Ensemble classification and its application to visual tracking

Zhang, Le

2016

Zhang, L. (2016). Ensemble classification and its application to visual tracking. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/69073>

<https://doi.org/10.32657/10356/69073>

Ensemble Classification and Its Application to Visual Tracking

Le Zhang

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of

Doctor of Philosophy

Februray 2016

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text.

Le Zhang
Februray 2016

Acknowledgements

I would like to express my sincere gratitude to all the people who are for me, with me and by me. First of all, I would like to take this opportunity to express my deep gratitude to my project supervisor Associate Professor P. N. Suganthan, who has provided me enough research freedom and excellent guidance during this whole PhD study. Not only he gave me the right direction of research, but also offered opportunities to improve my comprehension competitiveness, ranging from critical thinking, conference arrangement, presentation skills, etc.

Secondly, I would like to thank all my friends and colleagues for the great discussions. Those friends include but are not limited to : Mr Jinming Xu from Nanyang Technological University, Zhu Hao from 3M China Ltd, Tairui Chen from Worcester Polytechnic Institute, Zenglin Shi from Zheng Zhou University.

Finally, and most importantly, my greatest thanks and love belong to my parents, parents in law and my lovely wife, Qian Li, for their unwavering love, care and support. They allowed me to spend most of the time on this thesis. I would like to explore the coming unknown journey together with all of you.

Summary

In machine learning and statistics, ensemble methods employ multiple models to obtain better performance than that could be obtained from any of the constituent (base) models [1]. Many studies have been published, both theoretical and empirical, which demonstrate the advantages of ensemble methods for classification.

Three fundamental reasons for ensemble methods in machine learning are summarized in [2]: statistical, computational and representational. In addition, bias-variance decomposition [3], margin theory [4] and strength-correlation [5] also explain why ensemble methods work. Moreover, most ensemble methods naturally fit distributed or parallel computing environment. Thus, researchers from various disciplines such as artificial intelligence and statistics considered the use of ensemble methodology.

The work in this thesis is largely motivated by the aforementioned advantages of ensemble learning. More specifically, we work on decision tree ensemble in Chapter 3. Conventional decision tree induction methods work by exhaustively searching for a discriminative feature to divide the data at a specified internal node based on some criteria such as information gain, Gini, and so on. We directly leverage the data structure and propose to cluster the data by two hyperplanes. The proposed methods are shown to be more efficient and most importantly, lead to better generalization performance. In Chapter 4, we choose neural network as the base model. We firstly evaluate the classical random neural network model in a comprehensive and extensive manner. After identifying the key component of the random neural network model, we evaluate the proposed method in the ensemble framework. In Chapter 5, we work on kernel ridge regression, which admits a simple closed-form solution and can achieve competitive performance to more sophisticated methods such as support vector machines. We proposed a co-trained kernel ridge regression method which can train 2 kernel ridge regression models jointly.

The second part of this thesis is based on visual tracking, which can be posed as an online learning problem. It is a fundamental task in computer vision with a fast-growing interest

due to applications such as video surveillance, human computer interaction, medical imaging and so on. Despite significant progress made recently, it remains challenging due to factors such as lighting and background variations, occlusions, motion blur and so on. Although visual tracking can be formulated in different settings according to different applications, most researchers focus on the *one – pass model – free single object tracking* setting, which is the most fundamental case. More specifically, without any appearance model, it only provides the bounding box of one single object in the first frame. Given this single (labelled) instance, the goal is to predict the location of the object in an online manner. In this part, we apply the concepts proposed in previous section. We show the methods proposed in this thesis perform favorably against current state-of-the-art methods using 51 frequently used benchmark video sequences.

Table of Contents

List of Figures	ix
List of Tables	x
List of Abbreviations	xiii
1 Introduction	1
1.1 Research Background	1
1.2 Motivations	3
1.3 Contributions	5
1.4 Organization	6
I Ensemble Classification	7
2 Literature Review of Related Methods for Classification	8
2.1 Overview of Ensemble learning	8
2.2 State-of-the-Art Decision Tree Ensemble: Random Forest	13
2.3 Random Vector Functional Link Network	17
2.4 Kernel Ridge Regression	19
3 Oblique Decision Tree Ensemble Via Multisurface Proximal Support Vector Machine	21
3.1 Introduction	21
3.2 Oblique Decision Tree Ensemble	23
3.2.1 Oblique decision hyperplane	23
3.2.2 Small Sample Size Problem	26

3.2.3	Details of Oblique Decision Tree Ensemble	27
3.3	Computational Complexity Analysis	27
3.4	Experiment	28
3.4.1	A Toy Example	28
3.4.2	Preliminary Results about Oblique Decision Tree Ensemble	29
3.4.3	Comprehensive Evaluation with Other 179 Methods on 121 Datasets	37
3.5	Conclusion	40
4	Diverse Random Vector Functional Link Networks Ensemble	41
4.1	Evaluation Protocol	42
4.2	Comprehensive Evaluation of RVFL Variants	43
4.2.1	Importance of Bias and Direct Link	43
4.2.2	Effect of Activation Function	44
4.2.3	Regularized Least Square Versus Moore-Penrose Pseudoinverse	46
4.2.4	Overall Comparison	47
4.2.5	Range of the Random Parameters	48
4.3	RVFL Ensemble with Diverse Parameters	51
4.3.1	Comprehensive Evaluation with Other 179 Methods on 121 Datasets	51
4.4	Conclusion	51
5	Co-trained Kernel Ridge Regression Ensemble	54
5.1	Co-trained KRR	55
5.2	Comprehensive Evaluation with Other 179 Methods on 121 Datasets	56
5.3	Conclusion	59
6	Summary of Part I	60
6.1	Overall Comparison of the Methods in Part I	60
II	Ensemble Tracking	63
7	Literature review for Visual Tracking	64
7.1	Visual Tracking	64
7.2	Performance Evaluation of Visual Tracking	67
8	Robust Visual Tracking via Incremental Oblique Random Forest	68
8.1	Ensemble Tracking	68
8.2	Incremental Oblique Random Forest for Visual Tracking	69

Table of Contents

8.3	Experiments	72
8.3.1	Overall Performance	72
8.3.2	Attribute-based Evaluation	73
8.4	Conclusion	75
9	Robust Visual Tracking via Convolutional Random Vector Functional Link Network	77
9.1	Tracking by CNNs	78
9.2	Convolutional Neural Network	79
9.3	Convolutional Random Vector Functional Link Neural Network	81
9.4	Experiments	83
9.4.1	Implementation Details	83
9.4.2	Results and Discussion	83
9.4.3	Overall Performance	84
9.4.4	Attribute-based Evaluation	86
9.4.5	Comparison with Existing Deep Learning Methods	86
9.5	Conclusion	90
10	Robust Visual Tracking via Co-trained Correlation Filters	92
10.1	Correlation Filters for Tracking	92
10.2	Co-trained Correlation Filters	95
10.2.1	Jointly Training of Ensemble Correlation Filters	95
10.2.2	Model Update	97
10.2.3	An Insight into the Methodology	97
10.3	Experiment	98
10.3.1	Overall Performance	98
10.3.2	Attribute-based Evaluation	100
10.4	Conclusion	100
11	Summary of Part II	103
12	Recommendation for Future Work	105
	List of Publications	107
	Bibliography	109
	Appendix A Details of the 121 UCI datasets	123

List of Figures

2.1	A typical framework of ensemble methods.	9
2.2	A toy example of binary classification boundary generated by orthogonal and oblique decision tree.	12
2.3	The structure of RVFL.	18
3.1	Decision boundary of the spirals dataset.	29
4.1	Performance of RVFL based for different ranges of randomization. Smaller rank indicates better accuracy and less number of hidden neurons. N stands for the number of hidden neurons corresponding to the testing accuracy used in the ranking.	49
7.1	Challenging scenarios in visual tracking.	65
8.1	Overall performance of the proposed RaF tacker compared with other state-of-the art methods.	73
8.2	Precision plot of RaF with under each challenging factor.	74
8.3	Precision plot of RaF with under each challenging factor (continued from Figure 8.2.	75
9.1	A convolutional node in CNN.	80
9.2	Basic structure of the CNN used in this work.	81
9.3	Overall performance of a single CRVFL model.	84
9.4	Overall performance of a ensemble of CRVFL model.	85
9.5	Attribute-based Evaluation.	87
9.6	Attribute-based Evaluation (continued from Figure 9.5).	88
10.1	The precision plots of the results.	101

