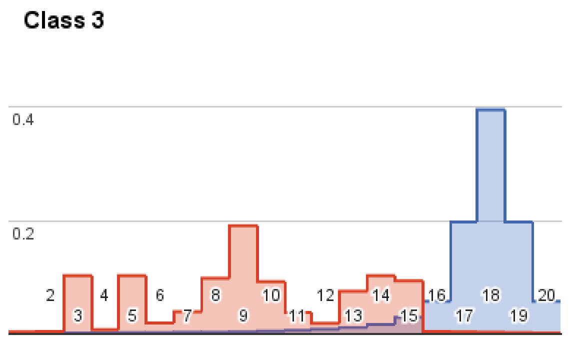
(c)



(d)



(e)



(f)

Figure 3.3: IN-class and OUT-class histograms after modification using Cauchy distribution for ELM-based outputs (Figures 3.3a, 3.3c and 3.3e) and SVM-based outputs (Figures 3.3b, 3.3d and 3.3f)

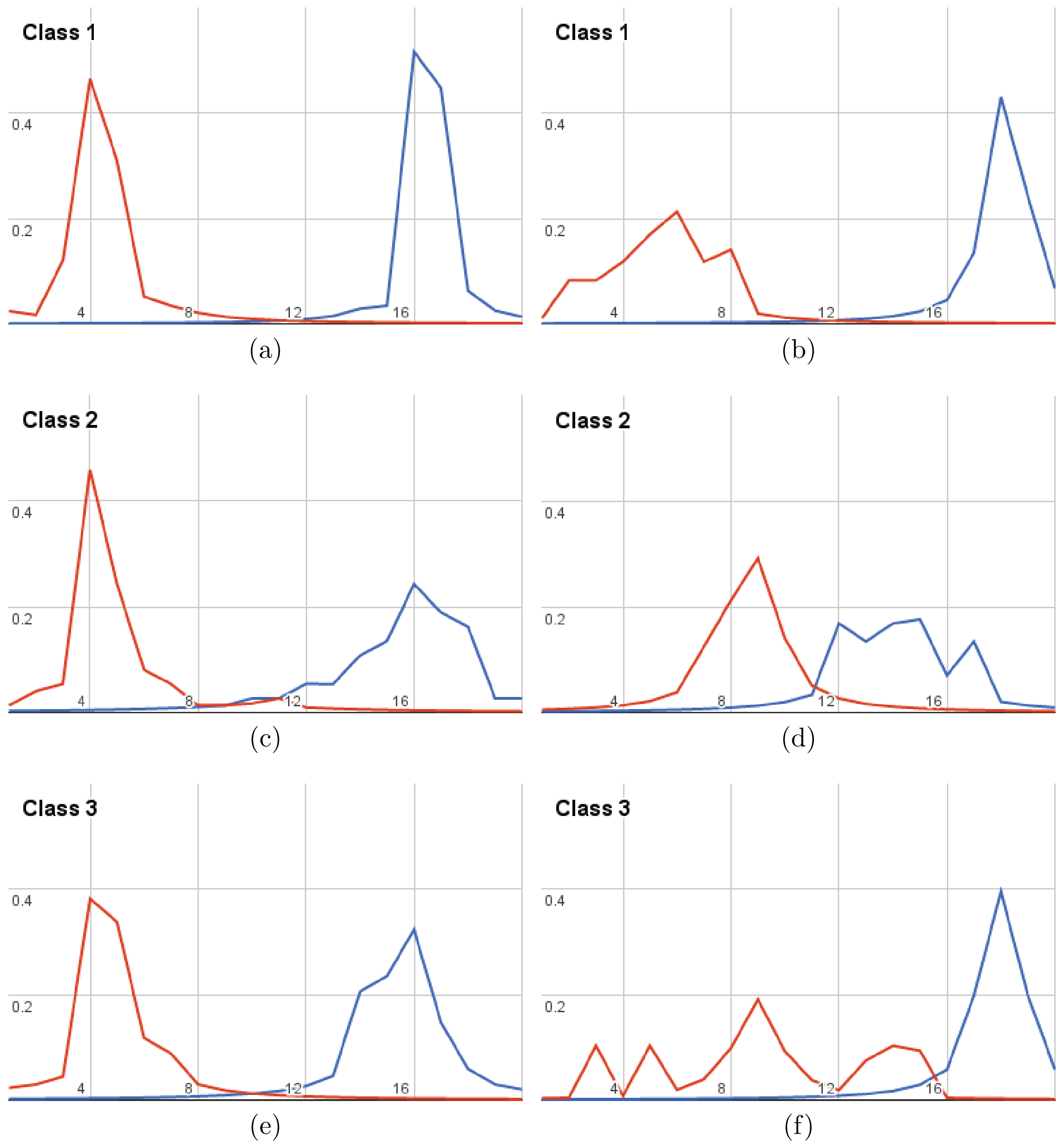


Figure 3.4: IN-class and OUT-class histograms after using Lagrange interpolation for ELM-based outputs (Figures 3.4a, 3.4c and 3.4e) and SVM-based outputs (Figures 3.4b, 3.4d and 3.4f)

By definition, IN-class and OUT-class histograms are built based on training data. Though, there is no guarantee that for test data there also wouldn't be any output values fitting to the zero bin. The worst case is when the output value hits IN-class zero bin, which would automatically assign zero probability for the corresponding value. In order to avoid such unreliable results and improve the overall reliability of the method, it was proposed to use the Cauchy distribution to fill the values of zero bins of the corresponding histograms (Figure 3.3). For each such zero-value bin a value of probability density function (pdf) of corresponding Cauchy distribution is calculated

$$f\left(x, x_0, \gamma\right) = \frac{1}{\pi\gamma\left[1+\left(\frac{x-x_0}{\gamma}\right)^2\right]} = \frac{1}{\pi\gamma}\frac{\gamma^2}{\left(x-x_0\right)^2+\gamma^2}\ , \tag{3.12}$$

where argument for Cauchy distribution PDF x – is the number of the corresponding bin, parameters of Cauchy distribution *location* x_0 and *scale* γ – respectfully mean value and standard deviation of the histogram distribution for the current class. The choice of Cauchy distribution was made because it fits and retains features of histogram values distribution better that other distributions (Gaussian, Poisson, Student's t-distribution).

According to the Equation (3.11) probability of the certain class is calculated regardless of the output value until it belongs to a certain bin. That means that for two values, one of which is located in the center of a bin and other is close to its edge, probability would be the same. Obviously, that is a very rough estimate, especially for standalone bins with high values (fig. 3.2a: bins 3,4,17 and 18, fig. 3.2b: bins 18 and 19; fig. 3.2e: bins 3 and 18, and fig. 3.2f: bins 5 and 13). In order to neglect this feature and make probability distribution more smooth the Lagrange interpolating polynomial [DB03] was used (Figure 3.4). Here, for any output value x of base classification method, in-class and out-class IN-class and OUT-values are computed using Lagrange interpolating polynomial for two interpolation points that are chosen as centers of two closest bins

$$L\left(x\right) = \sum_{k=0}^{1} L_k\left(x\right) y_k = \frac{x - x_1}{x_0 - x_1} y_0 + \frac{x - x_0}{x_1 - x_0} y_1 \ . \tag{3.13}$$

For output values x, which are smaller than value of the center of the first bin and greater than value of the center of the last bin, IN-class and OUT-class values are equated to values of the centers of corresponding bins. Two-point version of Lagrange interpolating polynomial has been chosen because it is the simplest interpolating algorithm that is very easy to implement, still it provides sufficient improvement in resulting performance of the method.

Algorithm 3.3 presents the overall process of finding probabilistic outputs for a certain multiclass classification problem using Histogram Probability method, including refining methods mentioned above.

3.2.3 Characterization of Probability Output and Its Interpretation

The most common way of interpreting output of any multiclass classification method is to compute accuracy - treat a class with the highest output value as the predicted class and compute how often the predicted class match the correct class. The other way is to use different types of estimators, e.g. mean square error estimator [MG63, Bri50, OWN97], to assess the error between predicted and correct classes.

Algorithm 3.3 Finding unseen data class probabilities based on trained classification model

```
▷ Training step
Require: Targets $\mathbf{T}$, raw output values $\hat{\mathbf{T}}$ for both training and test sets
 1: Compute the range of raw output values for training set $\left[\min \hat{\mathbf{T}}_{train}; \max \hat{\mathbf{T}}_{train}\right]$
 2: Form sets of $IN$-class and $OUT$-class values from raw outputs (eq. (3.8))
 3: For each class $C$ do
 4:     Build IN-class and OUT-class histograms based on corresponding sets of raw output values
 5:     Replace histogram values in 0-value bins using Cauchy distribution (eq. (3.12))
 6:     Use Lagrange two-point interpolating polynomial to switch from probabilities' histograms (eq. (3.13))
 7:     Normalize IN-class and OUT-class histograms using eqs. (3.9) and (3.10)
 8: End for
 9: Calculate $p\left(C_i \mid \hat{T}_{ij}\right)$ for each class for each sample in test set according to eq. (3.11)
10: Return Probabilities $p\left(C_i \mid \hat{T}_{ij}\right)$ normalized for each sample
```

Still these evaluation methods either do not use features of probabilistic outputs or have sufficient drawbacks, e.g. MSE-based estimators heavily weight outliers.

The proposed approach is to evaluate *how far* the correct class ends up from the predicted class with the highest probability output by measuring its *score* s

$$s = (\hat{p}_{max} - \hat{p}_{correct\,class}) \ , \tag{3.14}$$

where $\hat{p}_{max}$ and $\hat{p}_{correct\,class}$ are respectively the highest output value in the output vector in terms of probability and the output value for the correct class, and *rank* r. In order to have an equivalent measurements of rank for problems with different number of classes, rank values are scaled into $[0;1)$ interval

$$r = \frac{rank_{correct\,class} - 1}{N_{classes}} \ , \tag{3.15}$$

where $rank_{correct\,class}$ is the rank of the correct class in a descending-ordered output vector, $N_{classes}$ – number of classes in multiclass classification problem; by definition, score values are scaled into $[0;1]$ interval. To evaluate the performance of a multi-class probabilistic classification method average score S and rank R are used

$$S = \frac{\sum_{i=1}^{N_{samples}} s_i}{N_{samples}} \ , \tag{3.16}$$

$$R = \frac{\sum_{i=1}^{N_{samples}} r_i}{N_{samples}} \ . \tag{3.17}$$

3.2.4 Experiments

In the following subsections, presented earlier probabilistic classification methodologies are compared on a number of multiclass datasets. These compared methods include the original ELM and SVM algorithms, the two variants of the proposed combination of ELM and GMM: ELM-GMM and ELM-GMMr, and also combination of the proposed Histogram Probability method with both ELM and SVM.

The comparison was performed in two ways: at first(see Table 3.6), on training time and standard definition of accuracy (where the prediction is considered as correct if a class with the highest output value is the same as a real class, otherwise incorrect). Second (see Table 3.7), in a way presented in Section 3.2.3 to process probabilistic outputs, so only ELM-GMM, ELM-GMMr, ELM-HP and SVM-HP algorithms are compared.

The current SVM implementation is built on top of scikit-learn, which is in turn built on LIBSVM [FGA$^+$11]. The SVM is a standard C-SVC. The C value is optimized by training several times with different values, and selecting candidate values via a bisection search. The bisection search experimentally performs as well as an exhaustive search of the C values, but decreases training time by 80%. The C-scans indicate that the accuracy-vs-C curve is semi-convex and semi-smooth for the region we test over, which lends itself to such a search without sacrificing accuracy (see Figure 3.5, X-axis indicates the C-value and Y-axis represents the accuracy of SVM model for the corresponding C-value).

For each experiment, the number of neurons for ELM-based methods and C-value for SVM-based methods are chosen during the cross-validation phase. ELM-based methods perform the PRESS Leave-One-Out cross-validation technique [All74, Mye90]. Time spent for cross-validation phase has been included in the training time.

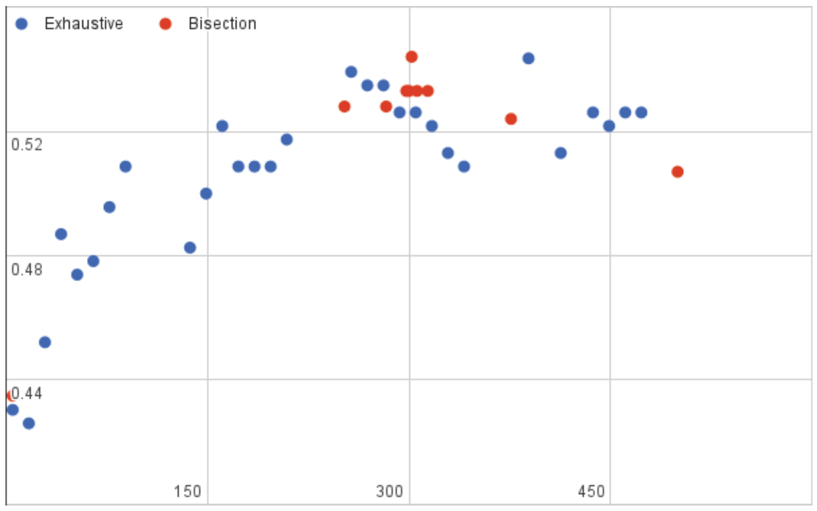


Figure 3.5: Bisection and exhaustive search of the C-values, by the example of *HOG-180* dataset

Gaussian Mixture Model was implemented using built-in OpenCV functions realizing Expected Maximization (EM) algorithm [Bil98]. For each class, Gaussian Mixture Model was built with the number of mixture components equal to 2 for each model. Amount of mixture components was chosen empirically as the number, for which the accuracy of ELM-GMM method has the smallest error.

Datasets

Ten different datasets have been used for the experiments. Results for the *Iris* dataset were also included in the resulting Tables 3.6 and 3.7 because it was used as an example dataset for the detailed explanation of Histogram Probability method. The next four datasets have already been used to compare results of ELM-GMM probabilistic classification methodology in [EGA+15], and for this paper have been mainly chosen to provide an overall comparison of two different probabilistic classification techniques. These datasets have been collected from the University of California at Irvine Machine Learning Repository [Lic13]. The last five datasets, whose detailed description is presented hereafter, are industry-driven datasets based on Caltech-256 Object Category Dataset – a set of 256 object categories containing a total of 30607 images [GHP07].

A subset of the Caltech-256 database is evaluated by generating feature vectors from the images. For the purposes of the tests, there were several well-known feature types chosen for review.

SURF features [BTVG06] were also computed‘ (referred to as *PHOW* in the literature below). The scale invariant feature transform has become a staple of image processing techniques. The features were clustered using a standard bag of words (BoW) technique. The size of the feature vectors refers to the number of clusters allowed by the BoW.

Color features are simply a computation of the color histogram of the object in question. These features are highly variable, and tend to be either extremely good predictors or entirely poor ones. In real-world applications, these are useful only when combined with other approaches to refine the classification.

The *Gabor* features [FS78] were generated using the OpenCV built-in functions. The *gabor* wavelet can be summarized as the combination of a plane wave and a Gaussian, which allows for both frequency and spacial localization.

HOG, or histogram of oriented gradients, is another commonly used feature type [DT05] in the field of image processing. This feature is another macro-shape descriptor, meaning that it describes the overarching structure of the object by constructing histograms of the weights due to different oriented gradients (i.e. the gradient computed in different directions).

Table 3.5: Information about used datasets

Dataset	# of variables	# of classes	Balanced classes	Samples Train	Samples Test	Neurons Max	Neurons Mean
Iris	4	3	Yes	100	50	50	17
Wine	13	3	No	118	60	64	35
Cardiotocography (CTG)	21	10	No	1417	709	700	449
Image Segmentation (IS)	18	7	Yes	1540	770	700	496
First-Order Theorem Proving (FOTP)	51	6	No	4078	2040	1000	765
Caltech-256 PHOW feature	50	10	Yes	600	300	200	72
Caltech-256 Color feature	3600	10	Yes	600	300	200	117
Caltech-256 Gabor feature	2500	10	Yes	600	300	200	88
Caltech-256 HOG180 feature	1700	10	Yes	600	300	200	129
Caltech-256 HOG360 feature	1700	10	Yes	600	300	200	129

Table 3.5 summarizes the different attributes for all ten datasets. The datasets have been preprocessed in the same way: two thirds of randomly permuted points are used to create the training set and the remaining third is used as the test set. Seven of ten datasets have balanced classes, meaning that all classes are represented by the same number of data samples. Only *Wine, First-Order Theorem Proving* and *Cardiotocography* datasets have unbalanced classes. For all datasets, proportions of classes have been preserved for both training and test sets. For all datasets, the training set is standardized to zero mean and unit variance, and the test set is also

standardized using the same mean and variance calculated and used for the training set. Because test sets are standardized using the same parameters as for the corresponding training sets, it is less likely for them to have exactly zero means and unit variances.

Experimental Procedure

In order to have statistically stable results, experiments for ELM-based methods were repeated 1000 times, and average performance has been calculated. Moreover, Monte Carlo method has been applied to run experiments: for a new experiment each dataset was subjected to random permutation and then split into training and test sets in the ratio of 2:1 with the same proportion of classes in each subset as in the original data.

Instead of exhaustive search of optimal C-value for SVM, bisection search was applied in order to reduce the training time. This approach works well for SVM algorithm and able to find global optima as the dependence function of SVM performance from the C-value is smooth (see Figure 3.5).

As it was already mentioned earlier, to define the optimal number of neurons for ELM method the PRESS LOO cross-validation [Mye90] has been used, performed for each 5 neurons starting from 5 neurons.

The maximum number of neurons in ELM varies for different datasets depending on the number of training data samples N and was chosen as not exceeding half of this number based on the performance results obtained in [MvHB$^+$11] and knowledge about the datasets themselves. Table 3.5 has the information on the maximum number of neurons chosen for each dataset and the rounded mean number of neurons that have been chosen in 1000 experiments by the described above LOO-based parameter selecting method.

In [EGA$^+$15] a method to refine the training of GMM was presented. The suggestion was to use as inputs for GMM only correct predictions outputs, because GMM trained on wrong predictions inherit the error. For datasets, for which ELM made predictions with high accuracy, the proposed method (displayed as ELM-GMMr in Tables 3.6 and 3.7) shows better results in comparison to GMM trained on ELM outputs for both correct and wrong predictions. In this paper, both general and refined versions of ELM-GMM probabilistic classification algorithm are compared also in the framework of datasets, for which ELM shows low performance.

Results

Tables 3.6 and 3.7 contain results of performing probabilistic classification for ten different datasets. Results for ELM and SVM algorithms are also included in the Table 3.6 as baseline to compare the accuracy of presented probabilistic classification methods in a standard way (using MSE [BC01, OWN97]).

From the Table 3.6 it can be concluded that on average Histogram Probability method has the same or very close both accuracy and running time, when based on ELM outputs. Other probabilistic methods, GMM and GMMr, show slightly worse accuracy (up to 5%) for Caltech-256 datasets, while approximately the same

Table 3.6: Comparison of the accuracy (in percents) and training time (in seconds) for presented methods

	ELM		ELM-GMM		ELM-GMMr		ELM-HP		SVM		SVM-HP	
Dataset	Time	Acc.	Time	Acc.	Time	Acc.	Time	Acc.	Time	Acc.	Time	Acc.
Iris	0.02	94.02	0.05	94.64	0.05	94.02	0.02	94.64	0.87	98.00	0.87	97.00
Wine	0.06	94.23	0.11	92.83	0.11	93.83	0.06	95.77	0.61	96.75	0.61	98.39
CTG	27.63	75.01	48.58	76.03	48.57	72.18	27.69	74.99	46.19	52.97	46.19	57.45
IS	20.67	95.06	44.24	94.47	22.09	94.82	20.74	95.03	12.21	84.64	12.21	80.86
FOTP	95.12	54.10	109.92	52.09	109.91	53.38	95.32	53.54	600.43	40.41	600.43	26.66
PHOW	17.46	30.43	17.55	25.27	17.48	25.97	17.49	29.40	37.12	40.40	37.12	44.75
Color	18.44	23.04	18.51	20.63	18.46	21.13	18.58	21.37	849.15	29.69	849.15	26.25
Gabor	17.73	17.25	17.79	13.17	17.79	13.56	17.81	17.57	164.69	24.38	164.69	24.69
HOG180	17.37	30.61	17.59	26.43	17.46	26.60	17.67	28.87	1460.17	51.88	1460.17	53.44
HOG360	17.39	30.20	17.51	26.47	17.47	27.29	17.53	29.17	1373.22	51.56	1373.22	50.00

for other datasets. In terms of computational time, ELM-GMM and ELM-GMMr take significantly longer time only for *Iris*, *Wine* and *Cardiotocography* datasets. It also can be stated that computational time of SVM-HP method is basically the same as of SVM method, though accuracy may rise (as for *Wine*, *Cardiotocography*, *PHOW* and both of *HOG* datasets), stay almost the same (as for *Gabor* dataset), or low (for other datasets).

Comparing ELM-based and SVM-based methods it can be noticed that for 7 out of 10 datasets (except of *Cardiotocography*, *ImageSegmentation* and *First-OrderTheoremProving*) SVM methods show higher accuracy (up to almost 25%, as for *HOG-180* and *HOG-360* datasets), though they also show much higher computational time.

From the results presented in Table 3.7, it can be implied that combination

Table 3.7: Evaluation of probabilistic methods in terms of 'rank-score' estimation

	ELM-GMM		ELM-GMMr		ELM-HP		SVM-HP	
Dataset	*Rank*	*Score*	*Rank*	*Score*	*Rank*	*Score*	*Rank*	*Score*
Iris	0.023	0.054	0.027	0.066	0.023	0.031	0.010	0.009
Wine	0.037	0.082	0.034	0.073	0.026	0.028	0.015	0.011
CTG	0.044	0.209	0.055	0.248	0.063	0.109	0.122	0.111
IS	0.009	0.044	0.008	0.041	0.012	0.022	0.051	0.052
FOTP	0.199	0.301	0.194	0.394	0.198	0.198	0.350	0.074
PHOW	0.297	0.445	0.312	0.620	0.273	0.277	0.201	0.198
Color	0.337	0.553	0.339	0.663	0.336	0.363	0.345	0.241
Gabor	0.373	0.476	0.356	0.472	0.377	0.306	0.365	0.279
HOG180	0.289	0.506	0.298	0.616	0.269	0.327	0.201	0.198
HOG360	0.281	0.504	0.295	0.604	0.265	0.320	0.130	0.200

of ELM and HP methods in majority cases shows better results than ELM-GMM or ELM-GMMr, both for 'rank' and 'score' measures. The exceptions are *Cardiotocography*, *ImageSegmentation* and *Gabor* datasets, for which ELM-GMM and ELM-GMMr have better 'rank' values; also ELM-GMMr performs better than ELM-GMM in terms of 'rank' for *First-OrderTheoremProving* dataset. In comparison with SVM-based HP method, ELM-HP results in significantly higher 'score' value for almost all datasets (except of *ImageSegmentation* dataset). On the other hand, the relation between 'rank' values coincides with the relation between corresponding accuracies: higher accuracy correlates with the lower 'rank' value.

Conclusions

In this paper, two approaches for probability-based class prediction are presented: both are using Extreme Learning Machines algorithm as a first stage, then the output is transformed into a probability in the second stage. The second stage was performed using Gaussian Mixture models or a new proposed Histogram Probability method.

Based on 10 different experiments, it was concluded that the new proposed Histogram Probability method is providing superior performances. In order to fairly evaluate these performances, new criteria have been proposed: the score and the ranking. This was made necessary since no traditional criteria can easily be used when probability outputs are provided by the classifiers.

Furthermore, the results were compared with SVMs, used at the first stage instead of ELM. As expected, the performances are similar or better for SVMs, but the computational time obtained with SVMs are much larger and may become unrealistic with very large datasets.

Using the new criteria, it was also concluded that the proposed methodologies are efficient in term of probability estimation. Even when the classifiers fail in providing the actual class, this real class has a probability that is very high (but not the highest). To validate this statement, problems that are very challenging in term of classification accuracy have been used: the Caltech-256 datasets. For these datasets, the number of classes is large (10), and number of variables is very large (1700-3600) and the accuracy lie between 13 and 53 percents. Therefore, getting classification probabilities is helpful and may lead to the selection of the most probable classes or may lead in discarding improbable classes.

In the future, the proposed methodologies have to be improved in terms of computational time in order to target Big Data problems. For example, comparison with k-nearest neighbors will be investigated since they can also be used to provide classification probabilities. Furthermore, the new proposed criteria have to be further developed and potentially merged into a new single global criterion.

CHAPTER 4
EXTREME LEARNING MACHINES FOR VISUALIZATION

One of the goals of dimensionality reduction, particularly in visualization, is to represent high-dimensional data onto low-dimensional space while preserving the original information as much as possible. The visualization methods may be divided into two major categories, according to whether the method tries to keep the distances or the topology. Distance-preserving methods include linear MultiDimensional Scaling (MDS) [Kru64] which gives the same solution as PCA, Sammon's mapping [Sam69], Isomap [Ten98, TdL00]. Topology preserving methods include Self-Organizing Maps (SOM) [Koh82, MSML10, LCWV02, DSL$^+$03], Generative Topographic Mapping (GTM) [BSW98], Laplacian Eigenmaps [BN01, BN03].

Such techniques are by definition unsupervised methods; they do not use any external information on each sample, such as a class. In the field of chemometrics, modifications of Principal Component Analysis [Jol86] have been introduced to influence the dimensionality reduction in order to take into account the given class for each sample. As a result, a hybrid dimensionality reduction was proposed, namely the Partial Least Square (PLS) [Wol66, LC08]. Unfortunately, this method is intrinsically linear in terms of dimensionality reduction. Furthermore it assumes that the associated supervised model is also linear.

This chapter introduces a fast nonlinear dimensionality reduction technique named ELMVIS+, originally proposed and described in [AMB$^+$16, ABM$^+$17, CHK$^+$13]. This method has a number of advantages including nonlinearity, scalability, speed, and the capability of intrinsically defining the projected space itself. This chapter presents two extensions of ELMVIS+ technique on the domain of classification and regression problems that incorporate the supervised learning scheme within the existing unsupervised scheme.

4.1 ELM for Non-Linear Visualization

In visualization, data points $\mathbf{x}_i \in \mathbb{R}^d,\ i \in [1, N]$ are projected to the corresponding visualization points $\mathbf{v}_i \in \mathbb{R}^{\tilde{d}},\ i \in [1, N]$ in a smaller dimensional space $\tilde{d}$, usually $\tilde{d} = 2$ or $\tilde{d} = 3$. ELMVIS+ replaces a projection problem by an assignment one, where visualization points $\mathbf{v}_i$ are fixed and the method searches the best $\mathbf{x}_j$ for each $\mathbf{v}_i$ such that the global projection error is minimized. That projection is unknown, but it is approximated with an ELM (treating visualization space $\mathbf{V}$ as inputs and original data $\mathbf{X}$ as targets). Thus, ELM performs a reversed projection (deprojection) compared to standard visualization methods, because it projects visualization space points $\mathbf{V}$ back into the original data space $\mathbb{R}^d$, as shown on Figure 4.1. This approach for building a projection allows to compute an error in the original large-dimensional data space instead of a small-dimensional visualization space, improving the performance.

The basic idea of ELMVIS+ is illustrated in Figure 4.2. ELMVIS+ searches for an optimal assignment of data points $\mathbf{x}_i$ (denoted by circles) in a high-dimensional space (here $d = 2$) to fixed visualization points $\mathbf{v}_j$ (denoted by squares) in a lower-dimensional space (here $d = 1$). ELM learns a projection $\mathbf{V} \rightarrow \mathbf{X}$ and estimates $\hat{\mathbf{X}} = f(\mathbf{V})$.

Although the visualization points and their order are fixed, the order of data samples is not defined. ELMVIS+ method computes an error (a cost function of visualization) with the current order of data samples. The cost function is optimized by swapping two random data samples to change the order, and computing the error;

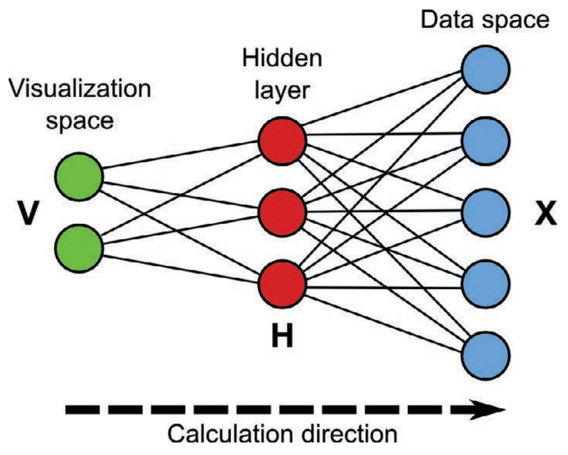


Figure 4.1: ELM learns a reverse projection of visualization points $\mathbf{V}$ in to the data space $\mathbb{R}^d$ with the original samples $\mathbf{X}$

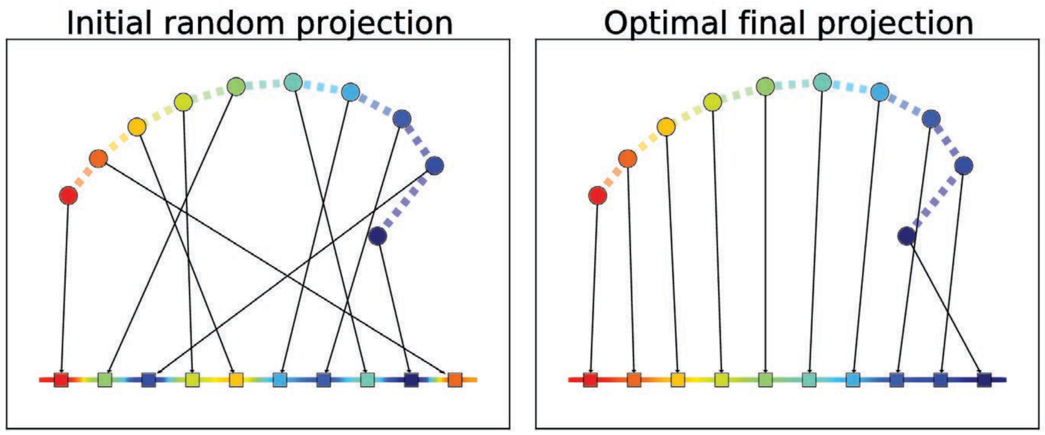


Figure 4.2: Initial and final steps of ELMVIS+ method on toy example

a permutation which reduces the error is kept and the ELM projection is then updated accordingly. Permutations which increase an error are reverted. Swapping and update are the two main steps of ELMVIS+, and they are repeated until the optimal projection is found. What follows is an outline of the ELMVIS+ method:

1. Choose fixed visualization points $\mathbf{V}$ (Can be uniformly distributed, the result of PCA or any other predefined projection)
2. Project visualization points back into the original data space $\mathbf{X}$ with ELM
3. Compute a visualization cost function as an error between the original $\mathbf{X}$ and the projected data samples $\hat{\mathbf{X}}$ (Use average negative cosine similarity for the error)
4. Swap (exchange) two random data samples $\mathbf{x}_a, \mathbf{x}_b$ to change their order and obtain new projection $\hat{\mathbf{X}}_{ab}$ ($\mathbf{V}$ remains constant)
5. If a new permutation decreases the error (error between $\hat{\mathbf{X}}_{ab}$ and $\mathbf{X}$ is less than error between $\hat{\mathbf{X}}$ and $\mathbf{X}$), keep the permutation and update the projected data; otherwise undo the swap
6. Continue swapping until a stopping criterion is reached (such as a maximum number of search steps without a single update, a maximum number of method iterations, or a run-time limit)

In practice, the change in error after swapping samples $\mathbf{x}_a$ and $\mathbf{x}_b$ is computed without performing the swap or explicitly re-training full ELM. If an error decreases for a particular pair of (a, b), then the swap is actually done and $\hat{\mathbf{X}}$ is updated to $\hat{\mathbf{X}}_{ab}$.

ELM outputs an approximated data samples $\hat{\mathbf{X}} = f(\mathbf{V})$ projected from the input points $\mathbf{V}$. If there is a smooth relation between $\mathbf{V}$ and $\mathbf{X}$, the approximation $\hat{\mathbf{X}}$ learned by ELM is close to the true data $\mathbf{X}$. If visualization points $\mathbf{V}$ are located arbitrary and unrelated to $\mathbf{X}$, an ELM with a limited number of neurons fails to learn an accurate projection model, so samples $\hat{\mathbf{X}}$ are far from the original samples in $\mathbf{X}$. Note that sample similarity for the visualization is computed in the original data space $\mathbb{R}^d$, which provides more accurate results that finding it in a reduced visualization space $\mathbb{R}^2$ or $\mathbb{R}^3$.

4.1.1 Optimization and Fast Error Estimation

An ELM in ELMVIS+ is trained on a dataset consisting of inputs $\mathbf{V}$ and targets $\mathbf{X}$. The data is arranged in pairs $(\mathbf{v}_i, \mathbf{x}_i),\ i \in [\![1, N]\!]$. The indexes i in $\mathbf{V}$ and $\mathbf{X}$ are given implicitly by the position of row i in the corresponding data matrix. The optimization starts by selecting two random indexes $a, b : a \neq b,\ a, b \in [\![1, N]\!]$. Then two samples $\mathbf{x}_a, \mathbf{x}_b$ in $\mathbf{X}$ are swapped (exchanged) while $\mathbf{V}$ remains constant. This creates a new dataset, with a different mapping between $\mathbf{V}$ and a new $\mathbf{X}_{ab}$. The same ELM is trained on this new dataset, and outputs a new estimate $\hat{\mathbf{X}}_{ab}$. If the new estimate $\hat{\mathbf{X}}_{ab}$ is closer to the original data in $\mathbf{X}_{ab}$ than the previous one $\hat{\mathbf{X}}$ is to its original $\mathbf{X}$, then the swap is kept. Otherwise the swap is reverted by exchanging $\mathbf{x}_a$ and $\mathbf{x}_b$ again. In practice, the change in error after swapping samples $\mathbf{x}_a$ and $\mathbf{x}_b$ is computed without performing the swap or explicitly re-training full ELM. If an error decreases for a particular pair of (a, b), then the swap is actually done and $\hat{\mathbf{X}}$ is updated to $\hat{\mathbf{X}}_{ab}$. This is when an update step takes place.

The error of ELMVIS+ method is the negative cosine similarity between samples $\mathbf{x}_i$ and $\hat{\mathbf{x}}_i$. The cosine similarity can be used because the absolute value of an error is irrelevant for the optimization, and it provides a convenient formula with a significant speedup over MSE. Let us assume that the input data is normalized to

$\|\mathbf{x}\| = 1$, then $\|\hat{\mathbf{x}}\| \approx 1$ and

$$\text{similarity} = \cos\theta = \frac{\mathbf{x} \cdot \hat{\mathbf{x}}}{\|\mathbf{x}\|\,\|\hat{\mathbf{x}}\|} = \mathbf{x} \cdot \hat{\mathbf{x}} = \mathbf{x}^T \hat{\mathbf{x}} \tag{4.1}$$

For the whole data matrices $\mathbf{X}$, $\hat{\mathbf{X}}$ the error E, which is a negative cosine similarity, is

$$E = -\text{trace}(\mathbf{X}^T \hat{\mathbf{X}}) \tag{4.2}$$

Here $\hat{\mathbf{X}}$ is an ELM prediction. But because the visualization points $\mathbf{V}$ are fixed, the output of an ELM hidden layer $\mathbf{H}$ never changes and needs to be computed only once. A formula based on $\mathbf{H}$ is derived from the ELM solution:

$$\begin{aligned}
\hat{\mathbf{X}} &= \mathbf{H}\beta && (4.3)\\
\beta &= \mathbf{H}^\dagger \mathbf{X} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{X} && (4.4)\\
\hat{\mathbf{X}} &= \mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{X} && (4.5)\\
\mathbf{X}^T\hat{\mathbf{X}} &= \mathbf{X}^T\mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{X} && (4.6)\\
\text{define } \mathbf{A} &\coloneqq \mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T && (4.7)\\
E &= -\text{trace}(\mathbf{X}^T\mathbf{A}\mathbf{X}) && (4.8)
\end{aligned}$$

The matrix $\mathbf{A}$ in equation (4.7) is constant; it depends only on $\mathbf{H}$ and needs to be computed once after an ELM model is built. It is re-used to update error E for every permutation of samples in $\mathbf{X}$. Minimizing equation (4.8) by changing the order of samples in $\mathbf{X}$ is similar to a quadratic assignment problem [BDM12], except that the cost matrix $\mathbf{A}$ has size $(N \times N)$ instead of $(d \times d)$. In fact, there exist an explicit formula for the change of error Δ_E if a row $\mathbf{x}_a$ in matrix $\mathbf{X}$ is changed by an amount (vector) δ. The formula is given below. If only Δ_E is required, two last update steps are omitted.

$$\Delta_E = \sum_{j=1}^{d} \Big(\mathbf{A}_{a,a}\delta_j^2 + 2\hat{\mathbf{x}}_{a,j}\delta_j \Big) \tag{4.9}$$

$$\hat{\mathbf{X}} \leftarrow \hat{\mathbf{X}} - \mathbf{A}_{:,a} \times \delta \tag{4.10}$$

$$\mathbf{X}_{:,a} \leftarrow \mathbf{X}_{:,a} + \delta \tag{4.11}$$

4.1.2 ELMVIS+ Algorithm

The method starts by taking a random permutation of $\mathbf{X}$ and training an ELM on $(\mathbf{V}, \mathbf{X})$ dataset. Then a fixed matrix $\mathbf{A}$ is computed from equation (4.7), and an initial data reconstruction $\hat{\mathbf{X}} = \mathbf{AX}$ is obtained. This is done only once when the method starts. Then two random indices $a, b \in [\![1, N]\!]$ are taken, and the change in error Δ_E is computed by formula (4.9). This is a search step. To reduce computational complexity of a search step from $\mathcal{O}(Nd^2)$ to $\mathcal{O}(d)$ (in terms of a pair of candidates for swap), the matrix $\hat{\mathbf{X}}$ is being stored explicitly. If the change of error is negative $\Delta_E < 0$, then an update step is performed which swaps rows $\mathbf{x}_a, \mathbf{x}_b$ in $\mathbf{X}$, and updates $\hat{\mathbf{X}}$ to be the optimal reconstruction for the new $\mathbf{X}$. According to the equation (4.10), computational complexity of an update step is $\mathcal{O}(Nd)$. If the change of error is non-negative $\Delta_E \geq 0$, then search continues. Search and update steps continue until some stopping criterion is met, such as a maximum number of search steps without a single update, a maximum number of method iterations, or a runtime limit. When ELMVIS+ terminates, data samples in matrix $\mathbf{X}$ are in the optimal order for the given visualization points in $\mathbf{V}$. A graphical illustration of ELMVIS+

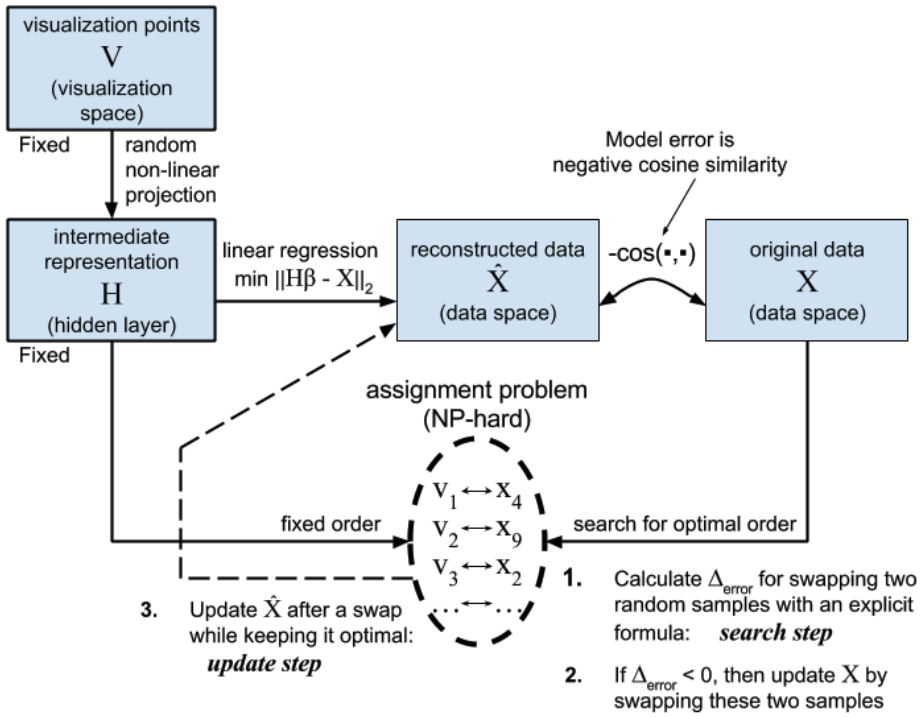

visualization points
V
(visualization space)
Fixed
random non-linear projection
intermediate representation
H
(hidden layer)
Fixed
linear regression
min ||Hβ - X||$_2$
reconstructed data
$\hat{X}$
(data space)
Model error is negative cosine similarity
-cos(▪,▪)
original data
X
(data space)
assignment problem (NP-hard)
$v_1 \leftrightarrow x_4$
$v_2 \leftrightarrow x_9$
$v_3 \leftrightarrow x_2$
fixed order
search for optimal order
1. Calculate Δ_{error} for swapping two random samples with an explicit formula: search step
2. If $\Delta_{error} < 0$, then update X by swapping these two samples
3. Update $\hat{X}$ after a swap while keeping it optimal: update step

Figure 4.3: Schematic representation of ELMVIS+ algorithm

method is shown on Figure 4.3.

4.1.3 Example of using ELMVIS+ on different datasets

As an example of the ELMVIS+ method the visualization results of some different real-world datasets is provided in this section: Artificial Faces Dataset and MNIST dataset. The first dataset is a set of 698 face images proposed in [TdL00], and then widely used for benchmark purposes, for instance in [LV07, VPN$^+$10]. These images are computer renderings of a 3D sculpture head under different poses and lighting directions. Each image consists of an array of 64 by 64 brightness values of pixels, giving the input data dimensionality of 4096. An example visualization is shown on Figure 4.4, where the method shows a clear manifold structure organisation of sculpture faces.

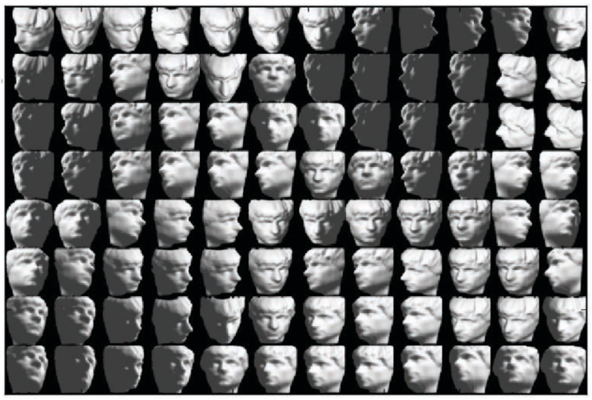


Figure 4.4: ELMVIS+ visualization using 20 neurons

The second dataset is the well-known MNIST database of handwritten digits. The digits are written by 250 different writers. The original black and white images of digits were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. The images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field [LBBH98]. Visualization results for part of this dataset are shown on Figure 4.5, where each digit is colored for better distinction.