10.2	The precision plots of the results (continued from Figure 10.1).	102
------	--	-----

List of Tables

2.1	Details of Bagging algorithm	10
2.2	Random Forest	13
2.3	Rotation Forest	16
3.1	Multi2binary Algorithm	25
3.2	NULL space approach	27
3.3	Details of 44 datasets.	30
3.4	Classification Accuracy of Random Forest and its MPSVM based variants.	32
3.5	Classification Accuracy of Rotation Forest and its MPSVM based variants.	33
3.6	Classification Accuracies of the proposed Random Rotation Forest and its MPSVM based variants.	35
3.7	Significance of the difference for Random Forest.	36
3.8	Significance of the difference for Random Rotation Forest	37
3.9	Detailed results of the evaluations.	39
3.10	Summarization of the top 10 classifiers. Methods other than MPRaF-P and MPRRoF-P are from [6].	40
4.1	Activation functions used in this work. s and y are the inputs and outputs, respectively.	42
4.2	Average rank values based on classification accuracies of RVFL variants. .	43
4.3	Result of statistical test on RVFL variants.	44
4.4	Statistical significance test for different activation functions.	45

4.5	Comparisons between ridge regression and Moore-Penrose pseudoinverse. The entry in each column represents the number of times ridge regression (pseudoinverse) is better than pseudoinverse (ridge regression) for the same activation function. Statistically significant columns are highlighted.	47
4.6	Overall Comparison based on overall ranks of 32 RVFL variants ¹ . Lower values in rank reflects better performance.	48
4.7	Average rank values of RVFL based on accuracies for different scale factor values over 121 datasets. Each column stands for a distinct comparison. Lower rank means higher accuracy. All RVFLs use ridge regression solution and <i>radbas</i> activation function.	50
4.8	Average rank values of RVFL based on the number of hidden neurons for different scale factor values over 121 datasets. Each column stands for a distinct comparison. Lower rank means less number of hidden neurons. All RVFLs use ridge regression solution and <i>radbas</i> activation function.	50
4.9	Parameters setting for base RVFL model.	51
4.10	Details Results of RVFL ensemble.	53
5.1	Detailed results of Co-KRR and other KRR variants.	57
5.2	Friedman rank of different KRR methods.	59
6.1	Top 20 out of 183 methods.	61
6.2	Statistical comparisons of the top 20 methods.	62
8.1	Challenging Attributes in the benchmark	73
9.1	Performance (Mean precision (20 px)) comparisons of proposed methods with conventional back propagation based baseline	84
9.2	Number of successful sequences and overall performance	86
9.3	Improvement of the mean precision (over 20 pixels) over MEEM for each attributes	89
9.4	Results of deep learning method	90
10.1	Performance (Mean precision (20 px)) comparisons of <i>CoKCF_HC</i> with <i>KCF</i>	98
10.2	Performance (Mean precision (20 px)) comparisons of <i>CoKCF_CNN</i> with <i>KCF</i> [7] and the <i>KCF_CNN</i> [8]	99
A.1	Details of the 121 UCI datasets	123

List of Tables

B.1	Detailed results of Regularized Least Square based RVFL variants on <i>Sigmoid</i> , <i>Sine</i> and <i>Hardlim</i> Activation Function	125
B.2	Detailed results of Regularized Least Square based RVFL variants on <i>Tribas</i> , <i>Radbas</i> and <i>Sign</i> Activation Function	127
B.3	Detailed results of Moore-Penrose pseudoinverse based RVFL variants on <i>Sigmoid</i> , <i>Sine</i> and <i>Hardlim</i> Activation Function	128
B.4	Detailed results of Moore-Penrose pseudoinverse based RVFL variants on <i>Tribas</i> , <i>Radbas</i> and <i>Sign</i> Activation Function	130

List of Abbreviations

Abbreviations	Full Name
RaF	Random Forest
RoF	Rotation Forest
RRoF	Random Rotation Forest
MPSVM	Multisurface Proximal SVM
MPRaF-T	MPSVM based RaF with Tikhonov Regularization
MPRaF-P	MPSVM based RaF with axis-parallel Regularization
MPRaF-N	MPSVM based RaF with NULL Space Regularization
MPRoF-T	MPSVM based RoF with Tikhonov Regularization
MPRoF-P	MPSVM based RoF with axis-parallel Regularization
MPRoF-N	MPSVM based RoF with NULL Space Regularization
MPRRoF-T	MPSVM based RRoF with Tikhonov Regularization
MPRRoF-P	MPSVM based RRoF with axis-parallel Regularization
MPRRoF-N	MPSVM based RRoF with NULL Space Regularization
RVFL	Random Vector Functional Link Network
KRR	Kernel Ridge Regression
2KRR-avg	Average Result of 2 KRR model
2KRR-max	Maximum Result of 2 KRR model
KRR-O	Single KRR Trained on Original Feature
KRR-T	Single KRR Trained on Transformed Feature
KRR-C	Single KRR Trained on Concatenation of Original and Transformed Feature
Co-KRR	Co-trained KRR
Co-KRR-avg	Average Result of Co-KRR
Co-KRR-max	Maximum Result of Co-KRR
CRVFL	Convolutional Random Vector Functional Link Network
CoKCF_HC	Co-trained Kernelized Correlation Filters with Handcrafted Features
CoKCF_CNN	Co-trained Kernelized Correlation Filters with Features Extracted From ConvNets

Introduction

1.1 Research Background

Classification aims to identify the discrete category of a new observation by learning on a training set of data to get:

$$y_c = f_c(\mathbf{x}, \theta_c), y_c \in \mathcal{Z} \quad (1.1)$$

where $\mathbf{x}$ is the new observation as a feature vector, y_c is the category that the new observation belongs to, $f_c(\cdot)$ is the trained classification function, θ_c is the classification function's parameter set. $\mathcal{Z}$ is the set of class labels/categories.

Ensemble methods leverage the “perturb and combine” strategy. The rationale of ensemble methodology in machine learning is to build a predictive model (either classifier or regressor) by integrating multiple models. Researchers from various disciplines such as artificial intelligence and statistics considered the use of ensemble methodology. Most ensemble methods naturally fit distributed or parallel computing environment. It is well-known that ensemble methods can be used for improving prediction performance [1]. Dietterich [2] explained three fundamental reasons for the success of ensemble methods: statistical, computational and representational. From statistical point of view, learning can be posed as searching a space H of hypotheses to identify the best hypothesis in the space for data fitting. However, when the amount of training data available is limited compared to the size of the hypothesis space, statistical problem may arise. Without sufficient data, one learning algorithm may end up with many different hypotheses in H which all fit the training data well. In this case, the algorithm can “average” their votes and reduce the risk of choosing the “wrong” classifier by generating an ensemble. From computational point of view, many learning algorithms (such as training of neural networks and induction of decision trees) work by performing some form of local search which suffers from local optima. Ensemble

Chapter 1. Introduction

methods that leverage local searches from many various starting points may find a better approximation to the true unknown function than any of the individual classifiers. Another reason is representational. It is possible that the true function f may not be well represented by any of the hypotheses in H . It may be possible to expand the space of representable functions, thus enhance the representation ability, by forming weighted sums of hypotheses drawn from H . In addition, bias-variance decomposition [3] and strength-correlation [5] also explain why ensemble methods work. As a result, for decades researchers applied numerous ensemble methods to enhance the performance of a single model. In this section, we give a brief review of the works related to this thesis. We refer the interested reader to our recent survey [1] for more information.

This strategy, also called as “ensemble classifiers” [2] or “Multiple classifier systems” (MCS) [9], works in two steps. “Perturb” step means applying a given learning algorithm (homogeneous or heterogeneous) to a set of perturbed training datasets. Then in the “combine” step, the output of the learned classifiers are aggregated in a suitable manner. Ensemble methods gain their popularity with the introduction of bagging [10] and boosting [11].

Bagging [10], or bootstrap aggregation, works by training independent classifiers from a set of instances taken with a replacement from the training set. Bagging is suitable for unstable/weak models where small perturbations of the training set may cause significant performance variations. In case case, bagging can significantly reduce the variance of such unstable models and lead to better generalization ability. Different from bagging, boosting [11] must be trained in a sequential manner, where the current classifier being trained will focus on those training sampled which can not be classified correctly by previous ensemble members. The distribution of each sample in the training set will be updated in each training iteration. Those “difficult” samples will be more likely to be sampled than those “easy” ones.