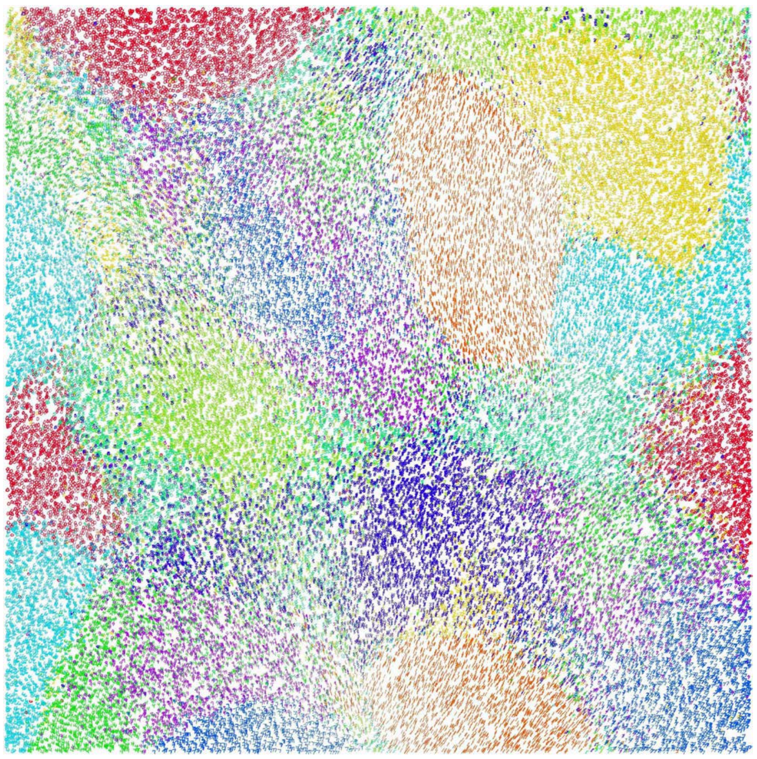


Figure 4.5: Visualization of 15000 samples from the MNIST dataset

4.2 ELMVIS+ for Classification

This section describes an extended version of ELMVIS+ that incorporates the supervised learning scheme within the existing unsupervised scheme. In other words, ELMVIS++C has an additional supervised learning component compared to ELMVIS+, which is originally an unsupervised method as like the majority of the other dimensionality reduction method. This component prevents samples under the same class being separated apart from each other. In this improved method, the importance of the supervised component can be further tuned to have different level of influence. The test results on four datasets indicate that the proposed improvement not only maintains the performance of ELMVIS+, but also is extremely beneficial for certain applications where the visualization of the data in relation with the class becomes an important issue.

4.2.1 Methodology

ELMVIS+ is using only data from feature space for visualization, not considering the class information. For some problems, like MNIST [LBBH98], data instances may have similar values of features while belonging to different classes. As the result, instances from one class form several separated groups in the visualization projection rather than constituting a sole group per class (see Figure 4.5). In this case, the class information can be essential and help for better visualization by preventing instances under the same class to be separated apart from each other. To add class information to the visualization model the class is concatenated as an extra feature vector

$$\tilde{X} = [X\ Y] , \tag{4.12}$$

where X stands for original data, and Y is a class vector. For multiclass datasets, class vector should be at first transformed into a target matrix T in a following way. Assume $y_i \in \{1 \ldots C\}$ is a class of i-th sample, where C is the number of distinct classes. Encode classification targets as one binary variable for each class (one-hot encoding). T is the matrix of targets such that $\mathbf{T}_{ij} = 1 \iff y_i = j$, i.e., sample i belongs to class j. Otherwise, $\mathbf{T}_{ij} = 0$. In this case

$$\tilde{X} = [X\ T] \tag{4.13}$$

Adding class information as additional feature to the data forces ELMVIS+ method to project samples that belong to a same class together on a projection plane and increase the penalty of projecting samples from different classes close to each other. On the other hand, considering a high-dimensional binary-class problem, adding a sole feature vector would not influence a lot the results. To overcome this issue, the class information can receive larger weight. One of the ways to do so is to add a class vector several times. Applying this approach the importance of class information can be increased to a needed extent. Furthermore, an initialization of ELMVIS++C from an existing ELMVIS+ projection always increases the cosine similarity, and decreases the number of iterations before convergence.

To show how class information alters the visualization results, ELMVIS++C visualization method is tested on four binary-class datasets: *Banknote*, *Climate*, *Di-*

Table 4.1: Description of used datasets

Dataset	# of instances	# of features	Class ratio
Banknote	1372	5	762 / 610
Climate	540	18	46 / 494
Diabetic	1151	20	540 / 611
Spectrum	206	299	108 / 98
Location	21048	529	5785 / 5503 / 9760

abetic and *Spectrum*, and one dataset with three classes – *Location*. The *Spectrum* dataset is taken from [BL12], and other four datasets are taken from the University of California at Irvine (UCI) Machine Learning Repository [Lic13].

4.2.2 Experiments

The statistical information on datasets (number of instances, dimensionality and class ratio) is provided in Table 4.1.

Banknote Dataset

The dataset is based on the data that were extracted from images taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images have 400 $\times$ 400 pixels. Wavelet Transform tool was used to extract features. The dataset uses the following features: variance, skewness and kurtosis of the Wavelet Transformed image; and the entropy of the image. The class is assigned to be 0 if the banknote is genuine and 1 otherwise.

Climate Dataset

This dataset contains records of simulation crashes encountered during climate model uncertainty quantification (UQ) ensembles. Ensemble members were constructed using a Latin hypercube method in LLNL's UQ Pipeline software system to sample the uncertainties of 18 model parameters within the Parallel Ocean Program (POP2) component of the Community Climate System Model (CCSM4). The goal is to use classification to predict simulation outcomes (fail or succeed) from input parameter values, and to use sensitivity analysis and feature selection to determine the causes of simulation crashes. Further details about the data and methods are given in [LKT+13]. The goal is to predict climate model simulation outcomes given values of climate model input parameters scaled in the interval $[0, 1]$. The simulation outcome is coded as 0 for failure, and 1 for success.

Diabetic Dataset

This dataset contains features extracted from the Messidor image set [KMC+16, DZC+14] to predict whether an image contains signs of diabetic retinopathy or not. All features represent either a detected lesion, a descriptive feature of a anatomical part or an image-level descriptor. The underlying method image analysis and feature extraction as well as our classification technique is described in [AH14]. The classes in the dataset are labeled in the following way: 1 if an instance contains signs of diabetic retinopathy, and 0 if there are no signs of diabetic retinopathy.

Location Dataset

The original dataset was created by means of more than 20 different users and 25 Android devices and split into training set consisted of 19937 and test set with 1111 records. Each sample in the dataset is described by a set of 529 attributes that contain the WiFi fingerprint, the coordinates where it was taken, and other additional information. Using this dataset for classification task, building identification has been chosen as target variable.

Spectrum Dataset

In this dataset each instance of data represents a body shape of an individual. The body shape data used in this experiment is from the same 3D whole body scan database of [BL12]. It is composed of 3D laser scanned triangular mesh models of 98 females and 108 males who lives in Korea. Individuals were scanned with tight shorts and a swimming cap, and a bra-top for the females. The raw scan data contains approximately 120,000∼130,000 vertices. However, since the raw data are not compatible to each other, i.e. one-to-one correspondences between body parts are not defined, they went through additional preprocess before the manifold learning was actually conducted. For the compatible description of the 3D scan data, the first 300 low-frequency spectrums of the Laplace-Beltrami operator defined over the 3D model are used. Such a way of describing a 3D model is one of the most commonly accepted method in computational geometry, and shows the most reliable result in terms of posture-invariant shape description [RWP06]. Exploration of shape space is one of the important problems in computer-aided design (CAD) that is not completely solved yet. The aim of such is to estimating a manifold that describes the space where the shapes are distributed. Accurately estimating the shape space manifold is of paramount importance with a number of applications, including parametrization of the complicated geometry, geometry modeling with shape synthesis, statistical shape analysis, visualization of geometry data distribution for design exploration, and so on. Especially for human model data, adequate mapping and visualization of the data distribution to the lower dimensional space is particularly important, because this could be used for a plenty of ergonomic and human factor engineering applications, including the design of anthropometric dummies, ergonomic design optimization, assessment of work environment, and etc.

Using the similarity as an error estimation (see Section 4.1.1), it is necessary to normalize data. To do so, for each feature vector x, at first the mean value $\bar{x}$ is subtracted and then the obtained vector is divided by L_2 norm $\|x\|$.

The projection space is initialized uniformly at random, projection points are normalized to zero mean and unit variance.

For each dataset reconstruction ELM is built with the number of neurons L depending on the dimensionality d of the original data. This dependence is expressed in the following way

$$L = L_{lin} + L_{sigm} = 2 + \max(\lfloor\sqrt{d}\rfloor + 1,\ 20), \tag{4.14}$$

where L_{lin} refers to neurons with linear activation function, and L_{sigm} refers to neurons with sigmoid activation function.

ELMVIS+ toolbox [Aku16] based on High-Performance ELM toolbox [ABML15] was used to perform the experiments. The following parameter settings were used to constrain its running time: maximum number of iterations was set to 10^7, and *stall* parameter was set to 10^4.

To show how class information influences visualization results, the following setups are used: original ELMVIS+, ELMVIS++C with unweighted class information (referred as ELMVIS++C in Table 4.2), and ELMVIS++C that weights class information correspondingly to feature data (referred as ELMVIS++C2 in Table 4.2). Both ELMVIS++C methods are initialized from an existing projection of ELMVIS+. For each dataset and each setup the final cosine similarity shows significant improvement.

Results

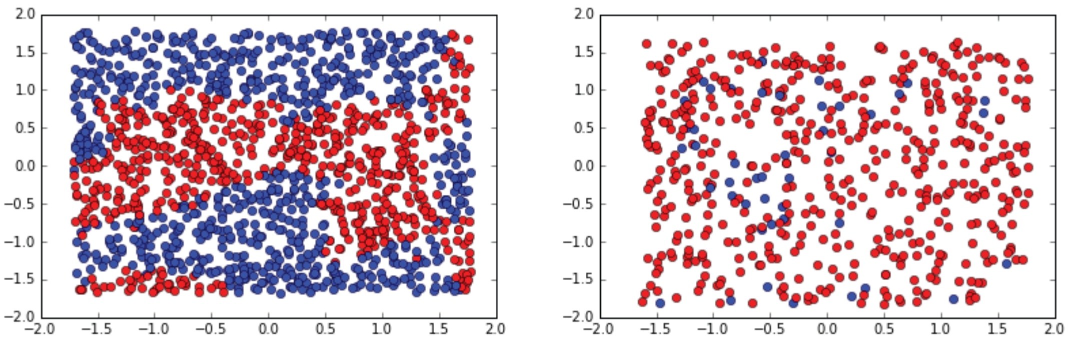
Because the original goal is not to compare the proposed improvement of ELMVIS+ method and the original visualization method, but to show how adding class information can change the projection, the following parameters are monitored: original similarity, final similarity before achieving the convergence conditions and presented in Table 4.2.

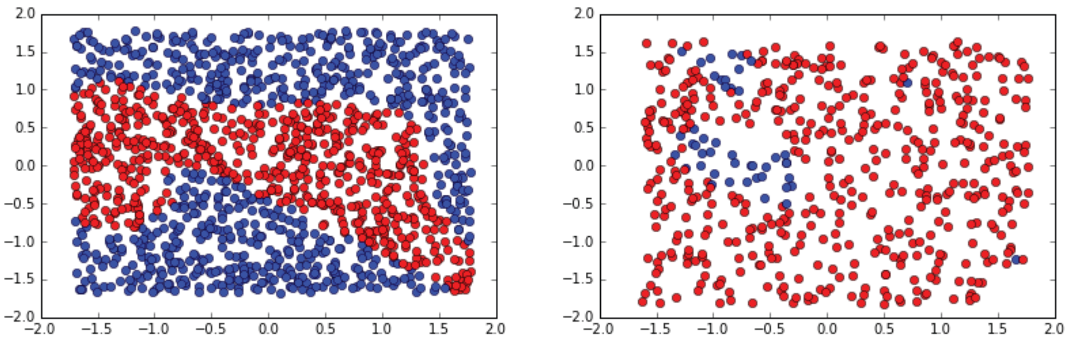
Table 4.2: Comparison of cosine similarities between original and de-projected datasets.

	ELMVIS+		ELMVIS++C		ELMVIS++C2	
	initial	final	initial	final	initial	final
Banknote	0.021	0.936	0.634	0.924	0.574	0.874
Climate	0.035	0.334	0.238	0.334	0.181	0.518
Diabetic	0.022	0.705	0.601	0.692	0.323	0.721
Spectrum	0.085	0.985	0.764	0.985	0.756	0.924

Figures 4.6 and 4.7 depicts visualization results of aforementioned methods. It can be noticed that giving class information larger weight results in projection that put samples from one class together, forming clearly distinguished clusters. Figure 4.8 shows the visualization of *Spectrum* dataset, obtained with ELMVIS++C2 method, where each data sample is replaced with the original body shape. From here it can be observed, that for the second method, body shapes on the border between two classes have more visual differences than for the original ELMVIS+, though inside 'male' and 'female' regions the shapes preserve visual similarity. Figure 4.9 presents visualization results obtained with ELMVIS++C method for *Location* dataset. In the current example, building ID variable has been chosen as the supervised component, used

to improve visualization. From the Figure 4.9 the difference between visualization without class information (Figure 4.9a), with class information (Figure 4.9b) and with weighted class information (Figure 4.9c) can be clearly observed.


(a) ELMVIS+: Visualization without class variable


(b) ELMVIS++C: Visualization with unweighted class variable

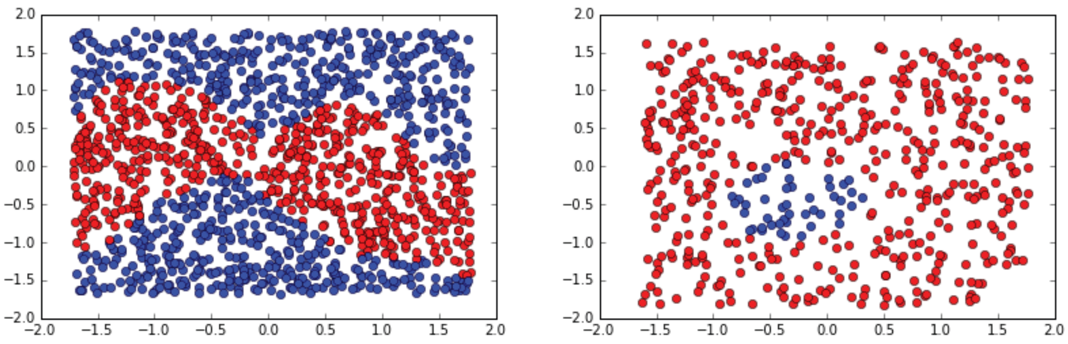
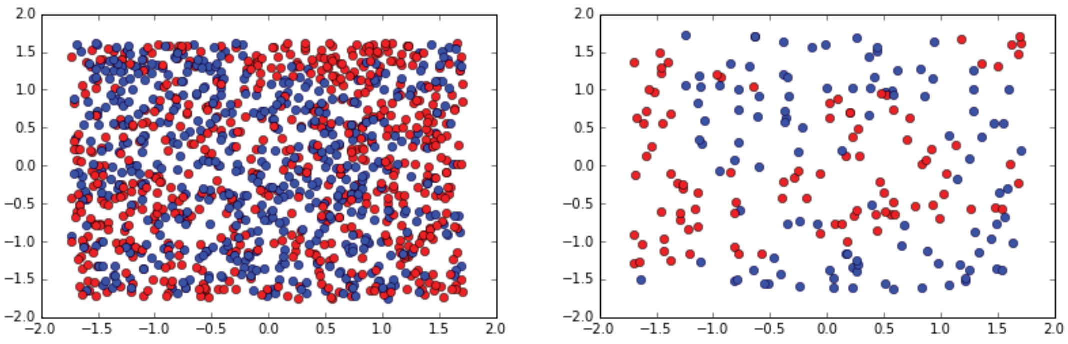
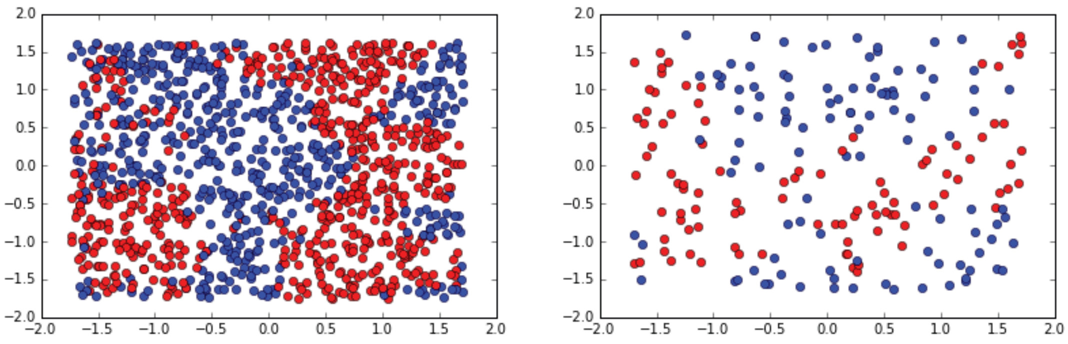

(c) ELMVIS++C: Visualization with weighted class variable

Figure 4.6: Comparison of visualization of *Banknote* (*left*) and *Climate* (*right*) datasets


(a) ELMVIS+: Visualization without class variable


(b) ELMVIS++C: Visualization with unweighted class variable

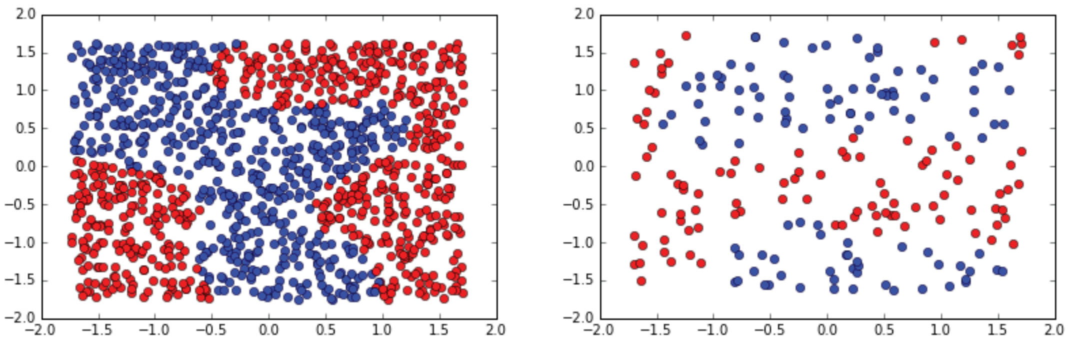

(c) ELMVIS++C: Visualization with weighted class variable

Figure 4.7: Comparison of visualization of *Diabetic* (*left*) and *Spectrum* (*right*) datasets

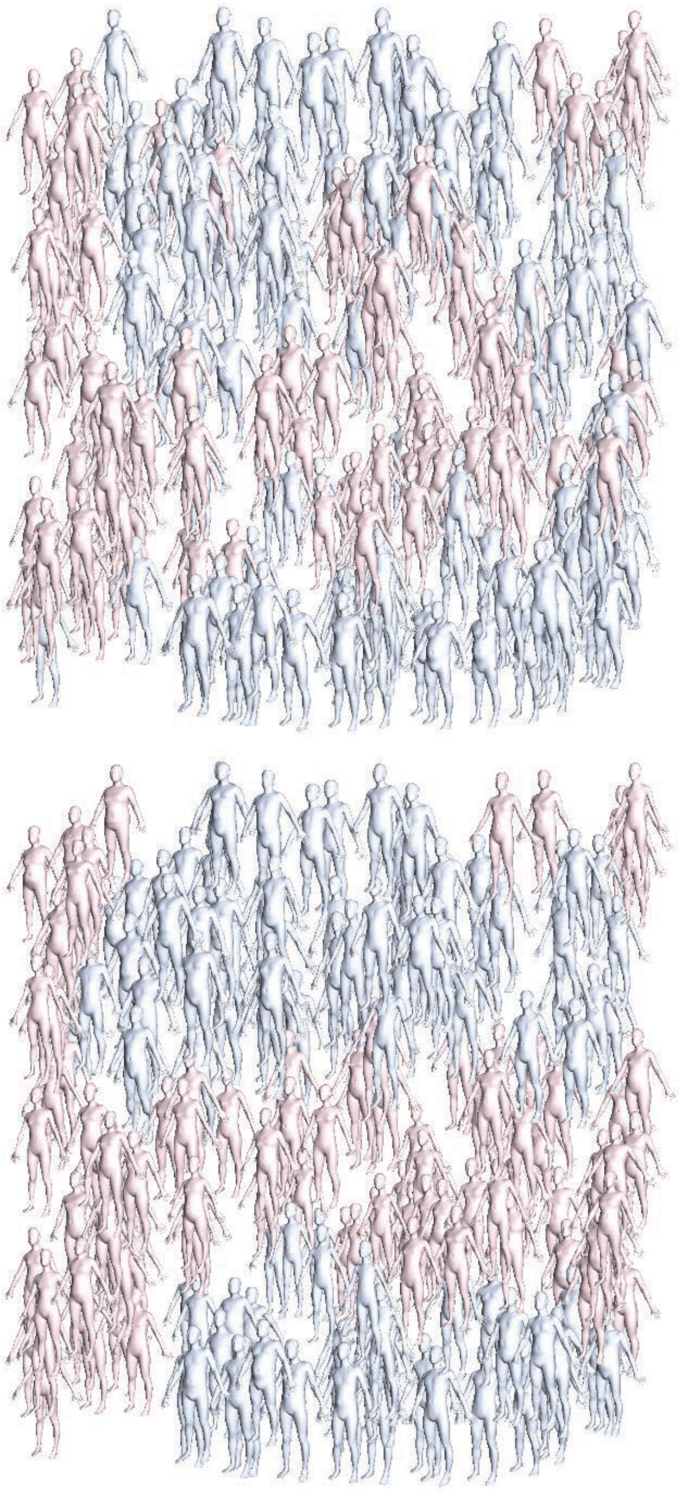
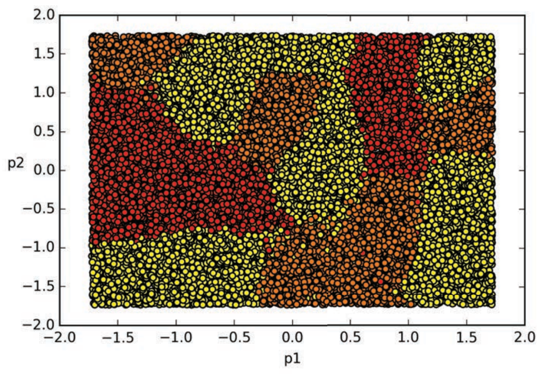
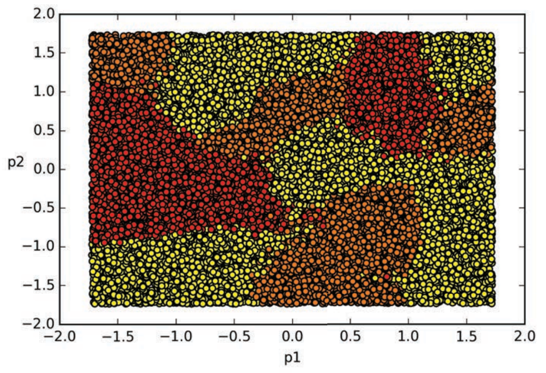