Proposed by Breiman [5], Random Forest combines the concepts of bagging and random subspace. Breiman [5] pointed out Random Forest is similar to AdaBoost even though AdaBoost is sequential whereas random forest is parallel. Lin and Jeon [12] showed that random forest is a kind of adaptive nearest neighbour. Random forest has gained popularity in high-dimensional and ill-posed problems [13, 14]. In a recent work in [6], the authors evaluate the performance of 179 classifiers arising from 17 families (ensemble methods, neural networks, support vector machines, etc) on 121 UCI dataset [15]. According to their result, Random Forest achieves the top performance and is the most likely to be the best classifier.

1.2 Motivations

With simple concept and outstanding performance, Random Forest has received a wide range of research interest and has being applied to solve many real-life problems. In a decision tree, each hyperplane at a non-leaf node separates the data so that the further separations in the children nodes are easier. The hyperplane itself does not have to be a good classifier at that stage [16]. Most decision tree induction algorithms work by finding a split that has the lowest impurity score such as the gini impurity, entropy, or twoing rule. The most commonly used is the “axis-parallel” decision trees which for each feature, the algorithm will exhaustively find a cutting-plane and score it based on one pre-defined criteria. The nature of impurity criterion is to measure the skewness of the distribution of different classes in the set of samples reaching the node. It gives low impurity score to those distributions that are near uniform and gives high score to those distributions where the number of patterns of one class is much larger than the rests. Most of the popular decision tree induction algorithms work by optimizing an impurity measure. However, many of the impurity measures are not differentiable with respect to the hyperplane parameters. Thus, those algorithms employ some search techniques for finding the best hyperplane at each node. For example, CART-LC [17] uses a deterministic hill-climbing algorithm, OC1 [18] uses a randomized search based on CART-LC. Both of these approaches search in one dimension at a time, which becomes computationally cumbersome in high-dimensional feature spaces. Moreover, both algorithms face the local optimum problem. Multiple trials or re-starts can be employed to decrease the possibility of ending up with a locally optimal solution. One alternative method to optimize in all dimension is evolutionary approaches [19, 20]. They are claimed to be tolerant to noisy evaluations of the rating function and also facilitate optimizing multiple rating functions simultaneously [21, 22]. In [23], the authors propose to involve stronger randomization in the tree classifier. [24] generate this idea to oblique case. Other existing works relating to the decision tree integration and its generalization can be found in [25–27]. However, besides higher computational complexity, those search approaches involve several parameters which are difficult for practitioners to tune. In [28], a “Decision Tree Support Vector Machine” (DTSVM) is proposed. This work uses standard support vector machine with RBF kernel to generate the optimal hyperplane. However, besides careful parameter tuning via extra cross-validation, this method is rather slow.

It is worth noticing that the problem with all impurity measures, which is reported by [16], is that they depend only on the distribution of different classes on each side of the hyperplane. Hence, the impurity measures do not really capture the geometric structure of class regions because the impurity measure of the hyperplane will not be changed if one changes the label of the data without changing the relevant features of each class on

Chapter 1. Introduction

either side of a hyperplane. Instead of only considering the label information, the geometric structure considers the internal data structure which measures the distance of the data sample to the decision hyperplane. So if any of the relevant feature changes, the hyperplane will change.

In order to search for such a hyperplane that captures the geometric structure of class distributions, Naresh Manwani [16] borrows ideas from a recent variant of the support vector machines method to generate the geometric decision tree. The multisurface proximal SVM (MPSVM) [29] finds two clustering hyperplanes, one for each class, where each hyperplane is generated such that it is closest to one class of data and as far as possible from the other class, and the data is classified based on the distance to the hyperplanes. In their work, at each node, they use MPSVM to find two clustering hyperplanes. They choose one of the angle bisector of these two hyperplanes with better impurity measure. As the tree grows, the samples reaching the subsequent nodes become fewer and the small sample size problem arises. The authors propose to use NULL space [30] method to tackle this. However, this work need careful tree pruning technique to avoid over-fitting. Time-consuming cross-validation is needed to determine the optimal parameters for the pruning process. This motivates us to investigate the performance of oblique decision tree based decision tree ensemble.

As an unstable learner, neural network is also widely used as the base model in ensemble learning. Conventional neural networks employ back-propagation to tune all their parameters, which is shown to be time consuming and usually suffers from local minimum. Random Vector Functional Link Network (RVFL) remedies this by randomly initializing the parameters of the hidden neurons and optimize the remaining ones. Though proposed a long time ago, some key properties in RVFL remain to be under-researched. Chapter 4 in the thesis fills this research gap and the basic conclusions therein can be used as a guideline for designing of RVFL in practice.

Kernel Ridge Regression (KRR) is originally proposed to solve the regression problem and is shown to achieve similar performance with more sophisticated models such as support vector regression. Kernel method is widely regarded as strong classifier and performs poorly under the ensemble framework [6]. Generally speaking, the complexity of KRR is $O(n^3)$, where n is the size of data sample. Thus, it may be very time consuming for KRR based ensemble methods with a large number of base models (usually several hundreds or even more). This also motivates us to investigate the ensemble KRR with a small number of base models.

1.3 Contributions

In this thesis, we propose to evaluate the performance of decision tree ensembles which employ geometric decision trees as base classifiers. We integrate the oblique decision tree in decision tree ensemble to determine their performances. We propose two different regularization approaches to address the small sample size problem. It was found that the NULL space approach is unstable compared to others [31].

According to the work of [32], ensemble methods can significantly improve the performance of unstable classifiers such as decision tree and neural networks. In this thesis, we bring back the machine learning community a very classical method, random vector functional link network (RVFL), which randomly generates the parameters in hidden layers of single hidden layer feed-forward neural networks. We conduct comprehensive evaluations of RVFL variants and figure out the key components which dominates the performances. We also evaluate the performance of RVFL under ensemble framework.

Kernel Ridge regression (KRR), on the other hand, can be regarded as a strong classifier and usually lead to competitive performance compared with other sophisticated methods such as support vector machine. However, in the literature, there is little work about KRR ensemble. In this thesis, we also propose an ensemble method which can jointly train an ensemble of KRR models. We have demonstrated the effectiveness of the proposed method for classification.

We are interested in making the proposed method applicable for real-life problems. A second significant contribution of this thesis is to validate the aforementioned methods (Random Forest, RVFL, KRR) to an incremental computer vision task: visual tracking. Visual tracking has received a fast-growing interest due to applications such as video surveillance, human computer interaction, medical imaging and so on. Despite significant progress made recently, it remains challenging due to wide factors such as lighting and background variations, occlusions, motion blur and so on. In this thesis, we have comprehensively compared the proposed methods for this task with a wide range of state-of-the-art trackers on 51 video sequences. Experimental results demonstrate the effectiveness of the proposed methods.

To summarize, the contributions of this thesis can be summarized as follows:

- Propose oblique decision tree forests in Chapter 3.
- Conduct a comprehensive evaluation of RVFL model and propose a RVFL ensemble methods with diverse parameters in Chapter 4.
- Propose an efficient KRR ensemble method in Chapter 5.

Chapter 1. Introduction

- Apply those three methods to solve visual tracking problem and achieve very competitive performance.

In order to have reasonable conclusion, all the performance evaluations aforementioned are based on a large amount of dataset and other state-of-the-art methods. In particular, the classification performances of decision tree ensemble, RVFL and KRR based ensemble are compared with other 179 classifiers on 121 datasets. Our proposed decision tree and KRR ensemble achieve the top performance. The proposed RVFL ensemble can achieve very competitive performance and outperforms other neural network methods. The proposed trackers are compared with other 29 state-of-the-art methods on 51 video sequences. In addition, the propose method can also achieve very competitive performance.

1.4 Organization

The remaining thesis is organized as follows. Chapter 2 elaborates the background and related work of the proposed classification methods in this thesis. Chapter 3 depicts the oblique decision tree ensemble approach. Chapter 4 details the classical RVFL methods and the proposed RVFL ensemble. Chapter 5 describes the proposed Co-trained KRR methods. Chapter 2, Chapter 3, Chapter 4 and Chapter 5 constitute the first part of the thesis and we present a summary on Part I in Chapter 6. The second part of this thesis consists of Chapter 7, Chapter 8, Chapter 9 and Chapter 10. In Chapter 7 we give some background about visual tracking. Oblique Random Forest, RVFL and KRR based trackers are presented in Chapter 8, Chapter 9 and Chapter 10, respectively. Chapter 11 summarizes the second part of this thesis. Future work can be found in Chapter 12.