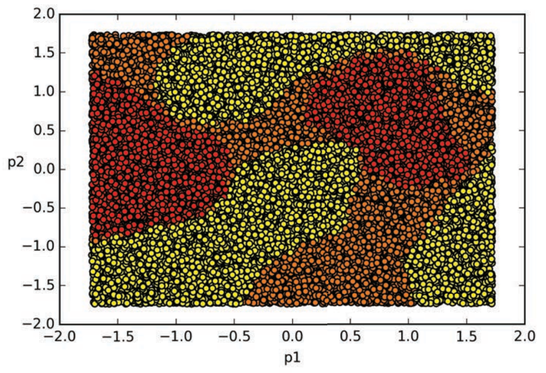
Figure 4.8: Visualization of *Spectrum* dataset: ELMVIS+ (*top*) and weighted ELMVIS++C (*bottom*)



(a) ELMVIS+: Visualization without class variable



(b) ELMVIS++C: Visualization with unweighted class variable



(c) ELMVIS++C: Visualization with weighted class variable

Figure 4.9: Visualization results for *Location* dataset, using building ID as class variable

4.3 ELMVIS for Regression

As the development of the idea of incorporating the supervised learning scheme within the unsupervised scheme for visualization task, and encouraged by the successful results of ELMVIS++C extension [GAM+16], a further improvement of ELMVIS+ method was proposed – apply ELMVIS+ for the regression problems, using target values to improve visualization results.

4.3.1 Methodology

According to the scheme, adding target variable as additional feature to the data forces ELMVIS+ method to project samples with similar target values together on a projection plane and increase the penalty of projecting samples with target values that differ a lot close to each other. On the other hand, for high-dimensional regression problems, adding a sole feature vector would not influence a lot the results. In order to overcome this issue, the supervised component can receive larger weight. The most obvious way to do so is to add a vector of target values several times to the feature space. Applying this approach the importance of supervised component can be increased to a needed extent. Furthermore, an initialization of ELMVIS++R from an existing ELMVIS+ projection always increases the cosine similarity, and decreases the number of iterations before convergence.

4.3.2 Experiments

To prove the idea that using supervised component can improve visualization results and to show how the influence on these results changes with the increase of component's weight, the proposed visualization method is tested on six different datasets: *Yacht*, *Concrete*, *CCPP*, *Wine*, *Gas* and *Location*. All datasets are taken from the University of California at Irvine (UCI) Machine Learning Repository [Lic13].

Datasets

The statistical information on datasets (number of instances and dimensionality) used to approve the proposed method is provided in Table 4.3. The datasets are chosen in a way to provide regression tasks with different ratio between number of samples N and number of features d:

- small N, small d (*Yacht*, *Concrete*)
- big N, small d (*CCPP*, *Wine*)
- big N, big d (*Gas*, *Location*)

Yacht Hydrodynamics Dataset

Essential inputs include the basic hull dimensions and the boat velocity. The current dataset comprises 308 full-scale experiments, which were performed at the Delft Ship Hydromechanics Laboratory to predict residuary resistance of sailing yachts [GOV81, OLG07]. All of the attributes and predicted values are adimensional.

Concrete Compressive Strength Dataset

The concrete compressive strength is a highly nonlinear function of age and ingredients. These ingredients include cement, blast furnace slag, fly ash, water,

Table 4.3: Description of used datasets

Dataset	# of instances	# of features
Yacht	308	7
Concrete	1030	9
CCPP	9568	4
Wine	6497	11
Gas	13910	129
Location	21048	529

superplasticizer, coarse aggregate, and fine aggregate [Yeh98]. The original data consists of 1030 measurements that are given in raw form (not scaled).

Combined Cycle Power Plant Dataset

The *CCPP* dataset contains 9568 data points collected from a Combined Cycle Power Plant over 6 years (2006-2011), when the power plant was set to work with full load [KTG12]. Features consist of hourly average ambient variables: temperature, ambient pressure, relative humidity, exhaust vacuum. The averages are taken from various sensors located around the plant that record the ambient variables every second and are given without normalization [T14].

Wine Quality Dataset

The *Wine* dataset used in these experiments is created using red and white wine datasets (consist of 1599 and 4898 instances, respectively). Due to privacy and logistic issues, only physicochemical variables are available as inputs, though there is no data about grape types, wine brand, wine selling price, etc. The output is based on sensory data and is calculated as a median of at least 3 evaluations made by wine experts. Each expert graded the wine quality between 0 (very bad) and 10 (very excellent) [CCA$^+$09].

Gas Sensor Array Drift Dataset at Different Concentrations Dataset

This data set contains 13,910 measurements from 16 chemical sensors exposed to 6 gases at different concentration levels and was gathered during the period of 36 months [VVA$^+$12]. Each feature vector contains 8 features extracted from each particular sensor, resulting in a 128-dimensional feature vector (8 features $\times$ 16 sensors) containing all the features, 129 considering gas type [RLFV$^+$14]. The original dataset is organized into ten batches, each containing different number of measurements per class and month.

UJIIndoorLoc Dataset

The original dataset was created by means of more than 20 different users and 25 Android devices and split into training set consisted of 19937 and test set with 1111 records. Each sample in the dataset is described by a set of 529 attributes that contain the WiFi fingerprint, the coordinates where it was taken, and other additional information. Coordinates are represented by latitude and longitude estimations that were used as target variables and corresponding datasets are named *Loc-lat* and *Loc-long*, respectively.

Regardless of the form, in which the original data was presented, all datasets were preprocessed in the following way. First of all, samples in each dataset were combined (if stored in different subsets) and randomly permuted. Additionally, using the similarity as an error estimation (see Section 4.1.1), it is necessary to normalize data. To do so, for each feature vector x, at first the mean value $\bar{x}$ is subtracted and then the obtained vector is divided by L_2 norm $\|x\|$.

Though for ELMVIS+ method data can be projected on the arbitrary assigned space, for the sake of simplicity in current experiments the visualization space is initialized uniformly at random in a range $[-1, 1]$ for both dimensions, and projection points are normalized to zero mean and unit variance.

For each dataset reconstruction ELM is built with the number of neurons L depending on the dimensionality d of the original data. This dependence is expressed in the following way

$$L = L_{lin} + L_{sigm} = 2 + \max(\lfloor\sqrt{d}\rfloor + 1,\ 20), \tag{4.15}$$

where L_{lin} refers to neurons with linear activation function, and L_{sigm} refers to neurons with sigmoid activation function.

ELMVIS+ toolbox [Aku16] based on High-Performance ELM toolbox [ABML15] was used to perform the experiments. Considering high dimensionality of datasets the following parameter settings were used to obtain higher accuracy:

maximum number of iterations was set to 10^7, and *stall* parameter (regulates how many unsuccessful swaps have to be performed before the algorithm stops) was set to 10^4.

In order to show how supervised component influences visualization results and general accuracy in terms of cosine similarity, the following three methods with different proportion of target information are used: ELMVIS+ built using original feature space in dataset, ELMVIS++R with unweighted target information, and ELMVIS++R2 that weights target variables proportionally to feature space (referred as ELMVIS+, ELMVIS++R and ELMVIS++R2 in Tables 4.4 to 4.6, respectively). Both methods built with supervised component use the final projection of ELMVIS+ as initialization visualization state. For each dataset and each setup the final cosine similarities, presented in Table 4.4), show significant improvement comparing to the initial projections. Table 4.4 also contains information about MSE, calculated for each dataset in a standard way. Table 4.5 displays cosine similarity, calculated separately for target vector and feature space, using the same projecting ELM model; this is done to give the comprehension on how feature space and supervised component (weighted or unweighted) influence the projection accuracy of each other. For comparison reasons, cosine similarity calculated for ELMVIS+ method is also included in the table. Table 4.6 contains information on how many updates and overall time it required to obtain final visualization results for each of the compared methods.

Table 4.4: Comparison of MSE and cosine similarities between original and de-projected datasets

	ELMVIS+			ELMVIS++R			ELMVIS++R2		
	MSE	initial	final	MSE	initial	final	MSE	initial	final
Yacht	0.031	0.071	0.816	0.027	0.629	0.807	**0.013**	0.616	**0.838**
Concrete	0.026	0.027	0.789	0.025	0.625	0.771	**0.013**	0.528	**0.794**
CCPP	0.016	0.004	**0.936**	0.017	0.819	0.913	**0.011**	0.808	0.910
Wine	0.025	0.010	0.720	0.026	0.549	0.686	**0.013**	0.394	**0.724**
Gas	0.002	0.043	0.817	0.002	0.726	0.804	**0.001**	0.667	**0.841**
Loc-lat	0.001	0.005	0.331	0.001	0.289	0.325	<**0.001**	0.532	**0.637**
Loc-long	0.001	0.005	0.351	0.001	0.332	0.351	<**0.001**	0.627	**0.680**

Results

Tables 4.4 to 4.6 show the experimental results of performing the proposed method for regression problems. It has to be noted that experiments for *Yacht* and *Concrete* datasets terminated before reaching the maximum amount of iterations, and for *Wine* dataset the number of iterations almost approached the limit. For the rest of datasets, the process terminated according to the restriction on the amount of

Table 4.5: Comparison of cosine similarities between original and de-projected datasets, calculated, using the same projecting model, for (1) only target features and (2) without target features

	ELMVIS+		ELMVIS++R				ELMVIS++R2			
			targets		w/o targets		targets		w/o targets	
	initial	final	initial	final	initial	final	initial	final	initial	final
Yacht	0.071	0.816	0.682	0.812	0.819	0.904	0.734	0.922	0.807	0.901
Concrete	0.027	0.789	0.495	0.728	0.818	0.882	0.435	0.854	0.816	0.850
CCPP	0.004	0.936	0.854	0.905	0.928	0.968	0.855	0.938	0.928	0.960
Wine	0.010	0.720	0.281	0.586	0.774	0.832	0.290	0.825	0.753	0.793
Gas	0.043	0.817	0.665	0.731	0.863	0.904	0.719	0.937	0.841	0.883
Loc-lat	0.005	0.331	0.681	0.717	0.521	0.553	0.744	0.923	0.551	0.557
Loc-long	0.005	0.351	0.841	0.833	0.560	0.577	0.888	0.930	0.573	0.579

iterations.

Although, the original goal was not to compare the proposed improvement of ELMVIS+ method and the original visualization method in terms of cosine similarity, but to show how using a supervised component (target variable) can change the projection, the following parameters were monitored: cosine similarities at the initialization stage, final cosine similarities before achieving the convergence condi-

Table 4.6: Descriptive table on the computational resources required to obtain final projections

	ELMVIS+		ELMVIS++R		ELMVIS++R2	
	updates	time	updates	time	updates	time
Yacht	1467	17.55	887	22.06	855	14.62
Concrete	5788	66.03	2535	68.97	3871	76.69
CCPP	61517	989.07	25982	878.88	33051	767.49
Wine	39744	936.73	15595	810.15	27547	752.36
Gas	90404	1027.28	45081	886.89	53796	787.09
Loc-lat	116096	1177.60	34068	1771.01	61630	1409.56
Loc-long	118053	1284.21	31678	1496.01	52019	1260.90

tions (presented in Tables 4.4 and 4.5 and referred as *initial* and *final*, respectively) and also the number of updates and overall running time (presented in Table 4.6). Additionally, MSE for each method was included in Table 4.4.

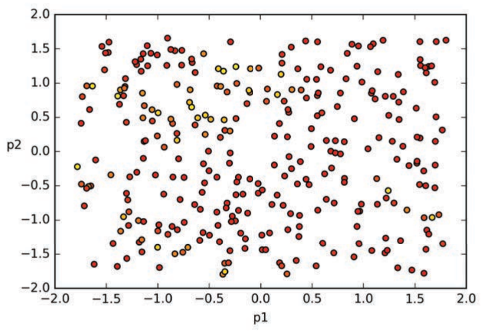
From Table 4.4, is can be observed that for the majority of datasets adding weighted supervised component to the data space results in higher cosine similarity. Also, it can be noticed that using target information to build projection always leads to the decrease of MSE. Results in Table 4.5 indicate that using target information to build a projection results in the increase of cosine similarity, though, adding weight to supervised component usually leads to the slight deterioration of the projection in terms of similarity (which is nevertheless higher than for the original ELMVIS+ method).

Figures 4.10 to 4.16 depict visualization results obtained with aforementioned methods: (a) ELMVIS+, (b) ELMVIS+ adding target variables to the feature space, (c) ELMVIS+ giving corresponding weight to target variables; (d) show the same visualization results as in (c), but presented as 3-D plots. It can be noticed that adding target variable and assigning it larger weight results in projection with more smooth transition of target values between samples.

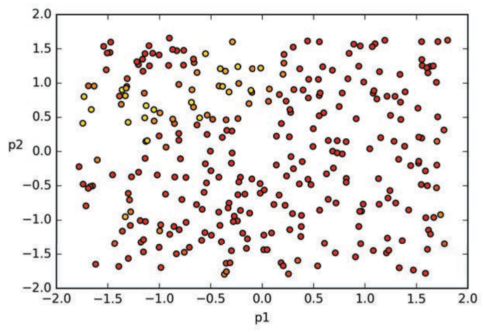
Conclusions

From the results in Table 4.4 it can be noticed, that the higher the dimensionality of the original feature space, the more ELMVIS++R increases similarity, when giving a higher weight to the target variable, while a little bit reducing it for small-dimensional datasets. Moreover, Table 4.5 gives a better comprehension on how weight of supervised component influences projections in terms of cosine similarity: the higher the number of samples in dataset, the less the influence of the weight of target information (see columns 6 & 10). With this, it can be concluded, that an automatic way of assigning weight to the supervised component has to be developed for successful utilization of the proposed method.

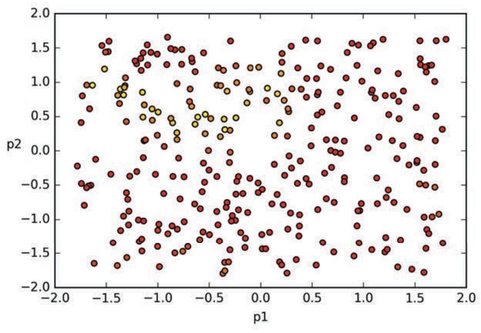
ELMVIS++R possesses the similar limitations as the original ELMVIS+; in order to perform the visualization in quasi-real time, it is beneficial to be able to store all the data in memory. In future work, we will analyze the bottleneck of ELMVIS+ family of visualization methods in order to reduce its computational complexity.



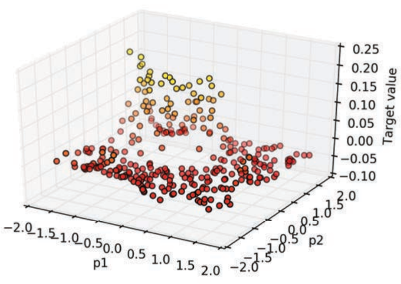
(a) Visualization without supervised component



(b) Visualization with supervised component

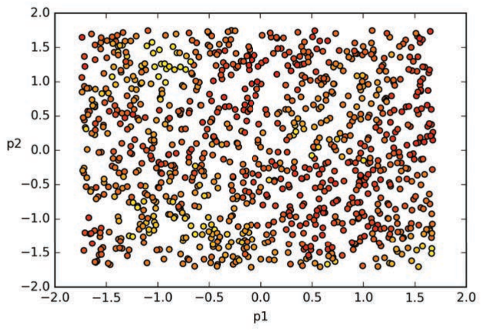


(c) Visualization with weighted supervised component

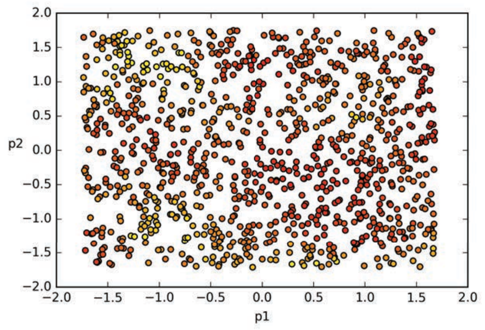


(d) 3D Visualization with weighted supervised component

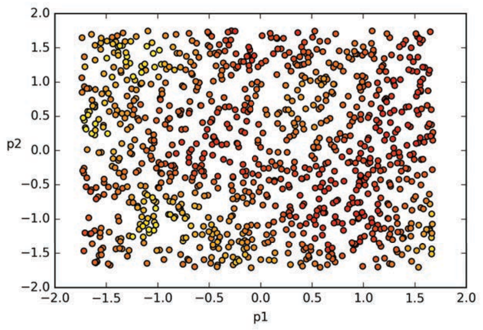
Figure 4.10: Visualization results for *Yacht* dataset



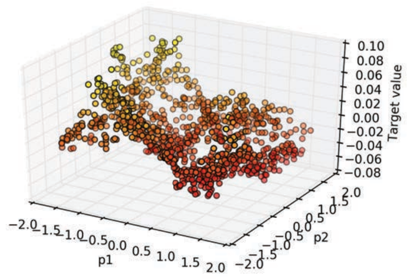
(a) Visualization without supervised component



(b) Visualization with supervised component

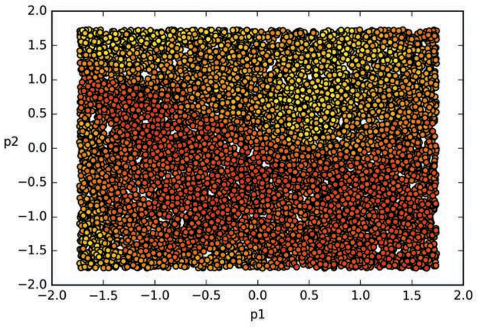


(c) Visualization with weighted supervised component

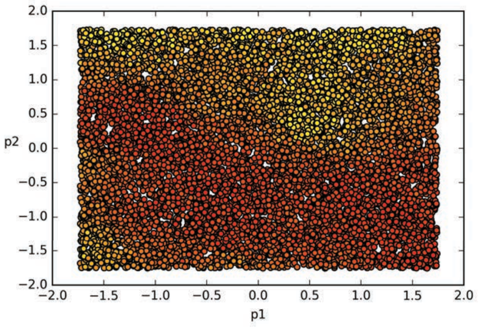


(d) 3D Visualization with weighted supervised component

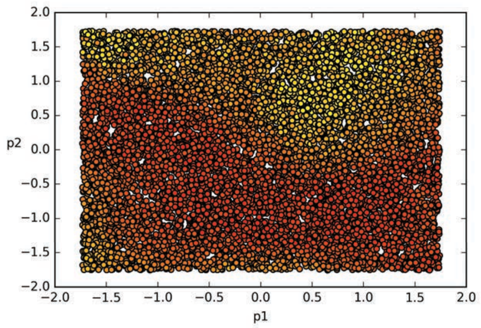
Figure 4.11: Visualization results for *Concrete* dataset



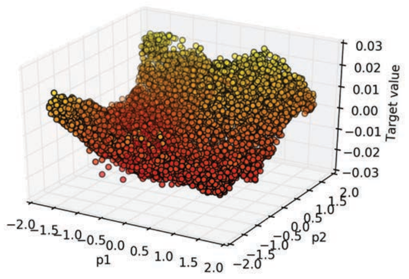
(a) Visualization without supervised component