Part I

Ensemble Classification

Literature Review of Related Methods for Classification

2.1 Overview of Ensemble learning

We consider the following ensemble learning framework as demonstrated in Fig 2.1 which consists of the following blocks:

- Training set: A labeled dataset (X, Y) used for training, where $X = [x_1; x_2; \dots; x_n]$, $Y = [y_1; y_2; \dots; y_n]$ denote the stack of feature vectors and class labels, respectively.
- Base model: A learning algorithm which generates a classifier that represents the generalized relationship between the input attributes and the target attribute.
- Diversity Generator: It aims at generating the diverse classifiers. For bagging, it samples with replacement from original training set to generate a unique training set for each classifier. For boosting, it updates the distribution for each training datum in each iteration of training. For Random Forest, it is equivalent to the combination of bagging and random subspace.
- Aggressor: The aggressor is responsible for combining the classifications result of the each base models.

In Fig 2.1, $\mathbf{X}$ is the training set. $\mathbf{X}^{(i)}, i \in \{1, \dots, L\}$ are the bootstrap version of datasets, which is the result of the diversity generator. $f^{(i)}$ are the base predictors and f_{en} is the aggressor. The dashed red lines in the generation and base prediction parts denote boosting related ensemble framework, which indicates training is conducted in a sequential manner.

Figure 2.1 A typical framework of ensemble methods.

Although ensemble methods have been extensively studied, bagging based approaches seem to be perceived as “state-of-the-art” at present [6]. They have been found to be accurate, computationally feasible across various data domains. In this section, we will give a comprehensive literature review about these methods related to this thesis.

Bagging, also known as bootstrap aggregation, is regarded as the most well-known independent ensemble learning methods. It samples with replacement from a given training set to generate diverse training sets for unstable classifiers such as decision tree, neural networks and so on. According to the work in [32, 33], bagging can significantly reduce the variance of unstable classifiers. In [34], the authors showed bagging works because it approximates the optimal procedure of Bayesian model averaging.

The pseudo-code of bagging algorithm is presented in Table 2.1. Note if we sample n samples to generate each X_i , only $(1 - 1/e)(\approx 63.2\%)$ unique training samples are expected to be presented in X_i . While the sampling procedure can make the training sets X_i be different from each other, they are certainly far away from independent from a statistical point of view. In the testing phase, each base classifier gives the classification results for the unknown datum. The bagging classifier returns the class that has been predicted most often (voting method). Bagging can produce a combined model which usually performs better than the single model trained on the original training dataset.

Chapter 2. Literature Review of Related Methods for Classification

Table 2.1 Details of Bagging algorithm

Bagging:

Input: X (the training set), f (the base classifier), M (the ensemble size)

· Training phase:

$n \leftarrow$ the number of training samples in X .

For $i = 1 \dots M$:

- for $i = 1, \dots, M$
- Sample n data with replacement to generate a training set X_i from X .
- Train a classifier f_i from X_i .

End For

· Classification phase:

For a given x , the final decision of the ensemble classifier will be made by majority voting.

There are many variants of bagging in the literature. In [35], double-bagging was proposed where using out-of-bag samples are used for combining different classifiers. A linear discriminant analysis (LDA) model was learned with the out-of-bag samples. This LDA model was further applied to those in-bag samples to grow a classical decision tree. In [36], the authors conducted the first trial on analysis the performance of bagging, boosting and randomization with 33 UCI data. He showed when there is substantial classification noise, bagging is much better than boosting, and sometimes better than randomization. Bagging for gaussian process regression was investigated in [37]. In [38], the author further demonstrated that bagging might improve the performance of the classifier for very unstable classifiers. In [39], a novel ensemble pruning method was proposed based on reordering the classifiers obtained from bagging and then selecting a subset for aggregation. This work was further generalized to [40]. In [41], they authors proposed to prune bagging based ensemble with boosting. Feature bagging for outlier detection was propose in [42] for very large, high dimensional and noisy databases. In [43], the authors showed how the combination of bagging as a variance reduction technique and boosting as a bias reduction technique can result in very high precision and low variance ranking models.

Boosting is a family of sequential ensemble methods and the most prominent member in the boosting family. There are several Adaboost variants. In [44], the authors improved Adaboost in a setting in which hypotheses may assign confidences to each of their predictions. They also applied this new boosting method to multi-class and multi-label problem. Ratsch et

2.1 Overview of Ensemble learning

al. [45] proposed several regularization methods and generalizations of the original Adaboost algorithm to achieve a soft margin Adaboost. In [46], “Modest Adaboost” was proposed which produces less generalization error compared to mentioned algorithms at the cost of somewhat higher training error. SpatialBoost [47] extended AdaBoost to incorporate spatial reasoning and shows the effectiveness on the problem of interactive image segmentation. Arc-x4 [48] was a simple arcing (adaptively resample and combine) algorithm which aims to demonstrate that Adaboost works because of the adaptive re-sampling and not because of the specific form of the weighing function. On the contrary, Domingo [49] proposed Madaboost by modifying the weighting system of AdaBoost which showed to be robust to random noise and could be casted in the statistical query learning model. Friedman et al. [50] presented a revised version of AdaBoost called Real Adaboost, which reduced the computation cost and may lead to better performance. It worked by combining the output class probabilities provided by the base classifiers using additive logistic regression model in a forward stage-wise manner. Ivoting [51] is an improvement on boosting which can be implemented in parallel. In addition, Ivoting is less vulnerable to noise and over-fitting.

Stacking, or stacked generalization [52] is a generalized method to bagging, random subspace or boosting because the aggregation stage is different yet more generalized. It uses another machine learning algorithm to estimate the weights of the outputs from each base predictor. The aggregation algorithm can also be a supervised learning algorithm. For example, SVM was used in the aggregation stage in [53–55]. Another advantage is that since the aggregation stage is supervised, the choice of base predictors are wider and it usually results in forming heterogeneous ensemble methods such as the methods reported in [56, 57].

In [6], the authors evaluate 179 classifiers arising from 17 families (discriminant analysis, neural networks, Support vector machines, decision trees, rule-based classifiers, Bayesian, generalized linear models, nearest neighbours, partial least squares and ensemble based classifiers such as boosting, bagging, stacking, random forest and others.) on 121 UCI [15] dataset. According to their result, the classifier most likely to be the best are Random Forest [5]. Random Forest has been applied to a various field such as machine learning, computer vision, medical imaging and so on. Before we focus on Random Forest, a brief introduction about the base model, decision tree, is presented in this section.

Decision trees are widely used in many diverse areas such as medical diagnosis, image classification, remote sensing, expert systems, decision making and speech recognition, to name only a few. The most important feature of decision tree is its simplicity and easy interpretability. It breaks down a complex decision-making process into a collection of simpler decisions.

Chapter 2. Literature Review of Related Methods for Classification

Figure 2.2 A toy example of binary classification boundary generated by orthogonal and oblique decision tree.

Decision tree works by recursively partitioning the data into different groups. In a decision tree classifier, each internal node is associated with a so-called decision function or splitting rule. The splitting rule is a (usually binary) function of the feature vector. Each leaf node in the tree is associated with a class label. For an unlabelled instance, at every internal node (or root node from the beginning), it is branched to one of the children of that node based on the splitting rule. This process follows a path in the tree. When it reach a leaf node, the class label of this datum is given by the label of that leaf node.

Decision trees can be broadly classified into two types: orthogonal and oblique [58]. Orthogonal decision tree is also known as axis-parallel decision tree or uni-variate decision tree. In an orthogonal decision tree, the split rule works by exhaustively searching for a feature with best discriminative power based on impurity measurement such as information gain, gini impurity and so on. Orthogonal decision tree may generate “stage-like” decision boundary as demonstrated in Fig 2.2. However, in more general situations, we have to approximate arbitrary and complex boundary with many “stage-like” pieces. In this case, the size of the resulting tree becomes large. The oblique decision trees, also known as multi-variate decision tree, on the other hand, use a decision function that depends on a (usually linear) combination of all feature components. An oblique decision hyperplane is usually associated with each internal node of the oblique decision tree. Generally speaking, for an arbitrary decision boundary, the resulting oblique decision tree may have less number of internal nodes than the orthogonal decision tree. Decision tree are very classical methods. For a comprehensive review about decision tree induction, please read the work in [59–61].