(b) Visualization with supervised component

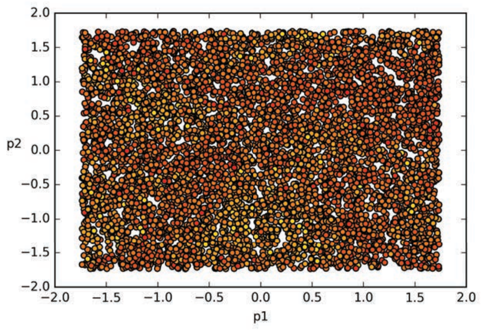


(c) Visualization with weighted supervised component

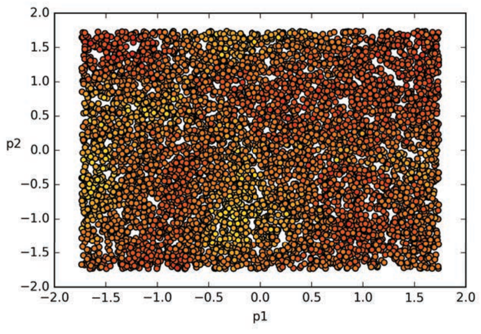


(d) 3D Visualization with weighted supervised component

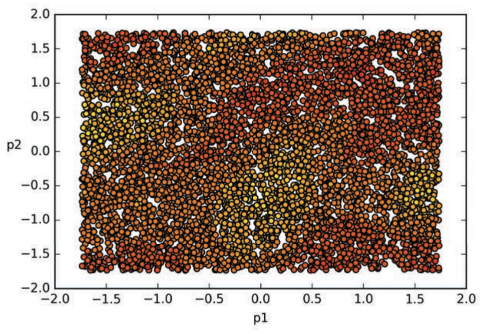
Figure 4.12: Visualization results for *CCPP* dataset



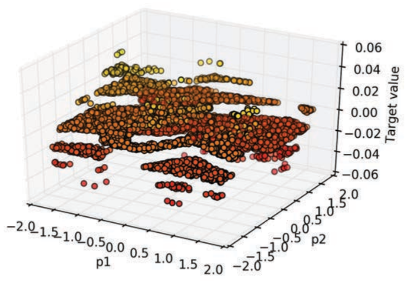
(a) Visualization without supervised component



(b) Visualization with supervised component

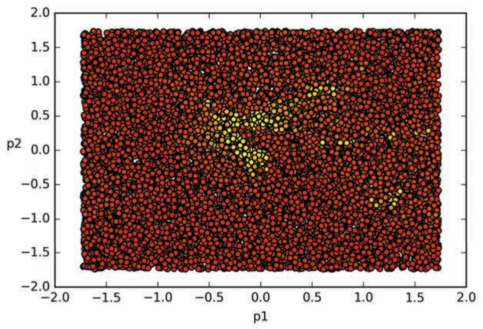


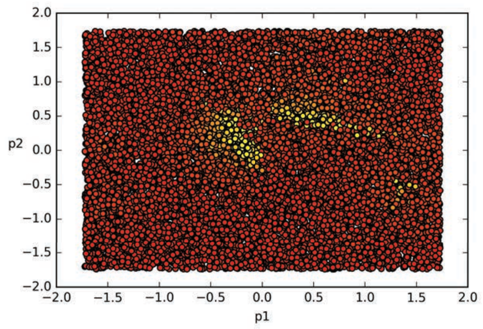
(c) Visualization with weighted supervised component

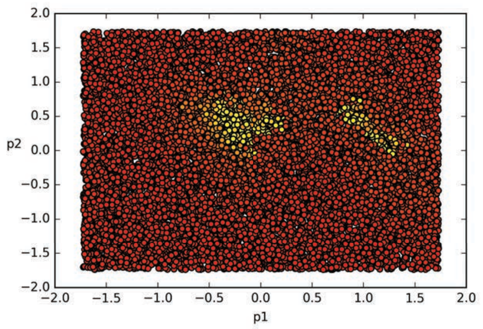


(d) 3D Visualization with weighted supervised component

Figure 4.13: Visualization results for *Wine* dataset


(a) Visualization without supervised component


(b) Visualization with supervised component


(c) Visualization with weighted supervised component

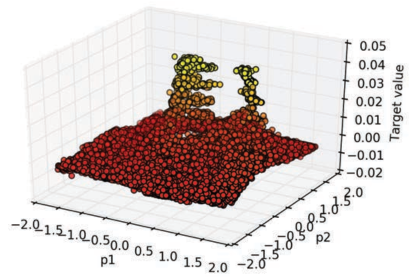
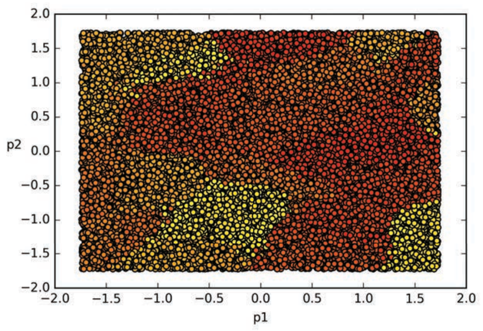
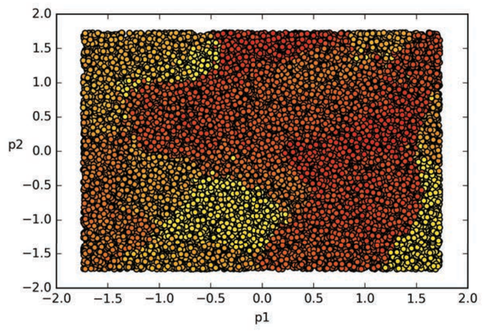

(d) 3D Visualization with weighted supervised component

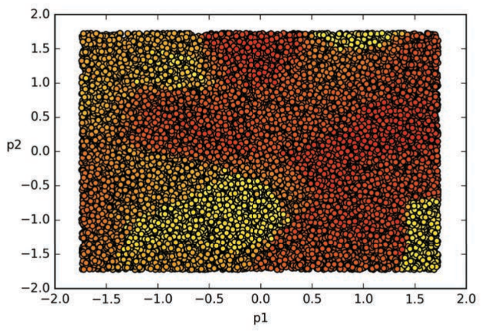
Figure 4.14: Visualization results for *Gas* dataset



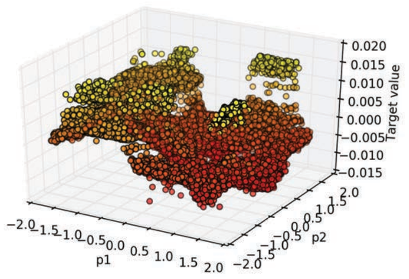
(a) Visualization without supervised component



(b) Visualization with supervised component

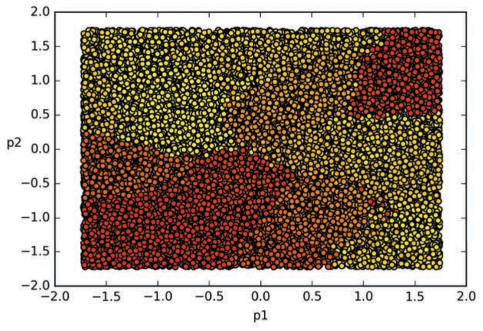


(c) Visualization with weighted supervised component

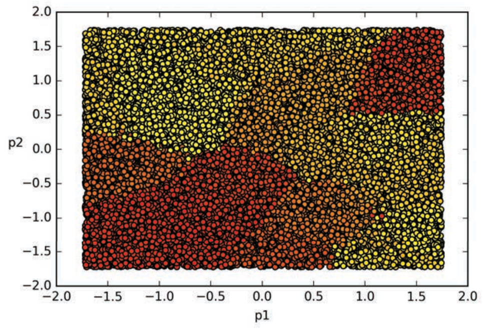


(d) 3D Visualization with weighted supervised component

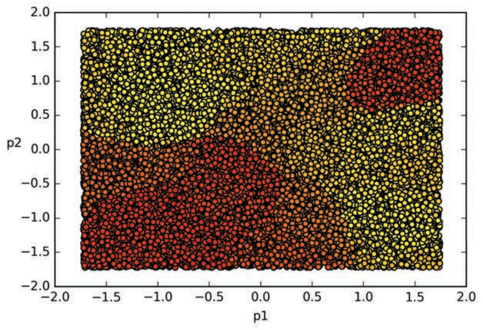
Figure 4.15: Visualization results for *Loc-lat* dataset



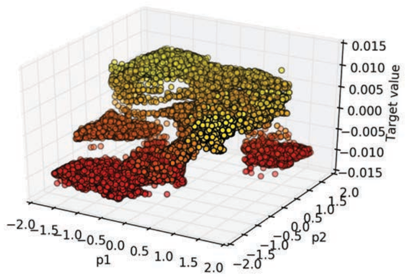
(a) Visualization without supervised component



(b) Visualization with supervised component



(c) Visualization with weighted supervised component



(d) 3D Visualization with weighted supervised component

Figure 4.16: Visualization results for *Loc-long* dataset

CHAPTER 5
EXTREME LEARNING MACHINES FOR SURFACE REGISTRATION

Deformable registration of surfaces is a fundamental problem in computer-aided design and computer graphics and is critical for a wide range of applications such as shape interpolation [BLL15, KMP07], statistical shape analysis and modeling [ACP03, BL12], geometry transfer [SP04], and so on. Among other problems, finding the point-wise correspondence between surfaces is the main challenge for the successful registration, but such a problem is known to be NP-hard. To this end, a well defined local shape descriptor and an accurate similarity metric can increase the performance and the success rate of a registration algorithm significantly, as they provide a tool for evaluating the similarity between two points computationally.

For the last decade, a family of spectral descriptors (e.g., [RWP06, Rus07, SOG09, ASC11b, BK10]) that utilizes the eigendecomposition of the Laplace-Beltrami operator [Ros97] has drawn lots of attention among the researchers in relevant fields. This is because the spectrum of the Laplace-Beltrami operator has many desirable properties, such as isometry invariance, multiscaleness, parametrization independence, and etc [RWP05, SOG09]. In fact, the spectral descriptors in general show a better performance compared to other types of descriptors [LGB$^+$13], especially for the tasks such as shape retrieval [LH14, BK10], mesh segmentation [ASC11a, FSKR11] and isometric matching [OMMG10]. However, despite of all the desirable properties, they often struggle in most of the surface registration tasks, especially when the disparity between the surfaces is relatively large. This is because the spectral descriptors are highly sensitive to the metric distortion. Therefore, even though they work generally well with the isometric registration tasks (e.g., surface scans of the same individual with different postures), but not so well with the deformable registration tasks (e.g., registering a skinny person to an obese person).

In order to overcome such a limitation of the spectral descriptors, it was proposed to find matching vertices on two compared models by the means of distance measure between vertices. In prior work [SGBL17], a novel approach using deep neural network for embedding spectral descriptors into a new lower dimensional space is introduced. This deep neural network is trained in a form of Siamese neural network in such a way that new embedding space of spectral descriptors can be considered as a metric space. In other words, calculating the Euclidean distance between spectral descriptors in the embedding space directly provides a suitable measure of similarity desirable for deformable registration tasks (Section 5.2.1). In the framework of this work it has also been noticed that measuring Euclidean distance between spectral descriptors before embedding also provides a measure of similarity. However, the results in the latter case are less impressive both in terms of absolute value of found correct matching vertices and their distribution over the model. Based on this observation, it is proposed to implement Extreme Learning Machines algorithm in order to learn an appropriate distance measure from original spectral descriptors, so that it becomes more suitable for registration tasks with non-isometric distortion (Section 5.2.2).

Dataset

For the experiments in this chapter, the *Dyna* dataset [PMRMB15] has been used to generate data samples. Original *Dyna* data set is composed of over 40,000 human body scans of ten subjects spanning a range of body shapes. The body scans are collected by using a custom-built multi-camera active stereo system, which captures 14 assigned motions (e.g. running, jumping, shaking, etc.) at 60 frames per second. The system outputs 3D meshes of a size around 150,000 vertices in average. Such dense sets of vertices are then registered by conforming a template mesh to each of the scanned meshes. After the registration, each mesh get to contains 6,980 vertices. More important, the topology of each mesh becomes compatible to each other, and the vertices with the same index get to correspond to each other geometrically. This information can be exploited by using it as the ground truth for the correspondence of a pair of vertices. On each of the meshes in the *Dyna* database, the spectral descriptors are computed in a way described in Section 5.1.

From the resulting dataset, two subsets (training and test sets) are formed in the following way. The training set is formed by 10,000 batches, where each batch contains 512 pairs of vertices from two randomly selected meshes. Vertices for pairs are also randomly selected and satisfy the following condition: 256 of pairs are pairs of matching vertices, i.e. vertices have the same indices on meshes, and 256 pairs are pairs of non-matching vertices. This scheme allows to have duplicates of vertices in the scope of one batch, while meshes that were used once for training set are removed from the pool of possible choices in the future. This means, that the dataset is divided in two equal parts: first half used for training and the rest data is used for test.

5.1 Spectral Shape Descriptors

A robust and informative local shape descriptor plays an important role in mesh registration. Spectral shape analysis is a branch of computational geometry that analyzes digital geometry using the spectrum of a linear operator defined on a surface. Among a variety of choices, the Laplace-Beltrami operator [Ros97] that generalizes the Laplacian to Riemannian manifolds has gained significant highlights because of its desirable properties. One such property is the isometry invariance of its eigenvalues [Lév06]. Since a lot of deformations in real-world can be characterized as an isometry or a near-isometry, the isometry invariance property of the eigenvalues of the Laplace-Beltrami operator gives advantage for many applications, such as shape retrieval [RWP05, JZ07, LH14, BK10], correspondence matching [DK10, OMMG10], segmentation [RBG$^+$09, ASC11a], and etc.

Reuter *et al.* [RWP06] defined a signature of a shape called, ShapeDNA, which essentially was a ascending sequence of eigenvalues of the Laplace-Beltrami operator. They encoded a given geometry data into a vector of several hundreds of dimensions containing the eigenvalues of the Laplace-Beltrami operator and showed the encoding was advantageous for shape retrieval and matching tasks.

Rustamov [Rus07] defined the global point signature (GPS) that also utilizes the eigendecomposition of the Laplace-Beltrami operator. He defined the GPS at each point on the surface as a vector containing the eigenfunctions of different modes scaled by the corresponding eigenvalues.

Sun *et al.* [SOG09] proposed the heat kernel signature (HKS) based on the different heat diffusion characteristics according to the geometric shape of the surface. They introduced the heat kernel equation that formulates the isotropic heat diffusion process on a manifold surface by the Laplace-Beltrami spectrum, and defined a shape signature using the heat kernel function. For each point on the surface, the heat kernel function is sampled at n different time scales, forming an n-dimensional feature vector. Later on, a lot of variations of HKS have been introduced, including the scale-invariant HKS (SIHKS) [BK10].

In the similar spirit of HKS, Aubry *et al.* [ASC11b] proposed the wave kernel signature (WKS) that is based on the characterization of the wave propagation on manifolds. The WKS represents the average probability of measuring a quantum mechanical particle at a specific location. This is achieved by solving the Shrödinger's equation, whose solution is represented by the Laplace-Beltrami spectrum.

Many relevant literature (see e.g., [LH14, LGB$^+$13]) reports that spectral descriptors outperform other types of shape representation methods for shape retrieval tasks in general, since they are invariant to the isometry and are proportional to a deformation, or a metric change. Especially, signatures such as HKS and WKS are also known to be multiscale in a sense that they inherently capture both the local and global shape characteristics through different time scales.

However, spectral descriptors are not quite suitable for the dense correspondence problems or deformable registration problems that involve large, non-isometric deformation. This is because, for non-isometric registration cases, the spectral descriptors tend to fail in recognizing the corresponding points between two models due to a large difference in local metrics.

The next paragraphs will describe implementation details of spectral shape descriptors that are to be used as inputs in the proposed neural network-based approach for surface registration. Implementation of each type of the spectral descriptors followed the standard parameters suggested in the original papers.

Global Points Signature

[Rus07]: Given a point on a 2-manifold, the GPS at the point x is defined as:

$$\mathrm{GPS}(x) = \left[\frac{\phi_1(x)}{\sqrt{\lambda_1}}, \frac{\phi_2(x)}{\sqrt{\lambda_2}}, \ldots, \frac{\phi_n(x)}{\sqrt{\lambda_n}}\right]^\top, \tag{5.1}$$

where λ_k and ϕ_k are the k-th eigenvalue and eigenfunction of the Laplace-Beltrami operator defined on the manifold respectively. An adequate number of eigenvalues suggested in [Rus07] is $n = 25$.

Heat Kernel Signature

[SOG09]: Given a 2-manifold, the heat flow from a point x to a point y on the manifold can be approximated by the heat kernel function:

$$H_t(x, y) = \sum_{k=0}^{\infty} e^{-\lambda_k t}\phi_k(x)\phi_k(y). \tag{5.2}$$

Physically, the function returns the amount of heat diffused from a point x to a point y on the manifold during a certain time t. From this, the HKS is defined as a series of heat kernel values $H_t(x, x)$ measured at discrete samples of time $t_1, t_2, \ldots, t_n$:

$$\mathrm{HKS}(x) = \left[H_{t_1}(x, x), H_{t_2}(x, x), ..., H_{t_n}(x, x)\right]^\top. \tag{5.3}$$

In [SOG09], the authors suggest using the first 300 eigenvalues and eigenvectors for the approximation of Equation 5.2. They also suggest uniformly sampling $n = 100$ time samples in logarithmic scale over the time interval from $4 \ln 10/\lambda_{300}$ to $4 \ln 10/\lambda_2$.

Wave Kernel Signature

[ASC11b]: Given a 2-manifold, the propagation of a quantum particle on the manifold is governed by the Schrödinger's wave function, whose solution is given as follows:

$$\psi_E(x,t) = \sum_{k=0}^{\infty} e^{i\lambda_k t}\phi_i(x) f_E(\lambda_k), \tag{5.4}$$

where i is the imaginary number and f_E^2 is an energy probability distribution with expectation value E. In practice, the energy probability distribution is approximated by the log-normal distribution, $e^{\frac{-(\rho - \ln \lambda_k)^2}{2\sigma^2}}$, where ρ is the energy scale.

Here, the l_2 norm of the $\psi_E(x,t)$ physically has a meaning that the probability of measuring the particle at a point x on the manifold at time t. The average probability is then achieved by integrating the norm over time:

$$P_\rho(x) = \lim_{T \to \infty} \frac{1}{T} \int_0^T \|\psi_E(x,t)\|^2 dt = \sum_{k=0}^{\infty} \phi_k^2(x) f_E^2(\lambda_k). \tag{5.5}$$

From this, the wave kernel signature is defined as a series of the probability values in different energy scales $\rho_1, \rho_2, \ldots, \rho_n$:

$$\mathrm{WKS}(x) = \left[P_{\rho_1}(x), P_{\rho_2}(x), ..., P_{\rho_n}(x)\right]^\top. \tag{5.6}$$

Similar to HKS, first 300 eigenvalues is used to approximate Equation 5.5, and the $n = 100$ energy scale values are uniformly sampled over an interval from $\ln(\lambda_1)$ to $\ln(\lambda_{300})$, in [ASC11b].

5.2 Matching Correspondence via Distance Measure

The goal of the presenting approach is to perform registration task between two models, more particular find point-to-point correspondence, by calculating the measure of distance between points. In prior work [SGBL17] this approach has been implemented by developing a deep neural network with a purpose of learning an embedding of original spectral descriptors into a lower dimensional metric space so that the Euclidean distance calculated between two points in this new space would represent a measure of similarity between corresponding vertices on the mesh. Based on the results of this work it is further proposed to train an Extreme Learning Machines model to learn directly from original spectral descriptors an appropriate distance metric with desirable suitability for deformable registration tasks.

5.2.1 Deep Spectral Descriptors

This section contains the description of a method proposed by Sun et al. [SGBL17]. The core idea of this method is to develop new shape descriptors by embedding the original spectral descriptors, namely, Global Point Signature (GPS), Heat Kernel Signature (HKS) and Wave Kernel Signature (WKS), into a new metric space such that the Euclidean distance between them directly provides a desirable similarity measure for deformable registration tasks. For this purpose, the spectral descriptors on each point of 3D models have been computed in advance, following the standard parameter selection scheme mentioned in Section 5.1. These spectral descriptors then serve as an input to the Siamese Neural Network, in which they are

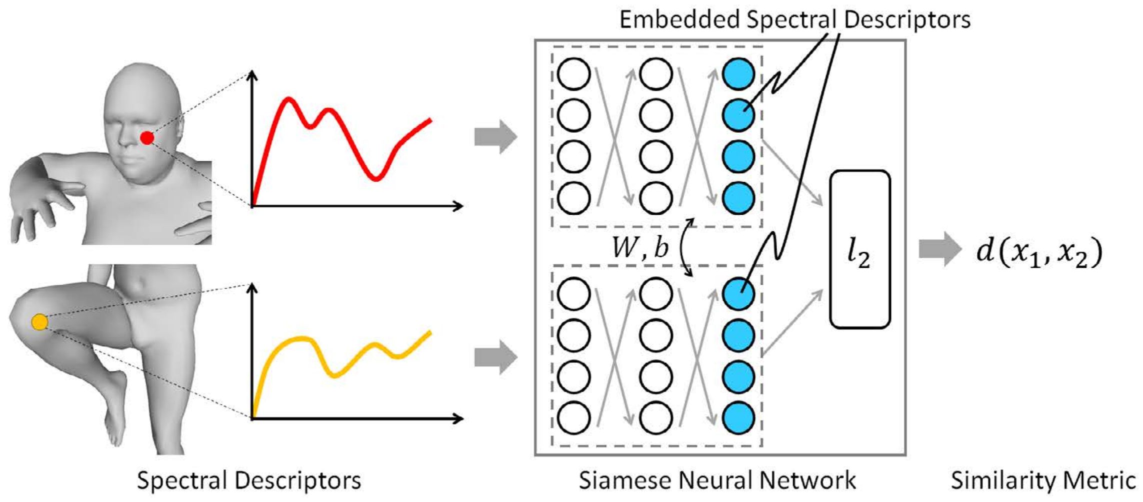


Figure 5.1: A schematic diagram of the Siamese architecture. The pair of the networks is identical, and shares the same coefficients (i.e., weights W, and bias b)

embedded into a new metric space. The dimensionality of the metric space has been chosen according to the results of the shape descriptors intrinsic dimensions analysis. The output of the trained deep Siamese neural network is the similarity measure as well as a mapping from the space of original shape descriptors to a new metric space. The result of the mapping has received the name of *deep spectral descriptors* after the deep neural network architecture of Siamese Neural Network used to learn the mapping. Figure 5.1 illustrates the main idea of obtaining deep spectral descriptors by means of Siamese neural network. For the sake of simplicity, in this work the similarity measure has been replaced by a mere Euclidean distance in [SGBL17].