2.2 State-of-the-Art Decision Tree Ensemble: Random Forest

A single decision tree generalizes poorly on real-life applications because it overfits easily. Roughly speaking, there are two main approaches to boost the performance of decision trees. The first one is tree pruning. It allows a decision tree to grow to the largest size, in this case the data can not be further partitioned based on the aforementioned impurity criteria. Then they employed pruning to reduce the size of a decision tree. It works by removing the sub-tree which provide little classification ability. Pruning reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting. Commonly used pruning methods include Error-Complexity Pruning [58], Critical Value Pruning [62], Reduced-Error Pruning [63] and so on. For more information, please refer to [60].

Decision tree pruning may improve the performance of decision trees and thus became an active research field, but then fell out of fashion with the rise of decision tree ensemble. In [64], a pioneering work to construct a decision tree based classifier was proposed which maintains excellent accuracy on training data and improves on generalization accuracy as it grows in complexity. The proposed classifier is a decision tree ensemble where each base model was constructed systematically by a random subsets of the feature vector. In [5], Breiman combined the concept of random subspace and bagging to generate the well-known Random Forest (RaF). The pseudo-code of Random Forest is presented in Table 2.2.

Table 2.2 Random Forest

Random Forest Algorithm:

· Training phase:

Given:

$X(n \times m)$ is the training dataset, where n is the number of the training data, m is the dimension of each data.

$Y(n \times 1)$ is the labels of the training set.

M is the ensemble size, which means the number of trees in the forest.

T_i refers to each random tree in the Random Forest, $i = 1 \dots M$.

$mtry$ is the number of features randomly selected to split in each non-leaf node.

$minleaf$ is the number of maximum number of samples in an impure node.

1. Generate the training set for X_i by sampling n times from all n available training cases with replacement.
2. At each node the best split is calculated using the $mtry$ randomly chosen features in the training set for T_i .
3. Go to Step 2 until T_i is fully grown until reach a pure node or the number of samples are smaller than $minleaf$.

· Classification phase:

For a given sample, it is pushed down each tree in the forests and each tree in the forests will give one vote on the predicted label of this sample. In this case, the predicted label of this sample is determined as the one which has the most votes in the forests.

Chapter 2. Literature Review of Related Methods for Classification

Since each base classifier is trained on a bootstrap data set whose distribution is similar to the whole population, all base classifiers perform well. The random subspace strategy at each node further increases the variance of each base classifier. The rationale behind the random subspace strategy is that the explicit randomization of the attribute selection combined with ensemble averaging should be able to reduce variance more strongly than the weaker randomization schemes. Since the work in [5], Random Forest has demonstrated its effectiveness in solving problems such as micro-arrays [13], time series [14], spectral data [65], object recognition and image segmentation [64, 66]. Random Forest is comparable in performance to many other non-linear learning algorithms. It performs well with little parameter tuning [67], and is able to identify relevant feature subsets even in the presence of a large number of irrelevant features [12, 13, 68]. More recently, additional applications of Random Forest such as feature selection [65] and the explorative analysis of sample proximities [69], have gained interest.

Breiman pointed out Random Forest is similar to Adaboost. Lin and Jeon [12] showed Random Forest is kind of adaptive nearest neighbourhood. Random Forest has gained popularity in high-dimensional and ill-posed problems [13, 14]. There are variations of Random Forest reported. Stratified Random Forest [70] modified Random Forest by replacing the uniform sampling of the feature in each node by weighting sampling. The weight is calculated by some other methods such as linear discriminant analysis or Chi-square. This modification is shown to work better on high-dimensional problems. Zhang et al. [71] proposed an instance based method to select several trees from the Random Forest. Geurts et al. [23] proposed Extremely Randomized Trees. In their method, in each node of the tree, the threshold for each randomly selected feature was randomly generated. With smaller computational complexity, Extremely Randomized Trees can outperform conventional Random Forest on many problems. In another work, Zhang et al. [24] generalized this idea to oblique Random Forest which used a combined feature to split in each node. In their work, both the combination coefficient for each feature and the threshold (or cutting-point) were randomly generated. Menze et al. [72] proposed another kind of oblique Random Forest which used a combined feature to get the “best-split” function with ridge regression in each node. Similarly, Zhang and Suganthan [73] proposed another hybrid oblique Random Forest. In their work, they propose to use different transformation methods in each node. The root node works to decide which transformation to use for the subsequent nodes. In another work [74], Zhang and Suganthan proposed to use Multisurface Proximal Support Vector Machine (MPSVM) [29] to obtain the “best-split” hyperplane in each node of the Random Forest. Experiments demonstrated the advantage of the MPSVM based hyperplane over the conventional axis-parallel hyperplane.

2.2 State-of-the-Art Decision Tree Ensemble: Random Forest

Though there are many Random Forest variants existing in the literature, here we want to emphasize the popular Rotation Forest. Rotation Forest [75] (RoF) draws upon the Random Forest idea but differs from it mainly in two aspects. Firstly, each tree in RoF is trained in a rotated feature space of the whole data, which is obtained by a rotation matrix as demonstrated in Eqn (2.2). See Table 2.3 for more details. Secondly, at each internal node, the best split is explicitly searched among all the features. The details of RoF can be found in Table 2.3. The authors claim that a small rotation of the axes may lead to a very different tree since the tree induction algorithm builds the classification regions using hyperplanes orthogonal to the feature axes. In this case, the variance of the tree classifiers can be significantly improved. In their study, the authors found RoF works better than RaF and even more better for smaller ensembles sizes. Recently, Rotation Forest has been applied in various research areas such as Economics and Business, medical [76, 77], and bioinformatics [78, 79].

Chapter 2. Literature Review of Related Methods for Classification

Table 2.3 Rotation Forest

Rotation Forest algorithm:

· Training phase:

Given:

$X(n \times m)$ is the training dataset, where n is the number of the training data, m is the dimension of each data.

F is the feature set (feature indexes).

$Y(n \times 1)$ is the labels of the training set.

M is the ensemble size, which means the number of trees in the forests.

T_i refers to each random tree in the Rotation Forest, $i = 1 \dots M$.

m is the number of subsets.

$\{w_1, \dots, w_c\}$ the set of class labels.

For $i = 1 \dots M$:

1. prepare the Rotation Matrix R_i in Eqn (2.2)

- Split F into m subsets: $F_{i,j}$ (for $j = 1 \dots m$)
- For $j = 1 \dots m$
 - Let $X_{i,j}$ be the data set X for the features in $F_{i,j}$.
 - Eliminate from $X_{i,j}$ a random subset of classes.
 - Select a bootstrap sample from $X_{i,j}$ of size 75% of the number of objects in $X_{i,j}$. Denote the new set by $X'_{i,j}$.
 - Apply PCA on $X'_{i,j}$ to obtain the coefficients in a matrix $C_{i,j}$.
- Arrange the $C_{i,j}$ for $j = 1 \dots m$ in a rotation matrix as in equation (2.2).
- Construct R_i^a by rearranging the columns of R_i so as to match the order of features in F .

2. Building classifier using (XR_i^a, Y) as the training set.

· Classification phase:

1. For a given x , let $d_{i,j}(xR_i^a)$ be the probability assigned by classifier D_i to the hypothesis that x comes from class w_j . Calculate the confidence for each class, w_j , by the average combination method:

$$\mu_j(x) = \frac{1}{L} \sum_{i=1}^L d_{i,j}(xR_i^a), \quad j = 1, \dots, c \quad (2.1)$$

2. Assign x to the class with the largest confidence.
-

$$R_i = \begin{bmatrix} \alpha_{i,1}^{(1)}, \alpha_{i,1}^{(2)}, \dots, \alpha_{i,1}^{(M_1)}, & [0] & \dots & [0] \\ [0] & \alpha_{i,2}^{(1)}, \alpha_{i,2}^{(2)}, \dots, \alpha_{i,2}^{(M_2)}, & \dots & [0] \\ \vdots & \vdots & \ddots & \vdots \\ [0] & [0] & \dots & \alpha_{i,m}^{(1)}, \alpha_{i,m}^{(2)}, \dots, \alpha_{i,m}^{(M_m)} \end{bmatrix}, \quad (2.2)$$

Zhang and Zhang proposed RotBoost [80] which combines the Adaboost and Rotation Forest. In their method, each tree classifier in Rotation Forest is replaced with an Adaboost

based decision tree ensemble. Their results showed that the RotBoost can generate significant better classification error than both Rotation Forest and Adaboost. In [81], the authors investigated the effects of various transformation methods on Rotation Forest and RotBoost. Principal Component Analysis (PCA), Independent Component Analysis (ICA) and Sparse Random Projections (SRP) were used in their work. In their work, RotBoost outperforms Rotation Forest in terms of accuracy, but none of the considered variations offers a clear advantage. However, in terms of AUC and top-decile lift, Rotation Forest is the best. Moreover, ICA-based Rotation Forests outperforms all other considered benchmark algorithms such as PCA and SRP based Rotation Forest. In [82], the authors proposed to combine several benchmark ensemble methods (bagging Random Forest, Rotation Forest and Adaboost) with a random linear oracle, this aggregation strategy achieved better performance.