Embedding Space

Originally, spectral descriptors are embedded in a canonical Euclidean n-space equipped with the Euclidean metric (i.e., l_2 norm). That is, each of the descriptors corresponds to a point in an n-dimensional Euclidean space, and the Euclidean distance between two points indicates the difference between the spectral descriptors, and thus, the geometric dissimilarity between the corresponding surface points.

Essentially, the goal of training the neural network is to find an embedding of the spectral descriptors in a different metric space, more suitable for deformable registration tasks than the canonical Euclidean n-space. Hence, it is critical to determine the dimensionality of the new embedding space in such a way that it preserves most of geometric information encoded in the original spectral descriptors while keeping the dimensionality as concise as possible. This has been realized by conducting an analysis on the local intrinsic dimensionality of the spectral descriptors by means of principal component analysis (PCA) [Kir00, WEG87]. For this analysis, 10,000 random samples were collected from the database. Then, for each sample the k-nearest neighborhood is determined and PCA is conducted on the set of k-nearest neighbors. To estimate the intrinsic dimensionality of the local tangent space, residual variances has been analyzed with respect to the number of principal components. Technically, the task is to find d number of first principal components that covers larger than 99% of the total variance, that is $\sum_i^d \lambda_i \geq 0.99 \sum_i^n \lambda_i$ where λ_i is the eigenvalue associated with the i-th principal component. This process is repeated multiple times to obtain statisically stable results. It has been noticed that residual variances drop signifi-

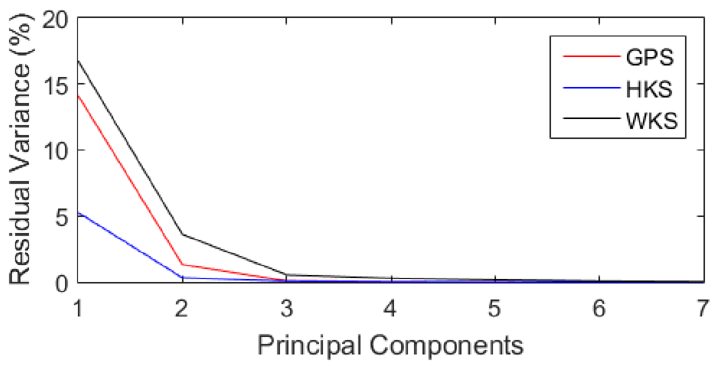


Figure 5.2: Intrinsic dimensionality analysis

cantly after the first five principal components, meaning that contribution of all the later components to the cumulative variance is insignificantly small (see Figure 5.2). Moreover, the same tendency has been observed for different types of spectral descriptors and for k number of nearest neighbors varying from 6 to 25. Therefore, it was conclude that the intrinsic dimension of the spectral descriptors in local area is 5. The dimensionality of the embedding space is set to 15 to give enough degrees of freedom for distortion of the data manifold.

Siamese Neural Network

A *Siamese* architecture [BBB+93] is an effective way of designing neural networks for comparative analysis. In Siamese network (Figure 5.1), two identical neural networks (called Siamese pair) are placed in parallel, and the output layers of these networks are merged and are fed into another layers of neural network. Neural networks in the Siamese pair share the same coefficients such that the weight values of

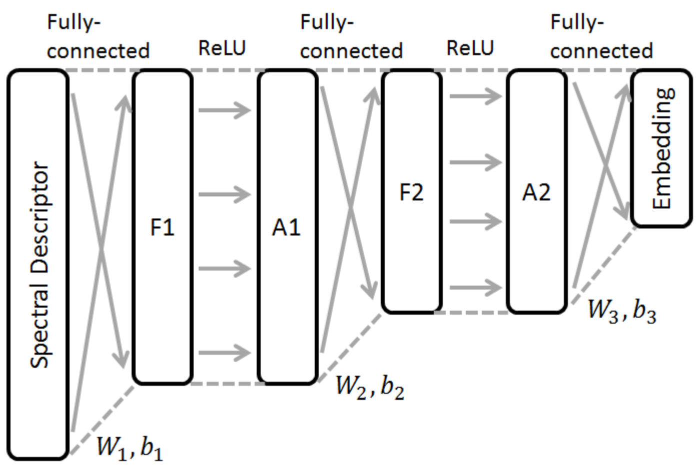


Figure 5.3: The structure of 5-layer deep neural network, used as one of Siamese pair

the neurons and the biases are all identical between the pairs. The outputs of the Siamese pair, which mathematically are the spectral descriptors embedded into a different metric space, are then compared using Euclidean metric (l_2 distance), which provides a measure of similarity. Architecture of the Siamese pair neural networks (Figure 5.3) has been designed in order to ensure that Siamese neural network learns the optimal embedding of spectral descriptors. Finally, it has been chosen that each neural network in Siamese pair has two fully connected hidden layers with 78 and 32 neurons for HKS and WHS, and 20 and 18 for GPS spectral descriptors, respectively. The particular model selection process is described with more details in [SGBL17].

To train the presented Siamese neural network a training subset formed from the *Dyna* dataset as described at the beginning of Chapter 5 has been used, with a total number of 5,120,000 of training samples (batch size $\times$ number of batches). For the optimization, no significant difference between different types of optimization methods has been found, and the Adam optimizer [KB14] has been chosen as the most popular state-of-the-art method for training deep neural networks. The learning rate has been set initially equal to 0.015 with a decay factor of 0.9999 for each of the training steps. With respect to the approach of solving the original problem, the objective of the optimization is to maximize the margin between non-matching pairs. This can be achieved by minimizing the sum of the following error terms over all training samples k:

$$E(f_k, g_k) = y_k \|D(f_k) - D(g_k)\|^2 + (1 - y_k) \max\left(0, C - \|D(f_k) - D(g_k)\|^2\right), \quad (5.7)$$

where (f_k, g_k) is the k-th pair, $D(\cdot)$ is an embedding derived from the Siamese branch, y_k is a Boolean value indicating the correspondence with $y_k = 1$ for a pair of matching vertices and $y_k = 0$ for a pair of non-matching vertices. The constant C is the minimum margin value between the non-matching pairs, and it was set $C = 5$. One practical consideration that needs to be done for the selection of the Boolean value y_k is that even the non-matching pairs might have a highly similar geometry. For instance, a point on the left thumb would probably be highly similar in geometry with a point on the right thumb. However, they are technically non-matching pair. If such cases happen in the training data set, it may confuse the neural network since it is a sort of conflicting example. In this reason, instead of setting y_k strictly equal to 0 for non-matching pair, y_k is assigned some random value between 0 and 0.2 thus implementing soft classification. The correct matching pairs are still kept to $y_k = 1$.

Experiments

Performance of the proposed approach implemented by the means of Siamese neural network has been tested in a series of experiments. For each experiment, a pair of meshes has been randomly selected from the test subset of *Dyna* dataset. For a given pair of meshes, all pairwise distances between vertices have been calculated in the embedding space. In other words, for each vertex from Model 1 a deep spectral descriptor has been obtained using the trained Siamese neural network and Euclidean distance to the deep spectral descriptors of all vertices from Model 2 is calculated. From the resulting distance matrix a pair of matching vertices is determined as a pair of vertices with the smallest Euclidean distance between deep spectral descriptors. The pair of matching vertices is said to be correctly defined if it satisfies two conditions: the Euclidean distance should not be over the threshold value equal to $0.5C$, and the geodesic distortion between vertices should not be more than 5% of the shape diameter.

Testing results for 6 pairs of different meshes are presented in Table 5.1 and Figure 5.4, with Table 5.1 containing the number of correct matching pairs calculated for each pair of models, and Figure 5.4 visually depicting them. D in the name of shape descriptors stands for *deep spectral descriptor*. For the sake of comparison, original spectral descriptors have also been included in the experiments. In their case, the Euclidean distance has been calculated directly for the corresponding spectral descriptors.

In general, a significant improvement with the proposed method can be ob-

Table 5.1: Statistics of the number of correct matching pairs for the results presented in Figure 5.4. For each experiment, Model 1 is presented by a yellow figure, and Model 2 by a blue one. Each of the rows in the table corresponds to each of the rows in Figure 5.4 in the same order. The left two entries of each row show ID of corresponding models

Model 1	Model 2	GPS	HKS	WKS	DGPS	DHKS	DWKS
00455	00228	0	211	669	631	1,585	706
00165	00170	147	2,209	2,124	2,374	2,300	1,325
00090	00266	84	1,145	1,328	3,478	2,654	1,595
00168	00374	25	627	1,072	1,072	2,229	1,794
00233	00302	206	455	1,162	998	1,637	1,684
00515	00214	17	1,474	1,943	921	1,975	1,565

served in terms of the absolute numbers of correct matching pairs. It should be noted that even for the cases when the improvement in the absolute number is not so significant or even negative, the quality of matching still has been improved in a sense that the pairs in deep signature results cover larger area then the original descriptors (see Figure 5.4).

5.2.2 Learning Similarity Metric with ELM

Results of the prior work [SGBL17] briefly described in Section 5.2.1 have proved the consistency of the proposed approach: learning the optimal similarity metric can improve the ability of spectral descriptors to perform deformable shape registration tasks. It has been shown that through careful design of a Siamese deep neural network, it was able to find an optimal embedding of the spectral descriptors

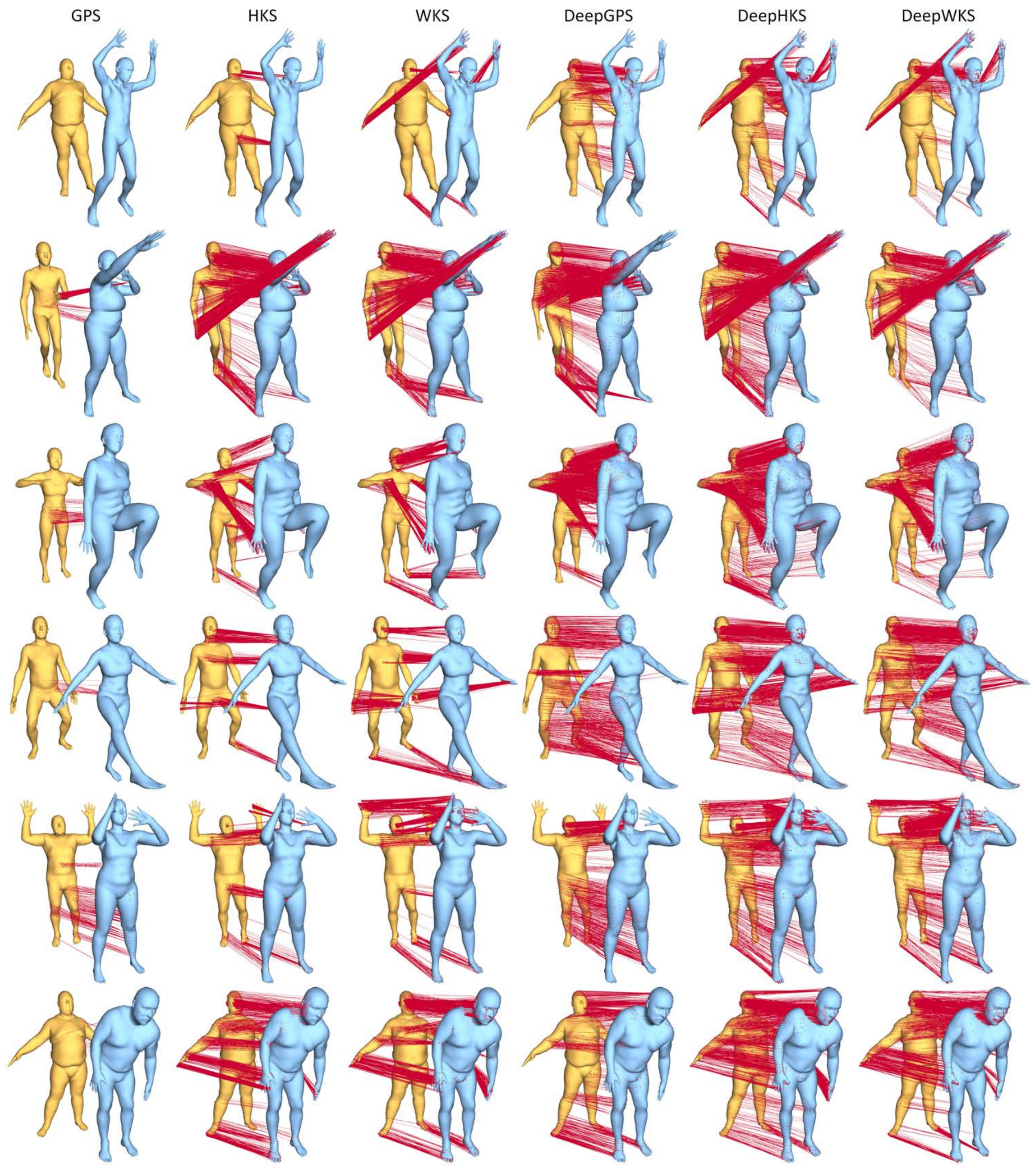


Figure 5.4: Visualization of the matches within the geodesic distortion of 5% of the shape diameter. Each row shows a comparison between the GPS, HKS, and WKS, and their embeddings, namely DeepGPS, DeepHKS, and DeepWKS, respectively

in a new metric space, in which a direct comparison based on the Euclidean distance already gives an excellent point-wise matching result between highly non-isometric cases.

Though the advantage of deep spectral descriptors over original spectral descriptors is certain, in some experiments the improvement is not so significant. In the further discussion on the results it was suggested that this phenomenon can be explained by a mere fact that Euclidean distance is not an appropriate metric for similarity analysis of spectral descriptors in the original space. As a result, it is proposed to implement an Extreme Learning Machines algorithm – a well-established technique for various regression and classification tasks – in order to learn a new similarity metric for shape descriptors in the original space, so that it would be suitable for deformable shape registration problem. In prior work this task has been performed in two steps: at first, a mapping function $g(x)$ of spectral descriptors from the original to the embedded space has been learned, and then, a distance metric has been applied to the embedded spectral descriptors as $d(g(x), g(y))$. The proven universal approximation property of ELM guarantees that it can learn this highly non-linear function $f(x, y) = d(g(x), g(y))$ (see Equation (2.6)). To prove the described hypothesis it is proposed to use Heat Kernel Signature spectral descriptor as an input of the ELM model, as the spectral descriptor that showed the best results in prior work both in the original and embedded spaces.

Implementation

In order to implement the Extreme Learning Machines model so it becomes available to properly learning the similarity metric, the training subset of the *Dyna* dataset has been altered along with the training process used for deep spectral descriptors: instead of simultaneously feeding shape descriptors corresponding to a pair of vertices to the neural networks in the Siamese pair, at first, these shape descriptors are merged together to form a single data sample with a doubled number of dimensions. This trick keeps the same size of the training data but makes it suitable to be fed to the ELM model.

The goal of the proposed ELM model is to learn an optimal similarity metric for spectral descriptors in the original space. In this case, it becomes intuitive to assign target values in the following way

$$y_k = \begin{cases} 0 & x_k^1 \equiv x_k^2 \\ 0.8 \sim 1 & x_k^1 \not\equiv x_k^2, \end{cases} \tag{5.8}$$

where x_k^1 and x_k^2 represent the first and second part of the k-th sample, respectively, i.e. shape descriptors of the first and second vertices in k-th pair. As a result, Extreme Learning Machines model is trained to directly compute the measure of similarity between a pair of shape descriptors, in contrast to the deep shape descriptors approach, where Siamese neural network was trained using the ground truth correspondence information. It should be noted, that such assignment of target variable preserves the soft classification assumption made for deep spectral descriptors, though no margin value is introduced in the ELM model. The threshold value has been defined as equal to 0.5, so that a pair of vertices with a similarity measure less or equal 0.5 should be considered as a matching pair.

Model Selection

The main parameter in the ELM model that influence its performance is the number of neurons in the hidden layer. In order to estimate the optimal number of hidden neurons, validation has been performed. When constructing the validation subset 1,000 batches has been used and the same policy as for the construction of training subset has been applied. The analysis of the validation subset results has shown that the model does not overfit even when the maximum number of hidden neurons is set equal to 40,000 (this number depends on the amount of available RAM in the system; for current implementation a system with 256GB of RAM has been used).

In order to evaluate the performance of the ELM model the summation of three parameters has been used: mean square error (MSE), false-positive rate (FPR) – ratio of the *negative* samples (non-matching pairs) that were predicted as *positive* (matching pairs), and false-negative rate (FNR) - ratio of the *positive* samples that have been predicted as *negative.* The left subfigure of Figure 5.5 presents the performance of the trained ELM model for the validation subset, in terms of three evaluation criteria. The right subfigure shows how the number of the hidden neurons influence the performance of the ELM model for the training and validation sets, in terms of combination of evaluation criteria. It can be observed, that the error continue

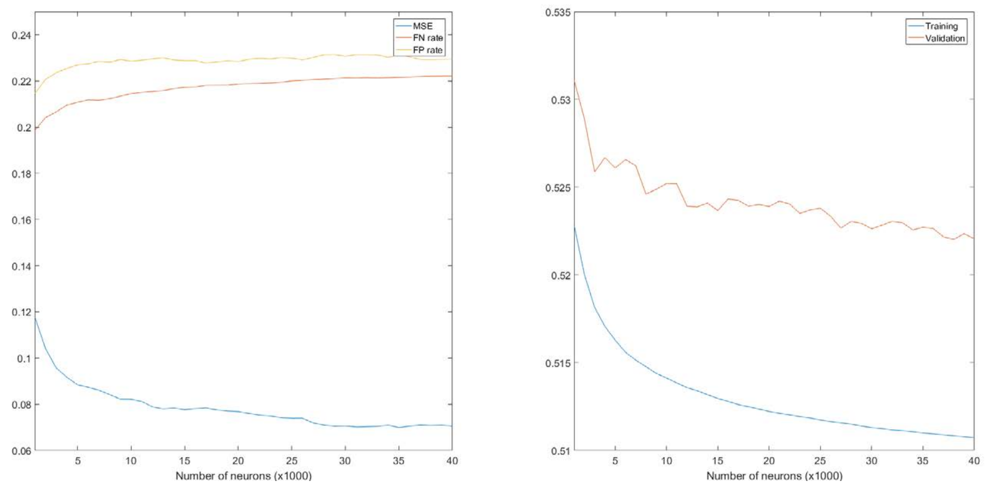


Figure 5.5: Influence of the number of hidden neurons in the ELM model on its performance. The relationship between the number of hidden neurons and values of the evaluation parameters (MSE, FNR, FPR), computed for the validation set, is shown on the left. For both training and validation sets, the decrease of the error, calculated as the sum of evaluation parameters, is depicted on the right

to decrease and the model never overfit, even when the number of hidden neurons reaches the maximum value. This behavior of ELM model is common for Big Data problems with huge number of data samples, and the size of the model, i.e. number of hidden neurons, is restricted only by the size of the available memory [ABML15]. Based on the analysis of the validation results and this reasoning, it is suggested to use the maximum possible number of hidden neurons that is equal to 40,000.

Experiments

To test the performance of the trained Extreme Learning Machines model, the same subset as for deep spectral descriptors has been used. The testing process has also been adopted from the one used for deep spectral descriptors with the only difference in forming data samples: as for the construction of the training subset, each pair of spectral descriptors has been merged to form a single data sample. Because only one spectral descriptor, Heat Kernel Signature, has been used to train the Extreme Learning Machines model, the comparison analysis has been conducted only between original HKS with Euclidean distance as similarity metric, DeepHKS with Euclidean distance as similarity metric, and original HKS with similarity metric learned by ELM (referred as ELMHKS). The results of 6 experiments with different 3D models are presented both in terms of absolute number of correctly determined matching pairs (see Table 5.2) and visually (see Figure 5.6). Again, in order to be correctly determined a matching pair has to satisfy two conditions: the similarity measure should be less or equal than the threshold value equal to 0.5, and the geodesic distortion should not exceed 5% of the shape diameter.