2.3 Random Vector Functional Link Network

Except decision tree, neural network is another kind of unstable classifier. Inspired by biological Neural network, neural networks are commonly used learning methods for approximating or estimating functions that may depend on a large number of inputs and outputs. Typically, training algorithms of neural network are based on minimizing a loss function defined on the desired output of the data and actual output of the ANN through updating the parameters. Classical approaches usually tune the parameters based on the derivatives of the loss function. However, much of the power of neural network comes from the nonlinear function in the hidden units used to model the nonlinear mapping between the input and output. Unfortunately, this kind of training protocol loses the elegance of finding the global minimum solution with respect to all the parameters of the network since the loss function depends on the output of nonlinear neurons. Thus, the optimization turns out to be nonlinear least square problem which is usually solved iteratively. In this case, the error signal has to be propagated backwards to serve as a guidance for tuning the parameters [83]. Due to this, it is widely acknowledged that these training methods are very slow [84] and may not converge to a single global minimum because there exist many local minima [85, 86] and also the resulting neural network is very weak in the real world noisy situations. These weaknesses of this family of methods naturally limit the applicability of gradient-based algorithms for training neural networks. Randomization based methods remedy this problem by either randomly fixing the network configurations (such as the connections) or some parts of the network parameters (while optimizing the rest by a closed form solution or an iterative procedure), or randomly corrupt the input data or the parameters during the training. Remarkable results have been achieved in various network structures, such as single

Chapter 2. Literature Review of Related Methods for Classification

hidden layer feed forward network [87], RBF neural networks [88], deep neural network with multiple hidden layers [89], convolutional neural network [90] and so on. We refer the interested reader to the recent survey [91] for this line of research.

One pioneering work on training neural network with randomization can be found in [92]. The so-called Random Vector Functional Link Neural Network (RVFL) is demonstrated in Fig. 2.3. The input features are firstly transformed into the enhanced features by the enhancement nodes where parameters within them are randomly generated. All the original and enhanced features are connected to the output neurons

Figure 2.3 The structure of RVFL.

The rationale behind this architecture is to improve the generalization ability of the network with some enhanced features achieved by nonlinear transformation on the original features. The weights H_{ij} from the input to the enhancement nodes are randomly generated such that the activation function $g(a_j^t x + b_j)$ is not saturated most of the time. For RVFL, only the output weights W_j need to be learned. Suppose the input data has m features and there are J enhancement neurons, there are in total $m + J$ inputs for each output node. Learning is achieved by minimizing the following objection function:

$$E = \frac{1}{2n} \sum_{i=1}^N (y^i - d^i * W)^2 \quad (2.3)$$

where W^t consists of the weight values $W_j, j = 1, 2, \dots, k + J$. There are n input data in total. E is quadratic with respect to each $m + J$ dimensional vector W_j , indicating that the unique minimum can be found in no more than $m + J$ iterations of the learning procedure such as the conjugate gradient method. For simplicity, let $o_p = \sum W_j d_{pj}$ be the output of the p^{th} data, then the changes in the weight are set to be $\Delta W_{pj} = \eta (y_p - o_p) d_{pj}$. In the $(k + 1)^{th}$ iteration, the weights are updated as $W_j(k + 1) = W_j(k) + \sum_p \Delta W_{pj}$. The learning procedure iterates until a stopping criterion is met. An alternative solution within a single learning step can be

achieved by the Moore-Penrose pseudo-inverse [93, 94]. In this case, the weight W can be achieved by $W = d^+Y$, where $+$ represents the Moore-Penrose pseudo inverse and Y is the stacked version of the labels. One can also use the commonly used regularized least square ¹ to get the closed form solution for W as:

$$W = (d^T d + \lambda I)^{-1} Y; \quad (2.4)$$

where λ is a user-defined regularization parameter and I is a identity matrix.

Some theoretical justifications were presented in [93] for RVFL as well as other neural networks with hidden nodes implemented as product of univariate functions or radial basis functions. They formulate the problem as a limit-integral representation of the function to be approximated. In this case, the limit-integral representation is further approximated by the Monte-Carlo method. It has been proven that with weights and biases from input layer to hidden layer sampled from uniform distribution with a proper range, RVFL is efficient universal approximator for continuous functions on bounded finite dimensional sets. Indeed the overall error of the approximation can be bounded by sum of the error of approximating the function by the integral and the error of the approximating the integral by the Monte-Carlo methods.

2.4 Kernel Ridge Regression

Ridge regression is a popular machine learning algorithm based on least square. Given a training set (X, Y) , where $X = [x_1; x_2; \dots x_n]$ is an $n \times m$ data matrix. n denotes for the number of data samples and m is the dimensionality of the feature vector. $Y = [y_1; y_2; \dots y_n]$ is a $n \times 1$ output vector. The learning objective is to find a $m \times 1$ vector w such that the mean square error between XW and Y can be minimized [95]. Thus, the learning objective can be formulated as:

$$\min_w \sum_i (x_i w - y_i)^2 + \lambda \|w\|^2; \quad (2.5)$$

The λ is a user-defined regularization parameter that controls the overfitting. The minimizer has an elegant closed-form solution, which is given by:

$$w = (X^T X + \lambda I)^{-1} X^T Y; \quad (2.6)$$

As we mentioned above, the data matrix X has one sample per row x_i , and each element of the vector Y is the output target y_i for x_i . I is an identity matrix.

¹In this work, "regularized least square" and "ridge regression" are used interchangeably.

Chapter 2. Literature Review of Related Methods for Classification

In [96], the authors proposed a dual version of the ridge regression procedure. They also applied the kernel trick in their work to form a nonlinear predictor. According to the Representer Theorem, the solution of w can be formulated as a linear combination of the samples in the feature space $\phi(x)$ as: $w = \sum_i \alpha_i \phi(x_i)$. The problem is then formulated as:

$$\min_{\alpha} \|Y - K\alpha\|^2 + \lambda \alpha^T K \alpha; \quad (2.7)$$

Similarly, the solution is given in a closed-form manner as:

$$\alpha = (K + \lambda I)^{-1} Y; \quad (2.8)$$

By applying the kernel trick, the kernel matrix K can be obtained by $K_{ij} = k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$. One can simply employ the commonly used kernel functions such as Gaussian kernel, RBF kernel, linear kernel without explicitly knowing the form the feature transformation function $\phi(x)$.

Oblique Decision Tree Ensemble Via Multisurface Proximal Support Vector Machine

As an unstable classifier, decision tree has very low bias and extremely high variance [32]. Ensemble methods can significantly boost the performance of decision tree by largely reducing its variance. In [64], the authors proposed to use random subspace to generate decision tree ensemble. Breiman combined this concept with bagging to form the well-known Random Forest. According to a recent benchmark which compare 179 classifiers on 121 datasets, Random Forest is the most accurate classifier so far.

3.1 Introduction

Although decision tree ensemble have been widely studied, most of them are based on orthogonal decision tree [97]. Actually the approaches for learning decision trees can be classified into two broad categories: orthogonal (or axis-parallel or univariate) decision tree explicitly searches the “best” feature to split among several features according to some impurity criteria. Oblique (or multivariate) decision tree performs a test at each node based on all or part of the features [98]. Generally speaking, univariate decision tree can approximate any oblique decision boundary by using a large number of “stage-like” decision boundaries.

In a decision tree, each hyperplane at an internal node separates the data in such a way that the further separations in the children nodes are easier. The hyperplane associated with each internal node may not be a good classifier at that stage [16]. Commonly used decision tree induction methods work by searching for a split that has the lowest impurity score

such as the entropy, twoing rule, gini impurity. The most commonly used method is the orthogonal decision tree. For each feature, the orthogonal decision tree will exhaustively find a cutting-plane and score it based on one pre-defined criteria. The nature of impurity criterion is to measure the skewness of the distribution of different classes in the set of samples reaching the node. Distributions which are near uniform will receive low impurity score while high score is assigned to those distributions where the number of patterns of one class is much larger than the rests. Most of the popular decision tree induction algorithms work by optimizing an impurity measure. However, many of the impurity measures are not differentiable with respect to the hyperplane parameters. Thus, those algorithms employ some search techniques for finding the best hyperplane at each node. For example, CART-LC [17] uses a deterministic hill-climbing algorithm, OC1 [18] uses a randomized search based on CART-LC. Both of these approaches search in one dimension at a time, which becomes computationally cumbersome in high-dimensional feature spaces. Moreover, both algorithms face the local optimum problem. Multiple trials or re-starts can be employed to decrease the possibility of ending up with a locally optimal solution. One may argue that one alternative method to optimize the function in all feature dimension is evolutionary approaches, and there does exist such work [19, 20]. This approach is claimed to tolerant to noisy evaluations of the rating function and most importantly, may facilitate optimizing multiple rating functions simultaneously [21, 22]. In [23], Stronger randomization was involved in the tree classifier. [24] generate this idea to oblique case. Other existing works relating to the decision tree integration and its generalization can be found in [25–27]. However, besides higher computational complexity, those search approaches involve several parameters which are difficult for practitioners to tune. In [28], a “Decision Tree Support Vector Machine” (DTSVM) is proposed. This work uses standard support vector machine with RBF kernel to generate the optimal hyperplane. However, besides carefully parameter tuning via extra cross-validation, this method is rather slow. We refer the readers to [61] for more information about decision tree induction methods.

However, without really capturing the geometric structure of class regions, the impurity measure only depend on the distribution of different classes on each side of the hyperplane. In this case, the impurity measure of the hyperplane is the same if one changes the label of the data without changing the relevant features of each class on either side of a hyperplane. This is first reported in [16]. Instead of only considering the label information, the geometric structure considers the internal data structure which measures the distance of the data sample to the decision hyperplane. In this case, the hyperplane may change if any of the relevant feature changes.

3.2 Oblique Decision Tree Ensemble

In order to find a hyperplane which captures the geometric structure of class distributions, the authors in [16] proposed the so called “geometric decision tree” which borrowed ideas from a recent variant of the support vector machines method. The multisurface proximal SVM (MPSVM) [29] finds two clustering hyperplanes, one for each class, where each hyperplane is generated such that it is closest to one class of data and furthest from the other class. In this case, the data instance is classified based on the distance to the hyperplanes. In their work, at each node, they use MPSVM to find two clustering hyperplanes. They choose one of the angle bisector of these two hyperplanes with better impurity measure. As the tree grows, the samples reaching the subsequent nodes become fewer and the small sample size problem arises. The authors propose to use NULL space [30] method to tackle this. The work in [29] has one extra parameter which defines minimum number of data samples for an internal node to be further splitted. The authors proposed to use cross-validation to find a proper value for this parameter.

In this section, we will describe the details of the oblique decision tree ensemble based on the similar idea as done in [16].

3.2.1 Oblique decision hyperplane

As we mentioned earlier, MPSVM is leveraged to capture the geometric structure of the data available at each internal node. Originally, MPSVM is proposed for binary classification problems. Denoted by matrix A with size of $n_1 \times m$ the samples from class 1 and matrix B with size $n_2 \times m$ the samples of class 2, MPSVM seeks two hyperplanes in R^m :

$$\begin{aligned} X * W_1 - b_1 &= 0, \\ X * W_2 - b_2 &= 0, \end{aligned} \tag{3.1}$$

where the hyperplane (W_1, b_1) is closest to the data of class 1 and furthest from the data in class 2. In the same way, the hyperplane (W_2, b_2) is closest to the data in class 2 and furthest from the data in class 1. To obtain the first plane of Eqn (3.1), one can simply minimize the sum of squares of two-norm distances between each of the samples of class 1 to the plane divided by the squares two-norm distance between each of the samples of class 2 to the plane.

Chapter 3. Oblique Decision Tree Ensemble Via MPSVM

Thus, it is straightforward to obtain the first hyperplane by solving the following optimization problem:

$$\min_{(W,b) \neq 0} \frac{\|AW - eb\|^2 / \|W_b\|^2}{\|BW - eb\|^2 / \|W_b\|^2}, \quad (3.2)$$

and similarly, the second plane can be obtained by solving:

$$\min_{(W,b) \neq 0} \frac{\|BW - eb\|^2 / \|W_b\|^2}{\|AW - eb\|^2 / \|W_b\|^2}, \quad (3.3)$$

where $\|\cdot\|$ denotes the two-norm and e is a vector of ones with the same dimension as AW and BW . By defining:

$$\begin{aligned} G &= [A - e]'[A - e], \\ H &= [B - e]'[B - e], \\ Z &= [W \quad b]', \end{aligned} \quad (3.4)$$

the optimization problem in Eqn (3.1) becomes:

$$\min_{z \neq 0} \frac{z'Gz}{z'H z}, \quad (3.5)$$

similarly, by defining:

$$\begin{aligned} L &= [B - e]'[B - e], \\ M &= [A - e]'[A - e], \end{aligned} \quad (3.6)$$

the optimization problem Eqn (3.1) becomes:

$$\min_{z \neq 0} \frac{z'Lz}{z'M z}, \quad (3.7)$$

Thus, the two clustering hyperplanes can be found by the eigenvectors corresponding to the smallest eigenvalues of the following two generalized eigenvalue problems:

$$\begin{aligned} Gz &= \lambda HZ, z \neq 0, \\ Lz &= \gamma Mz, z \neq 0, \end{aligned} \quad (3.8)$$

Due to the way they are defined, these clustering hyperplanes capture the geometric data property that are useful for discriminating the classes. Hence, a hyperplane that passes in between them could be good for splitting the feature space. One can simply take one of the bisectors of the clustering hyperplanes as the split rule at a non-leaf node inside a decision

tree. In general there would be two angle bisectors. The one with better discriminant ability based on impurity measure can be selected. If the two clustering hyperplanes happen to be parallel to each other, then the hyperplane midway between the two may served as the split rule.

Decision Trees are naturally multi-class classifiers. Thus, we have to transform the multi-class problem into binary case for MPSVM. Several approaches have been proposed to transform the multiclass problem into several binary ones such as one against all [99], one against one [100], DAG [101], ECOC [102] and so on. Promising as they are reported, they are used as pre-processing approaches and they increase the computational complexity greatly. Recursive partitioning mechanism makes decision tree advantageous over other binary algorithms. In order to capture the geometric structure of the data, we propose to separate all classes into two hyper-classes based on their separability. In statistics, the Bhattacharyya distance measures the similarity between two discrete or continuous probability distributions, which is considered a good measure for class separability between two normal classes w_1 and w_2 , $N(\mu_1, \Sigma_1)$ and $N(\mu_2, \Sigma_2)$. Multivariate Gaussian distribution is chosen here due to several reasons as reported in [31]. First, it is the most natural distribution and most importantly, the sum of a large number of independent random distributions obeys Gaussian distribution. For a given mean and variance, it has the maximum uncertainty of all distributions. Moreover, it is an appropriate model in many situations. The Bhattacharyya distance is defined as

$$B(w_1, w_2) = \frac{1}{8}(\mu_2 - \mu_1)^T \left(\frac{\Sigma_1 + \Sigma_2}{2} \right)^{-1} (\mu_2 - \mu_1) + \frac{1}{2} \ln \frac{|(\Sigma_1 + \Sigma_2)/2|}{\sqrt{|\Sigma_1||\Sigma_2|}}, \quad (3.9)$$

Our multiclass to 2-class transformation approach is presented in Table 3.1.

Table 3.1 Multi2binary Algorithm

multi2binary algorithm

Input:

$X(n \times m)$ is the training dataset, where n is the number of training samples, m is the dimension of each data.

$Y(n \times 1)$ are the labels of the training set.

$\{w_1, \dots, w_c\}$ the set of class labels

Output:

W_p and W_n are the two groups (hyper-classes).