5.2.3 Discussions

From the results presented in Sections 5.2.1 and 5.2.2, it can be observed that performance of the registration task for pairs of models can be increased by introducing a reliable distance metric robust to non-isometric shape deformations. To obtain this new distance metric two approaches are proposed and compared. The first

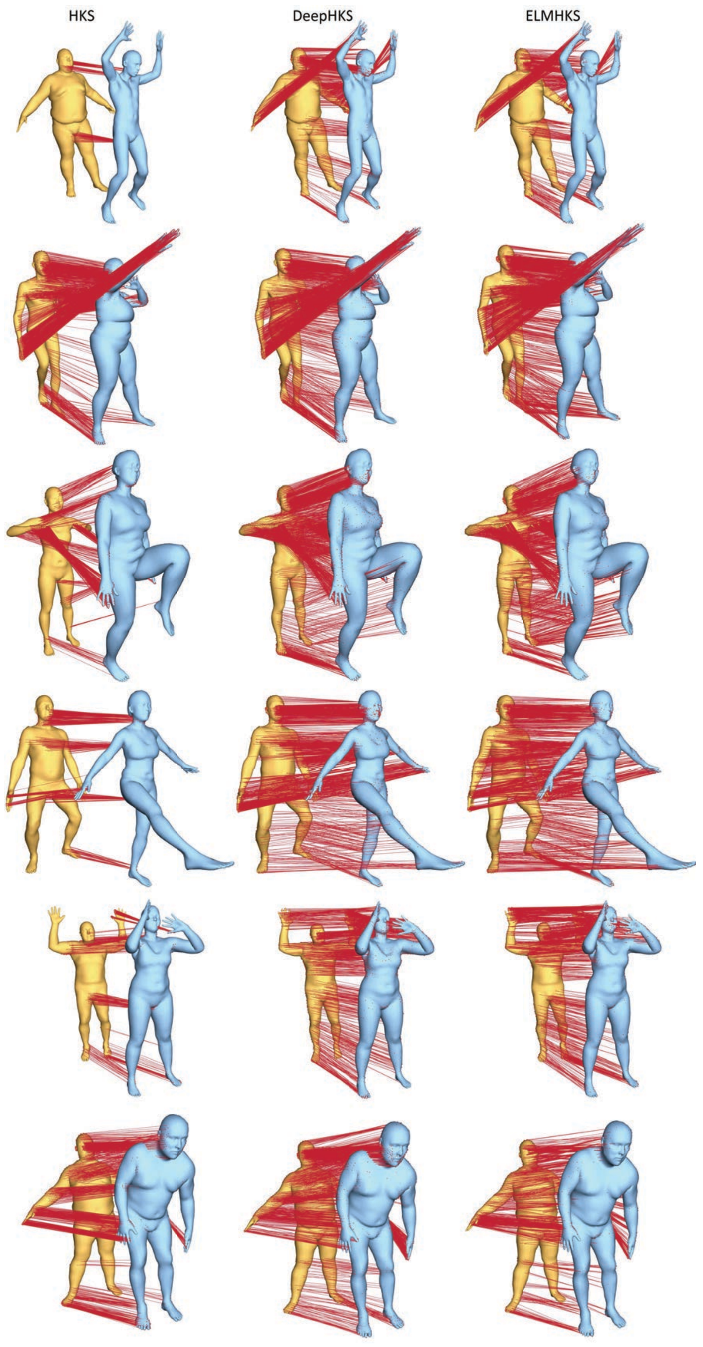


Figure 5.6: Visualization of the matches within the geodesic distortion of 5% of the shape diameter. Each row shows a comparison between the HKS, its embedding DeepHKS, and ELMHKS, respectively

Table 5.2: Statistics of the number of correct matching pairs for the results presented in Figure 5.6. For each experiment, Model 1 is presented by a yellow figure, and Model 2 by a blue one. Each of the rows in the table corresponds to each of the rows in Figure 5.6 in the same order. The left two entries of each row show ID of corresponding models

Model 1	Model 2	HKS	DHKS	ELMHKS
00455	00228	211	1,585	1,228
00165	00170	2,209	2,300	2,769
00090	00266	1,145	2,654	2,718
00168	00374	627	2,229	2,350
00233	00302	455	1,637	1,892
00515	00214	1,474	1,975	1,633

approach initially embeds original spectral shape descriptors into a new lower dimensional space (this embedding got a name of deep spectral descriptors), where euclidean distance is calculated between vertices to find matching pairs (see Section 5.2.1). For the second approach, the Extreme Learning Machines model is suggested to learn the appropriate distance metric directly in the original spectral descriptors space (see Section 5.2.2).

The comparison of these two approaches is performed by the example of Heat Kernel Signature spectral descriptor. Although both of the approaches show increase in the number of matching pairs, in four out of six cases ELMHKS method shows better results than DeepHKS. This can possibly happen due to the higher number of different types of parameters in the DeepHKS model and not all of them were tuned to the optimal values. Additionally, Euclidean distance that is used in DeepHKS model may be not the best choice of distance measure for the family of spectral shape descriptors, though it is used in the process of finding the optimal embedding for spectral descriptors. On the other hand, when designing the ELMHKS model the results of the research on the intrinsic dimensionality of spectral descriptors has not been considered. As a future research in this direction, the development of a new method can be proposed, which would combine the advantages of two approaches presented in this chapter: learn both the optimal embedding of spectral descriptors and the distance metric.

An interesting phenomenon can be observed when pairs of matching vertices are visualized (see Figures 5.4 and 5.6): regardless of the overall number of matching vertices between two models, there are certain regions on the models that consistently have lower number of matching vertices. These regions can be described as regions with relatively simple geometry (e.g. belly, tights, forearms). This phenomenon denotes a limitation of spectral descriptors that are designed to focus on capturing local geometry features. In conclusion, it should be stated that successful performance of shape registration task with non-isometric deformations should not rely solely on spectral descriptors but implement other techniques that will compensate the noted limitation of spectral descriptors and strengthen the reliability of the proposed methods.

CHAPTER 6
CONCLUSIONS

Extreme Learning Machines stand for a state-of-the-art machine learning technique of training a single hidden layer neural network. The core property of this technique is that it provides a solution for a linear system, whose components essentially are non-linear combination of input parameters. This thesis is dedicated to the solution of advanced applications in the field of machine learning and its intersection with other areas, like computational geometry and data visualization, mainly by means of Extreme Learning Machines (ELMs). With this, ELMs are able to combine linear and non-linear methods and take advantages from both of them: linear methods allows ELMs to provide a robust and fast unique solution, while nonlinearity in the ELM model guarantees that this solution satisfies universal approximation property. This means that ELMs are able to approximate any function with any precision, or, in terms of machine learning, ELMs are able to learn any underlying patterns or dependencies in the data. Additionally, considering linear nature of ELMs model, it becomes possible to implement different computational tricks to tune this model, e.g. use closed-form solution of Leave-One-Out error to find the optimal number of hidden neurons and thus avoid overfitting.

Since its invention, ELMs have proved they usefulness and ability to solve different well-studied tasks in the field of machine learning, like regression, classification (binary and multi-class), image recognition, etc. This thesis is focused on finding solution for several more advanced problems, found in the intersection of machine learning and other fields of study.

The first part of this thesis is focused on finding solution to the task of obtaining probabilistic outputs in multi-class classification problems. Outputs of the modern methods used in multi-class classification tasks do not possess characteristics of probabilities, even though for some methods output values may sum up to 1. At the same time, having probabilistic outputs can be very useful in applications where misclassification penalty differ a lot depending on the confused classes, or when results of classification are used part of a high-level decision-making pipeline. One of the solutions described in this thesis proposes a combination of ELMs algorithm with unsupervised clustering method with probabilistic outputs - Gaussian Mixture Model (GMM). Using ELMs as a preprocessing technique for the input data allows to overcome the main drawback of the latter method. The resulting model can be viewed as a tool of building a multi-dimensional Gaussian distribution to represent each of the presented classes (Section 3.1). The idea behind the second method is to derive probabilities from the statistics of classifier's output values using two types of histograms for each class - *in-class* (built from output values, corresponding to the ground truth class) and *out-class* (built from all other output values) histograms (Section 3.2). Both methodologies were tested on several different datasets and showed results comparative to the ones of base methods.

The second part discusses the opportunity of combining supervised learning approach with data visualization in order of obtaining more intuitively understandable visualization. In many areas of our everyday life, like medicine, business, education, sports, etc., information visualization techniques are used for one main purpose - data analysis, or sense-making, i.e. try to understand or discover underlying patterns and dependencies in data from its visual representation. As dimensionality of data increase, this task becomes more complicated due to inevitable information loss. To minimize information loss more sophisticated methods are to be used. As a result, the visualization becomes less intuitive and more difficult for the analysis. The method described in this thesis proposes to introduce a supervised component (class label or function value in classification or regression tasks, respectively) to the unsupervised scheme of data visualization techniques (Sections 4.2 and 4.3). The underlying incentive for this is too stimulate visualization algorithm to project samples from one class together and penalize it if they are projected far apart. When applied to ELMVIS+ – Extreme Learning Machines-based nonlinear data visualization technique, this methodology showed improvement not only in terms visualization per se, but in terms of cost function as well.

In the last part of this thesis, an application of machine learning methods to the area of computational geometry and computer graphics is described, namely, the development of a new similarity metric for shape descriptors able to advance shape registration and correspondence measure of surfaces with non-isometric distortions. The problem of deformable shape registration stays open even with the appliance of modern shape descriptors, especially for non-isometric cases where the metric distortion between 3D models is large. In the scope of the performed research, two approaches are proposed. The first approach introduces the development of a new type of shape descriptors for which a simple distance metric, like Euclidean distance, can perform as a reliable measure for point-to-point correspondence task. Using several well-known spectral shape descriptors and deep neural network with Siamese architecture, a new descriptor is presented as an embedding of original shape descriptor in the lower dimensional space (Section 5.2). The second approach presents a way of learning a similarity measure for spectral descriptors by means of Extreme Learning Machines. In a series of experiments, it has been proven that using a proper similarity metric, either learned directly for certain spectral descriptors or designed by embedding spectral descriptors in a special metric space, can provide much better results in finding the point-wise correspondence between 3D surfaces with large metric distortion, which is an important task in surface registration problem.

This thesis provides solution to several advanced problems at the intersection of machine learning and other fields of study, but at the same time it opens discussion: How these proposed methods can be improved to overcome their drawbacks? What are the next steps to be taken to advance in the designated directions? Some of these questions have already been addressed in the scope of this thesis, e.g. what new metric should be used to treat probabilistic outputs in multi-class classification tasks, and I'm planning to answer them and continue searching for new applications that could be solved by means of machine learning methods, and Extreme Learning Machines in particular since it is a powerful technique that is yet to meet its prime in the world of Big Data.

REFERENCES

[ABE$^+$55] Miriam Ayer, H. D. Brunk, G. M. Ewing, W. T. Reid, and Edward Silverman, *An Empirical Distribution Function for Sampling with Incomplete Information*, The Annals of Mathematical Statistics **26** (1955), no. 4, 641–647.

[ABM$^+$17] Anton Akusok, Stephen Baek, Yoan Miche, Kaj-Mikael Björk, Rui Nian, Paula Lauren, and Amaury Lendasse, *ELMVIS+: Fast Nonlinear Visualization Technique based on Cosine Distance and Extreme Learning Machines*, Neurocomputing (forthcoming 2017), 247263.

[ABML15] Anton Akusok, Kaj-Mikael Björk, Yoan Miché, and Amaury Lendasse, *High-Performance Extreme Learning Machines: A Complete Toolbox for Big Data Applications.*, IEEE Access **3** (2015), 1011–1025.

[AC92] J.Scott Armstrong and Fred Collopy, *Error measures for generalizing about forecasting methods: Empirical comparisons*, International Journal of Forecasting **8** (1992), no. 1, 69 – 80.

[ACP03] Brett Allen, Brian Curless, and Zoran Popović, *The space of human body shapes: reconstruction and parameterization from range scans*, ACM transactions on graphics (TOG) **22** (2003), no. 3, 587–594.

[AEM$^+$17] Anton Akusok, Emil Eirola, Yoan Miche, Andrey Gritsenko, and Amaury Lendasse, *Advanced Query Strategies for Active Learning with Extreme Learning Machines*, The European Symposium on Artificial Neural Networks (ESANN), upcoming 2017.

[AGLM13] Anton Akusok, Alexander Grigorievskiy, Amaury Lendasse, and Yoan Miche, *Image-based Classification of Websites*, Machine Learning Reports 02/2013 (Saarbrücken, Germany) (Thomas Villmann and Frank-Michael Schleif, eds.), Machine Learning Reports, vol. ISSN: 18, Workshop of the GI-Fachgruppe Neuronale Netze and the German Neural Networks Society in connection to GCPR 2013, September 2013, pp. 25–34.

[AGM$^+$17] Anton Akusok, Andrey Gritsenko, Yoan Miche, Kaj-Mikael Björk, Rui Nian, Paula Lauren, and Amaury Lendasse, *Adding Reliability to ELM Forecasts by Confidence Intervals*, Neurocomputing (upcoming 2017), 232–241.

[AH14] Bálint Antal and András Hajdu, *An ensemble-based system for automatic screening of diabetic retinopathy*, Know.-Based Syst. **60** (2014), 20–27.

[Aku16] Anton Akusok, *ELMVIS+ code*, 2016.

[All74] David M. Allen, *The relationship between variable selection and data agumentation and a method for prediction*, Technometrics **16** (1974), no. 1, 125–127.

[AMB$^+$16] Anton Akusok, Yoan Miche, Kaj-Mikael Björk, Rui Nian, Paula Lauren, and Amaury Lendasse, *Proceedings of elm-2015 volume 2: Theory, algorithms and applications (ii)*, ch. ELMVIS+: Improved Nonlinear Visualization Technique Using Cosine Distance and Extreme Learning Machines, pp. 357–369, Springer International Publishing, 2016.

[AMH$^+$14] Anton Akusok, Yoan Miche, Jozsef Hegedus, Rui Nian, and Amaury Lendasse, *A Two-Stage Methodology Using K-NN and False-Positive Minimizing ELM for Nominal Data Classification*, Cognitive Computation **6** (2014), no. 3, 432–445.

[AMK$^+$15] Anton Akusok, Yoan Miche, Juha Karhunen, Kaj-Mikael Björk, Rui Nian, and Amaury Lendasse, *Arbitrary Category Classification of Websites Based on Image Content*, IEEE Computational Intelligence Magazine **10** (2015), no. 2, 30–41.

[ASC11a] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers, *Pose-consistent 3d shape segmentation based on a quantum mechanical feature descriptor*, Joint Pattern Recognition Symposium, 2011, pp. 122–131.

[ASC11b] ______, *The wave kernel signature: A quantum mechanical approach to shape analysis*, 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), 2011, pp. 1626–1633.

[ASSK00] Erin L. Allwein, Robert E. Schapire, Yoram Singer, and Pack Kaelbling, *Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers*, Journal of Machine Learning Research **1** (2000), 113–141.

[AVB$^+$14] Anton Akusok, David Veganzones, Kaj-Mikael Björk, Eric Séverin, Philippe du Jardin, Amaury Lendasse, and Yoan Miche, *ELM Clustering–Application to Bankruptcy Prediction–*, International work conference on TIme SEries, 2014, pp. 711–723.

[BBB$^+$93] Jane Bromley, James W. Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah, *Signature verification using a "siamese" time delay neural network*, IJPRAI **7** (1993), no. 4, 669–688.

[BC01] Sergio Bermejo and Joan Cabestany, *Oriented principal component analysis for large margin classifiers*, Neural Netw. **14** (2001), no. 10, 1447–1461.

[BDM12] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment problems*, Society for Industrial and Applied Mathematics, 2012.

[Bel78] John B. Bell, *Review of* "Solutions of Ill-posed Problems" by A. N. Tikhonov, V. Y. Arsenin, Mathematics of Computation **32** (1978), no. 144, 1320–1322.

[Ben00] Paul N. Bennett, *Assessing the calibration of naive bayes posterior estimates*, Tech. Report CMU-CS-00-155, Carnegie Mellon University, 2000.

[Bil98] Jeff Bilmes, *A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models*, Tech. report, International Computer Science Institute and Computer Science Division, University of California at Berkeley, 1998.

[Bis06] Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[BK10] Michael M Bronstein and Iasonas Kokkinos, *Scale-invariant heat kernel signatures for non-rigid shape recognition*, Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE, 2010, pp. 1704–1711.

[BL12] Seung-Yeob Baek and Kunwoo Lee, *Parametric human body shape modeling framework for human-centered product design*, Computer-Aided Design **44** (2012), no. 1, 56–67.

[BLL15] Seung-Yeob Baek, Jeonghun Lim, and Kunwoo Lee, *Isometric shape interpolation*, Computers & Graphics **46** (2015), no. 1, 257–263.

[BMLR12] Heni Bouhamed, Afif Masmoudi, Thierry Lecroq, and Ahmed Rebaï, *A new approach for bayesian classifier learning structure via k2 algorithm*, Emerging Intelligent Computing Technology and Applications (De-Shuang Huang, Phalguni Gupta, Xiang Zhang, and Prashan Premaratne, eds.), Communications in Computer and Information Science, vol. 304, Springer Berlin Heidelberg, 2012, pp. 387–393 (English).

[BN01] Mikhail Belkin and Partha Niyogi, *Laplacian eigenmaps and spectral techniques for embedding and clustering*, Advances in Neural Information Processing Systems, vol. 14, MIT Press, 2001, pp. 585–591.

[BN03] ______, *Laplacian eigenmaps for dimensionality reduction and data representation*, Neural computation **15** (2003), no. 6, 1373–1396.

[Bri50] G. W. Brier, *Verification of Forecasts Expressed in Terms of Probability*, Monthly Weather Review **78** (1950), 1–3.

[BSW98] Christopher M Bishop, Markus Svensén, and Christopher K I Williams, *GTM: The generative topographic mapping*, Neural computation **10** (1998), no. 1, 215–234.

[BTVG06] H Bay, T Tuylelaars, and L Van Gool, *Surf speeded up robust features*, 9th European Conference on Computer Vision **61** (2006), no. 2, 346 – 359.

[CCA$^+$09] Paulo Cortez, Antnio Cerdeira, Fernando Almeida, Telmo Matos, and Jos Reis, *Modeling wine preferences by data mining from physicochemical properties*, Decision Support Systems **47** (2009), no. 4, 547 – 553, Smart Business Networks: Concepts and Empirical Evidence.

[CHK$^+$13] Erik Cambria, Guang-Bin Huang, Liyanaarachchi Lekamalage Chamara Kasun, Hongming Zhou, Chi Man Vong, Jiarun Lin, Jianping Yin, Zhiping Cai, Qiang Liu, Kuan Li, Victor C.M. Leung, Liang Feng, Yew-Soon Ong, Meng-Hiot Lim, Anton Akusok, Amaury Lendasse, Francesco Corona, Rui Nian, Yoan Miche, Paolo Gastaldo, Rodolfo Zunino, Sergio Decherchi, Xuefeng Yang, Kezhi Mao, Beom-Seok Oh, Jehyoung Jeon, Kar-Ann Toh, Andrew Beng Jin Teoh, Jaihie Kim, Hanchao Yu, Yiqiang Chen, and Junfa Liu, *Extreme Learning Machines*, IEEE Intelligent Systems **28** (2013), no. 6, 30–59.

[CV95] Corinna Cortes and Vladimir Vapnik, *Support-Vector Networks*, Machine Learning **20** (1995), no. 3, 273–297.

[DB03] G. Dahlquist and Å. Björck, *Numerical methods*, Dover Books on Mathematics, Dover Publications, 2003.

[DK10] Anastasia Dubrovina and Ron Kimmel, *Matching shapes by eigendecomposition of the laplace-beltrami operator*, Proc. 3DPVT, vol. 2, 2010.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin, *Maximum Likelihood from Incomplete Data via the EM Algorithm*, Journal of the Royal Statistical Society. Series B (Methodological) **39** (1977), no. 1, pp. 1–38 (English).

[DSDY13] George Dahl, Jack W. Stokes, Li Deng, and Dong Yu, *Large-scale malware classification using random projections and neural networks*, Proceedings IEEE Conference on Acoustics, Speech, and Signal Processing, IEEE SPS, May 2013.

[DSL$^+$03] S. Dablemont, G. Simon, A. Lendasse, A. Ruttiens, F. Blayo, and M. Verleysen, *Time series forecasting with SOM and local non-linear models - application to the DAX30 index prediction*, Proceedings of the Workshop on Self-organizing Maps (Hibikino, Japan), September 11-14 2003, pp. 340–345.

[DT05] Navneet Dalal and Bill Triggs, *Histograms of oriented gradients for human detection*, In CVPR, 2005, pp. 886–893.

[DZC$^+$14] Etienne Decencire, Xiwei Zhang, Guy Cazuguel, Bruno Lay, Batrice Cochener, Caroline Trone, Philippe Gain, Richard Ordonez, Pascale Massin, Ali Erginay, Batrice Charton, and Jean-Claude Klein, *Feedback on a publicly distributed database: the Messidor database*, Image Analysis & Stereology **33** (2014), no. 3, 231–234.

[EGA$^+$15] Emil Eirola, Andrey Gritsenko, Anton Akusok, Kaj-Mikael Björk, Yoan Miche, Du**v**san Sovilj, Rui Nian, Bo He, and Amaury Lendasse, *Extreme Learning Machines for Multiclass Classification: Refining Predictions with Gaussian Mixture Models*, Advances in Computational Intelligence, Lecture Notes in Computer Science, vol. 9095, Springer International Publishing, 2015, pp. 153–164.

[ELVB14] Emil Eirola, Amaury Lendasse, Vincent Vandewalle, and Christophe Biernacki, *Mixture of gaussians for distance estimation with missing data*, Neurocomputing **131** (2014), 32–42.

[FGA$^+$11] Pedregosa Fabian, Varoquaux Gal, Gramfort Alexandre, Michel Vincent, Thirion Bertrand, Grisel Olivier, Blondel Mathieu, Prettenhofer Peter, Weiss Ron, Dubourg Vincent, Vanderplas Jake, Passos Alexandre, Cournapeau David, Brucher Matthieu, Perrot Matthieu, and Duchesnay Édouard, *Scikit-learn: Machine learning in python*, Journal of Machine Learning Research **12** (2011), 28252830.

[FS78] I Fogel and D Sagi, *Gabor filters as texture discriminator*, Biological Cybernetics **61** (1978), no. 2, 103–113.

[FSKR11] Yi Fang, Mengtian Sun, Minhyong Kim, and Karthik Ramani, *Heat-mapping: A robust approach toward perceptually consistent mesh segmentation*, Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE, 2011, pp. 2145–2152.

[GAB$^+$17] Andrey Gritsenko, Anton Akusok, Stephen Baek, Yoan Miche, and Amaury Lendasse, *ELMVIS++R – Mastering Visualization with Target Variables*, Cognitive Computation (upcoming 2017), 1 – 22.

[GAM$^+$16] Andrey Gritsenko, Anton Akusok, Yoan Miche, Kaj-Mikael Björk, Stephen Baek, and Amaury Lendasse, *Combined Nonlinear Visualization and Classification: ELMVIS++C*, The 2016 International Joint Conference on Neural Networks (IJCNN), IEEE, July 2016, pp. 2617–2624.

[GES$^+$16] Andrey Gritsenko, Emil Eirola, Daniel Schupp, Edward Ratner, and Amaury Lendasse, *Probabilistic Methods for Multiclass Classification Problems*, Proceedings of ELM-2015 Volume 2, Proceedings in Adaptation, Learning and Optimization, vol. 7, Springer International Publishing, 2016, pp. 375–397.

[GES$^+$17] ———, *Solve Classification Tasks with Probabilities. Statistically-Modeled Outputs*, Hybrid Artificial Intelligent Systems (HAIS 2017), Lecture Notes in Artificial Intelligence, Springer International Publishing AG, June 2017, pp. 293–305.

[GHP07] G. Griffin, A. Holub, and P. Perona, *Caltech-256 Object Category Dataset*, Tech. Report CNS-TR-2007-001, California Institute of Technology, 2007.

[GOV81] J Gerritsma, R Omnink, and A Versluis, *Geometry, resistance and stability of the delft systematic yacht hull series*, International Shipbuilding Progress **28** (1981), no. 328, 276–297.

[Hay98] Simon Haykin, *Neural Networks: A Comprehensive Foundation (2nd Edition)*, 2nd ed., Prentice Hall, July 1998.

[HBKV15] Guang-Bin Huang, Zuo Bai, L.L.C. Kasun, and Chi Man Vong, *Local Receptive Fields Based Extreme Learning Machine*, IEEE Computational Intelligence Magazine **10** (2015), no. 2, 18–29.

[HCS06] Guang-Bin Huang, Lei Chen, and Chee-Kheong Siew, *Universal approximation using incremental constructive feedforward networks with random hidden nodes*, IEEE Transactions on Neural Networks **17** (2006), no. 4, 879–892.

[HK06] Rob J. Hyndman and Anne B. Koehler, *Another look at measures of forecast accuracy*, International Journal of Forecasting **22** (2006), no. 4, 679 – 688.

[HMZ$^+$06] Guang-bin Huang, Senior Member, Qin-yu Zhu, KZ Z Mao, Chee-kheong Siew, P Saratchandran, and Narashiman Sundararajan, *Can threshold networks be trained directly?*, IEEE Transactions on Circuits and Systems II: Express Briefs **53** (2006), no. 3, 187–191.

[HT98] Trevor Hastie and Robert Tibshirani, *Classification by pairwise coupling*, The Annals of Statistics **26** (1998), no. 2, 451–471.

[Hua14] Guang-Bin Huang, *An Insight into Extreme Learning Machines: Random Neurons, Random Features and Kernels*, Cognitive Computation **6** (2014), no. 3, 376–390.

[Hua15] ______, *What are Extreme Learning Machines? Filling the Gap Between Frank Rosenblatt's Dream and John von Neumann's Puzzle*, Cognitive Computation **7** (2015), no. 3, 263–278.

[HY01] David J. Hand and Keming Yu, *Idiot's bayes – not so stupid after all?*, International Statistical Review / Revue Internationale de Statistique **69** (2001), no. 3, 385–398.

[HZDZ12] Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang, *Extreme learning machine for regression and multiclass classification.*, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on **42** (2012), no. 2, 513–529.

[HZS04] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew, *Extreme learning machine: a new learning scheme of feedforward neural networks*, Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on, vol. 2, 25-29 July 2004, pp. 985–990 vol.2.

[HZS06] ———, *Extreme Learning Machine: Theory and Applications*, Neural NetworksSelected Papers from the 7th Brazilian Symposium on Neural Networks (SBRN '04)7th Brazilian Symposium on Neural Networks **70** (2006), no. 1–3, 489–501.

[JCMPK14] Luo Jiahua, Vong Chi-Man, and Wong Pak-Kin, *Sparse bayesian extreme learning machine for multi-classification*, Neural Networks and Learning Systems, IEEE Transactions on **25** (2014), no. 4, 836–843.

[Jol86] I.T. Jolliffe, *Principal Component Analysis*, Springer Verlag, 1986.

[JZ07] Varun Jain and Hao Zhang, *A spectral approach to shape-based retrieval of articulated 3d models*, Computer-Aided Design **39** (2007), no. 5, 398–407.

[KB14] Diederik Kingma and Jimmy Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980 (2014), 1–15.

[Kir00] Michael Kirby, *Geometric data analysis: an empirical approach to dimensionality reduction and the study of patterns*, John Wiley & Sons, Inc., 2000.

[KMC$^+$16] Jean-Claude Klein, Michel Menard, G. Cazuguel, C. Fernandez-Maloigne, G. Shaefer, P Gain, Batrice Cochener, Pascale Massin, Bruno Lay, and Béatrice Charton, *Methods to evaluate segmentation and indexing techniques in the field of retinal ophthalmology (MESSIDOR)*, 2016.

[KMP07] Martin Kilian, Niloy J. Mitra, and Helmut Pottmann, *Geometric modeling in shape space*, ACM SIGGRAPH 2007 Papers (New York, NY, USA), SIGGRAPH '07, ACM, 2007.

[Koh82] Teuvo Kohonen, *Self-organized formation of topologically correct feature maps*, Biological Cybernetics **43** (1982), no. 1, 59–69.

[Kru64] Joseph B Kruskal, *Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis*, Psychometrika **29** (1964), no. 1, 1–27.

[KTG12] Heysem Kaya, Pınar Tüfekci, and Fikret S Gürgen, *Local and global learning methods for predicting power of a combined gas & steam turbine*, Proceedings of the International Conference on Emerging Trends in Computer and Electronics Engineering (ICETCEE 2012), 2012, pp. 13–18.

[LBBH98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE **86** (1998), no. 11, 2278–2324.

[LC98] E. L. Lehmann and George Casella, *Theory of point estimation*, second ed., Springer Texts in Statistics, Springer-Verlag, New York, 1998. MR 1639875 (99g:62025)

[LC08] A. Lendasse and F. Corona, *Linear projection based on noise variance estimation: Application to spectral data*, Proceedings of ESANN 2008, European Symposium on Artificial Neural Networks, Bruges (Belgium) (M. Verleysen, ed.), d-side publ. (Evere, Belgium), April 23-25 2008, pp. 457–462.

[LCWV02] A. Lendasse, M. Cottrell, V. Wertz, and M. Verleysen, *Prediction of electric load using Kohonen maps - Application to the Polish electricity consumption*, Proceedings of the American Control Conference, 2002., vol. 5, 2002, pp. 3684–3689 vol.5.

[Lév06] Bruno Lévy, *Laplace-beltrami eigenfunctions towards an algorithm that" understands" geometry*, Shape Modeling and Applications, 2006. SMI 2006. IEEE International Conference on, IEEE, 2006, pp. 13–13.

[LGB$^+$13] Zhouhui Lian, Afzal Godil, Benjamin Bustos, Mohamed Daoudi, Jeroen Hermans, Shun Kawamura, Yukinori Kurita, Guillaume Lavoué, Hien Van Nguyen, Ryutarou Ohbuchi, et al., *A comparison of methods for non-rigid 3d shape retrieval*, Pattern Recognition **46** (2013), no. 1, 449–461.

[LH14] Chunyuan Li and A Ben Hamza, *Spatially aggregating spectral descriptors for nonrigid 3d shape retrieval: a comparative survey*, Multimedia Systems **20** (2014), no. 3, 253–281.

[Lic13] M. Lichman, *UCI Machine Learning Repository*, 2013, `http://archive.ics.uci.edu/ml`.

[LKT$^+$13] D. D. Lucas, R. Klein, J. Tannahill, D. Ivanova, S. Brandon, D. Domyancic, and Y. Zhang, *Failure analysis of parameter-induced simulation crashes in climate models*, Geoscientific Model Development **6** (2013), no. 4, 1157–1171.

[LV07] John A Lee and Michel Verleysen, *Nonlinear dimensionality reduction*, Springer, 2007.

[MAHN14] Yoan Miche, Anton Akusok, Jozsef Hegedus, and Rui Nian, *A Two-Stage Methodology using K-NN and False Positive Minimizing ELM for Nominal Data Classification*, Cognitive Computation (2014), 1–26.

[MBJ$^+$08] Yoan Miche, Patrick Bas, Christian Jutten, Olli Simula, and Amaury Lendasse, *A Methodology for Building Regression Models using Extreme Learning Machine: OP-ELM*, Proceedings of the European Symposium on Artificial Neural Networks (ESANN), 2008, pp. 247–252.

[MG63] A.M.F. Mood and F.A. Graybill, *Introduction to the theory of statistics*, International Student Edition: McGraw-Hill Series in Probability and Statistics, McGraw-Hill Book Company, Incorporated, 1963.

[MK97] Geoffrey McLachlan and Thriyambakam Krishnan, *The EM algorithm and extensions*, Wiley Series in Probability and Statistics, John Wiley & Sons, New York, 1997.

[MP00] Geoffrey J. McLachlan and David Peel, *Finite Mixture Models*, Wiley Series in Probability and Statistics, John Wiley & Sons, New York, 2000.

[MSB$^+$10] Yoan Miche, Antti Sorjamaa, Patrick Bas, Olli Simula, Christian Jutten, and Amaury Lendasse, *OP-ELM: Optimally Pruned Extreme Learning Machine*, IEEE Transactions on Neural Networks **21** (2010), no. 1, 158–162.

[MSL08] Yoan Miche, Antti Sorjamaa, and Amaury Lendasse, *OP-ELM: Theory, Experiments and a Toolbox*, Artificial Neural Networks - ICANN 2008 (Véra Kůrková, Roman Neruda, and Jan Koutnik, eds.), Lecture Notes in Computer Science, vol. 5163, Springer Berlin Heidelberg, 2008, pp. 145–154 (English).

[MSML10] Paul Merlin, Antti Sorjamaa, Bertrand Maillet, and Amaury Lendasse, *X-SOM and L-SOM: A double classification approach for missing value imputation*, Neurocomputing **73** (2010), no. 79, 1103 – 1108.

[MvB$^+$11] Yoan Miche, Mark van Heeswijk, Patrick Bas, Olli Simula, and Amaury Lendasse, *TROP-ELM: A double-regularized ELM using LARS and Tikhonov regularization*, Advances in Extreme Learning Machine: Theory and ApplicationsBiological Inspired Systems. Computational and Ambient IntelligenceSelected papers of the 10th International Work-Conference on Artificial Neural Networks (IWANN2009) **74** (2011), no. 16, 2413–2421.

[MvHB$^+$11] Yoan Miche, Mark van Heeswijk, Patrick Bas, Olli Simula, and Amaury Lendasse, *TROP-ELM: A double-regularized ELM using LARS and Tikhonov regularization*, Neurocomputing **74** (2011), no. 16, 2413–2421.

[Mye90] R.H. Myers, *Classical and Modern Regression with Applications*, Bookware Companion Series, PWS-KENT, 1990.

[OLG07] I Ortigosa, R Lopez, and J Garcia, *A neural networks approach to residuary resistance of sailing yachts prediction*, Proceedings of the international conference on marine engineering MARINE, vol. 2007, 2007, p. 250.

[OMMG10] Maks Ovsjanikov, Quentin Mérigot, Facundo Mémoli, and Leonidas Guibas, *One point isometric matching with the heat kernel*, Computer Graphics Forum **29** (2010), no. 5, 1555–1564.

[OWN97] A.V. Oppenheim, A.S. Willsky, and S.H. Nawab, *Signals and systems*, Prentice-Hall signal processing series, Prentice Hall, 1997.

[Pla99] John C. Platt, *Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods*, ADVANCES IN LARGE MARGIN CLASSIFIERS, MIT Press, 1999, pp. 61–74.

[PMRMB15] Gerard Pons-Moll, Javier Romero, Naureen Mahmood, and Michael J. Black, *Dyna: A model of dynamic human shape in motion*, ACM Transactions on Graphics, (Proc. SIGGRAPH) **34** (2015), no. 4, 120:1–120:14.

[PP12] Ajay S Patil and BV Pawar, *Automated classification of web sites using Naive Bayesian algorithm*, Proceedings of the International MultiConference of Engineers and Computer Scientists, vol. 1, 2012, pp. 519–523.

[PPS94] Yoh-Han Pao, Gwang-Hoon Park, and Dejan J Sobajic, *Learning and generalization characteristics of the random vector functional-link net*, Neurocomputing **6** (1994), no. 2, 163–180.

[QD09] Xiaoguang Qi and Brian D. Davison, *Web Page Classification: Features and Algorithms*, ACM Comput. Surv. **41** (2009), no. 2, 12:1–12:31.

[RBG$^+$09] Martin Reuter, Silvia Biasotti, Daniela Giorgi, Giuseppe Patanè, and Michela Spagnuolo, *Discrete laplace–beltrami operators for shape analysis and segmentation*, Computers & Graphics **33** (2009), no. 3, 381–390.

[RLFV$^+$14] Irene Rodriguez-Lujan, Jordi Fonollosa, Alexander Vergara, Margie Homer, and Ramon Huerta, *On the calibration of sensor arrays for pattern recognition using the minimal number of experiments*, Chemometrics and Intelligent Laboratory Systems **130** (2014), 123 – 134.

[RM71] C. Radhakrishna Rao and Sujit Kumar Mitra, *Generalized inverse of matrices and its applications*, John Wiley & Sons Inc, 1971.

[RM72] C. Radhakrishna Rao and Sujit Kumar Mitra, *Generalized inverse of a matrix and its applications*, Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Theory of Statistics (Berkeley, Calif.), University of California Press, 1972, pp. 601–620.

[Ros58] Frank Rosenblatt, *The perceptron: a probabilistic model for information storage and organization in the brain.*, Psychological review **65** (1958), no. 6, 386–408.

[Ros97] Steven Rosenberg, *The laplacian on a riemannian manifold: an introduction to analysis on manifolds*, no. 31, Cambridge University Press, 1997.

[RSTK03] Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger, *Tackling the poor assumptions of naive bayes text classifiers*, In Proceedings of the Twentieth International Conference on Machine Learning, 2003, pp. 616–623.

[RTWH11] Konrad Rieck, Philipp Trinius, Carsten Willems, and Thorsten Holz, *Automatic analysis of malware behavior using machine learning*, J. Comput. Secur. **19** (2011), no. 4, 639–668.

[Rus07] Raif M Rustamov, *Laplace-beltrami eigenfunctions for deformation invariant shape representation*, Proceedings of the fifth Eurographics symposium on Geometry processing, Eurographics Association, 2007, pp. 225–233.

[RWD88] T. Robertson, F.T. Wright, and R. Dykstra, *Order restricted statistical inference*, Probability and Statistics Series, John Wiley & Sons, 1988.

[RWP05] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke, *Laplace-spectra as fingerprints for shape matching*, Proceedings of the 2005 ACM symposium on Solid and physical modeling, ACM, 2005, pp. 101–106.

[RWP06] ______, *Laplace-beltrami spectra as 'shape-dna' of surfaces and solids*, Comput. Aided Des. **38** (2006), no. 4, 342–366.

[Sam69] John W. Sammon, *A Nonlinear Mapping for Data Structure Analysis*, IEEE Transactions on Computers **C-18** (1969), no. 5, 401–409.

[Sch78] Gideon Schwarz, *Estimating the dimension of a model*, The annals of statistics **6** (1978), no. 2, 461–464.

[SGBL17] Zhiyu Sun, Andrey Gritsenko, Stephen Baek, and Amaury Lendasse, *Deep Spectral Descriptors: Learning the Point-Wise Correspondence Metric via Siamese Deep Neural Networks*, The 25th Pacific Conference on Computer Graphics and Applications (Pacific Graphics 2017), upcoming 2017.

[SOG09] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas, *A concise and provably informative multi-scale signature based on heat diffusion*, Computer graphics forum **28** (2009), no. 5, 1383–1392.

[SP04] Robert W Sumner and Jovan Popović, *Deformation transfer for triangle meshes*, ACM Transactions on Graphics (TOG) **23** (2004), no. 3, 399–405.

[SRG15] Daniel Schupp, Edward Ratner, and Andrey Gritsenko, *U.S. Provisional Patent Application No. 7062320: Object categorization using statistically-modeled classifier outputs*, August 2015.

[STL09] Miki Sirola, Jaakko Talonen, and Golan Lampi, *SOM based methods in early fault detection of nuclear industry.*, ESANN, 2009.

[TdL00] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford, *A Global Geometric Framework for Nonlinear Dimensionality Reduction*, Science **290** (2000), no. 5500, 2319–2323.

[Ten98] Joshua B Tenenbaum, *Mapping a manifold of perceptual observations*, Advances in Neural Information Processing Systems 10 (M. I. Jordan, M. J. Kearns, and S. A. Solla, eds.), MIT Press, 1998, pp. 682–688.

[Tik95] A.N. Tikhonov, *Numerical Methods for the Solution of Ill-Posed Problems*, Current Plant Science and Biotechnology in Agriculture, Springer, 1995.

[T14] Pnar Tfekci, *Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods*, International Journal of Electrical Power & Energy Systems **60** (2014), 126 – 140.

[vML$^+$09] Mark van Heeswijk, Yoan Miche, Tiina Lindh-Knuutila, Peter A. Hilbers, Timo Honkela, Erkki Oja, and Amaury Lendasse, *Adaptive Ensemble Models of Extreme Learning Machines for Time Series Prediction*, Proceedings of the 19th International Conference on Artificial Neural Networks: Part II (Berlin, Heidelberg) (Cesare Alippi, Marios Polycarpou, Christos Panayiotou, and Georgios Ellinas, eds.), ICANN '09, Springer-Verlag, 2009, pp. 305–314.

[vMOL11] Mark van Heeswijk, Yoan Miche, Erkki Oja, and Amaury Lendasse, *GPU-Accelerated and Parallelized ELM Ensembles for Large-scale Regression*, Neurocomputing **74** (2011), no. 16, 2430–2437.

[VPN$^+$10] Jarkko Venna, Jaakko Peltonen, Kristian Nybo, Helena Aidos, and Samuel Kaski, *Information retrieval perspective to nonlinear dimensionality reduction for data visualization*, The Journal of Machine Learning Research **11** (2010), 451–490.

[VVA$^+$12] Alexander Vergara, Shankar Vembu, Tuba Ayhan, Margaret A. Ryan, Margie L. Homer, and Ramn Huerta, *Chemical gas sensor drift compensation using classifier ensembles*, Sensors and Actuators B: Chemical **166167** (2012), 320 – 329.

[WCW74] James O Westgard, R Neill Carey, and Svante Wold, *Criteria for judging precision and accuracy in method development and evaluation*, Clinical Chemistry **20** (1974), no. 7, 825–833.

[WEG87] Svante Wold, Kim Esbensen, and Paul Geladi, *Principal component analysis*, Chemometrics and intelligent laboratory systems **2** (1987), no. 1-3, 37–52.

[WEH14] Ning Wang, Meng Joo Er, and Min Han, *Parsimonious extreme learning machine using recursive orthogonal least squares*, Neural Networks and Learning Systems, IEEE Transactions on **25** (2014), no. 10, 1828–1841.

[WEH15] ______, *Generalized single-hidden layer feedforward networks for regression problems*, Neural Networks and Learning Systems, IEEE Transactions on **26** (2015), no. 6, 1161–1176.

[WHDE14] Ning Wang, Min Han, Nuo Dong, and Meng Joo Er, *Constructive multi-output extreme learning machine with application to large tanker motion dynamics identification*, Neurocomputing **128** (2014), 59 – 72.

[Whi89] Stephen H. White, *An additional hidden unit test for neglected non-linearity in multilayer feedforward networks*, Neural Networks, 1989. IJCNN., International Joint Conference on, 1989, pp. 451–455 vol.2.

[Whi06] Halbert White, *Chapter 9 Approximate Nonlinear Forecasting Methods*, Handbook of Economic Forecasting (C.W.J. Granger G. Elliott and A. Timmermann, eds.), vol. Volume 1, Elsevier, 2006, pp. 459–512.

[Wol66] H. Wold, *Estimation of principal components and related models by iterative least squares.*, In Multivariate Analysis (P.R. Krishnaiah, ed.), vol. 59, Academic Press, NY, 1966, pp. 391–420.

[WSEL15] N. Wang, J.-C. Sun, M.J. Er, and Y.-C. Liu, *A novel extreme learning control framework of unmanned surface vehicles*, Cybernetics, IEEE Transactions on **PP** (2015), no. 99, 1–1.

[Yeh98] I.-C. Yeh, *Modeling of strength of high-performance concrete using artificial neural networks*, Cement and Concrete Research **28** (1998), no. 12, 1797 – 1808.

[YME$^+$13] Qi Yu, Yoan Miche, Emil Eirola, Mark van Heeswijk, Eric Séverin, and Amaury Lendasse, *Regularized extreme learning machine for regression with missing data*, Advances in Extreme Learning Machines (ELM 2011) **102** (2013), 45–51.

[ZE01] Bianca Zadrozny and Charles Elkan, *Learning and making decisions when costs and probabilities are both unknown*, Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (New York, NY, USA), KDD '01, ACM, 2001, pp. 204–213.

[ZE02] ______, *Transforming classifier scores into accurate multiclass probability estimates*, Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (New York, NY, USA), KDD '02, ACM, 2002, pp. 694–699.

[Zha04] Harry Zhang, *The Optimality of Naive Bayes*, Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, Miami Beach, Florida, USA, 2004, pp. 562–567.

[ZHC13] Weiwei Zong, Guang-Bin Huang, and Yiqiang Chen, *Weighted extreme learning machine for imbalance learning*, Neurocomputing **101** (2013), 229–242.

[ZOT14] Yiteng Zhai, Yew-Soon Ong, and I. W. Tsang, *The Emerging "Big Dimensionality"*, IEEE Computational Intelligence Magazine **9** (2014), no. 3, 14–26.