- for $i = 1, \dots, c$
 - obtain the pairwise Bhattacharyya distance between class w_i and w_k for $k = i + 1, \dots, c$
 - find the pair of classes with the largest Bhattacharyya distance, name them as w_p and w_n and put them in group W_p and W_n respectively.
 - for every remaining class, say w_k assign it to the group W_p if $B(w_k, w_p) < B(w_k, w_n)$, otherwise assign it to the group W_n .
-

3.2.2 Small Sample Size Problem

Decision Tree works by recursively partitioning the data into 2 groups. As the tree grows, the number of data samples reaching a internal node will become less and less. In solving the generalized eigenvalue problem in Eqn (3.8), it cannot be guaranteed that the matrix G and L are always positive definite. This small sample size problem is more likely to occur in the process of decision tree induction. It is clear that matrix G and L have the size of $(m+1) \times (m+1)$ always while as the tree grows, the number of samples reaching lower nodes will be limited. In this case, matrix G and L will become rank deficient. The work in [16] propose to use NULL space approach as demonstrated in Table 3.2 to regularize the matrix G and L . However, to the best of our knowledge, NULL space approach is very unstable [31]. Thus, the approach in [16] needs careful control of minimum number of data samples for an node to be further splitted. The threshold for the minimum number of data samples can only be obtained through cross validation.

However, for a decision tree ensemble such as Random Forest, it is widely accepted that every single tree should grow to its largest depth to gain variance as much as possible. In this work, we apply two different regularization approaches because we also found that the NULL space approach often lower the performance with the perturbation of the data when there are limited data samples as highlighted in [103]. Decision tree pruning or tuning the threshold mentioned above may solve this problem because most of the node in high depth may be discarded. Those nodes are highly unstable because the number of data samples reaching them are quite limited. However, in decision tree ensembles, all base classifiers should be encouraged to grow to the largest extent. There is no consensus as to whether pruning or growing to the largest extend is the best strategy for decision tree ensembles [75].

Here, we apply two regularization approaches: Tikhonov regularization [104] and axis-parallel (or orthogonal) split regularization. Tikhonov regularization is often used to regularize least squares and mathematical programming problems [105, 106] and other applications. It works by simply adding a small constant term to the diagonal entries of the matrix to be regularized. In our case, suppose G becomes rank deficient, G is modified as

$$G' = G + \delta \times I, \quad (3.10)$$

The same operation can be applied to matrix H when it becomes rank deficient.

An straightforward alternative to Tikhonov regularization is the axis-parallel split regularization. If the matrix of G or H turns out to be rank deficient in a node, one can simply use the axis-parallel split to carry on the tree induction process. Hence, the decision tree grows using heterogeneous test functions. From the root node to the current node, MPSVM based

Table 3.2 NULL space approach

NULL space algorithm:
<hr/> Input : G (class 1), H (class 2) as defined in Eqn 3.4 output : clustering hyperplane $[w \ b]^T$
<ol style="list-style-type: none"> 1. Suppose matrix G has rank $r < m + 1$. calculate $Q = [\alpha_1, \dots, \alpha_{m+1-r}]$ whose columns are an orthonormal basis for the NULL space of G. 2. Project all the data samples in matrix H to the null space of G. Every vector x belonging to matrix H (each row of H) after projection will become xQQ^T. Then the matrix H after projection will become $H' = \sum_{x \in H} QQ^T x^T x QQ^T = QQ^T H QQ^T$. Similarly, the matrix $G' = QQ^T G QQ^T$ is the projected matrix of G. 3. Now G' would be zero because columns of Q span the null space of G. Hence, the eigenvector corresponding to the largest eigenvalue of H' is selected as the desired vector (clustering hyperplane).

hyperplane is associated. From the current node to the leaf node, the decision tree turns back to use the axis-parallel splitting method.

3.2.3 Details of Oblique Decision Tree Ensemble

The extension of a single decision tree to decision tree ensemble is straightforward. Here we consider two decision tree ensemble: Random Forest and Rotation Forest. Denote the training set as X , which is a $n \times m$ data matrix. Each row stands for a datum and each column is a feature. For Random Forest, we sample n samples with replacement from X to generate a unique training set for each decision tree. For Rotation Forest, we generate a unique rotation matrix as demonstrated in Table 2.3 and Eqn (2.2). In this thesis, RaF stands for the Random Forest and RoF represents the Rotation Forest. MP RaF-T, MP RaF-P, and MP RaF-N represent the MPSVM based Random Forests with Tikhonov, axis-parallel, and NULL space regularization, respectively. MP RoF-T, MP RoF-P, MP RoF-N represent the MPSVM based Rotation Forest with Tikhonov, axis-parallel, NULL space regularization, respectively.

3.3 Computational Complexity Analysis

Suppose we have n data samples in total and at each node m features are selected to search for the decision hyperplane. Without loss of generality, suppose each tree grows to its largest extent. In order to simplify the analysis, we further assume each tree is balanced, i.e., each decision node will divide the data into two branches with equal size. Denote by L the

maximum level of a decision tree (the root node has level 0 and the leaf nodes are in level L). In this case, $2^L = n$.

Considering an internal node at level i , there are 2^i nodes in this level and each node is associated with $\frac{n}{2^i}$ data samples. The uni-variate split algorithm firstly rank each feature and try to find a threshold to optimize the impurity criterion. Regardless of the computational time for calculating the impurity value, the computation complexity to find such a “best-split” takes time on the order of $m \frac{n}{2^i} \log \frac{n}{2^i}$ ($\frac{n}{2^i} \log \frac{n}{2^i}$ for each feature). For the MPSVM based decision tree, according to [29], complexity of the generalized problem is of order m^3 .

It is straightforward to show that the total complexity of orthogonal decision tree is:

$$C_{Orth} = \sum_{i=0}^L 2^i (m \frac{n}{2^i} \log \frac{n}{2^i}); \quad (3.11)$$

The overall complexity of an orthogonal decision tree, which is denoted by C_{Orth} , is around $O(mn(\log n)^2)$.

For oblique decision tree, the overall complexity can be calculated as:

$$C_{Obli} = (2n - 1) * m^3; \quad (3.12)$$

And it is straightforward to show C_{Obli} is about $O(nm^3)$. In the proposed decision tree ensemble, we usually randomly sample several features in each internal node, in this case, m is much less than $\log n$. In this case, $C_{Obli} \ll C_{Orth}$.

3.4 Experiment

3.4.1 A Toy Example

We firstly choose a toy example to demonstrate the superiority of different methods. A three class artificial spirals dataset is generated, as demonstrated in Fig 3.1a. The figure in Fig 3.1b demonstrates the hierarchical partitions and surface for a single axis parallel decision tree. We can clearly see that the spiral decision boundary is approximated by many “stage-like” decision hyperplanes. The decision boundary can be somehow smoothed by averaging many decision trees, which can be demonstrated by Fig 3.1c. The decision boundary in this case is obtained by a conventional orthogonal random forest. The last two figures show that the proposed oblique random forest can lead to much “smoother” decision surfaces which better represent the data than the axis-parallel equivalent.

Figure 3.1 Decision boundary of the spirals dataset.

3.4.2 Preliminary Results about Oblique Decision Tree Ensemble

In this section we mainly present the preliminary results of the proposed method [107]. Experiments were conducted to evaluate the performances of the MPSVM based Random Forests and Rotation Forests with different regularization methods. In order to compare fairly, classification and regression tree (CART) [17] was used as the base classifier for all

Table 3.3 Details of 44 datasets.

Datasets	Classes	Samples	Features	Datasets	Classes	Samples	Features
Adult	2	32561	123	Mam-masses	2	961	5
AFP-Pred	2	9974	119	Orl	40	400	1024
Australian	2	690	14	Ozone	2	2536	72
Balance scale	3	625	4	Page block	5	5473	10
Banknote	2	1372	4	Parkinsons	2	195	22
Biodeg	2	1055	41	Pima-diabetes	2	768	8
Blood transfusion	2	748	4	Planning Relax	2	182	12
BLProt	2	18943	545	Seeds	3	210	7
Breast cancer	2	699	9	Sonar	2	208	60
Breast tissue	6	106	9	Spambase	2	4601	57
Climate model	2	540	18	Statlog-segment	7	2310	19
DNA	3	3186	180	Teaching assistant	3	151	5
Ecoli	8	336	7	Twonorm	2	7400	20
Fertility	2	100	9	User Knowledge	4	258	5
Glass	7	214	9	Vehicle	4	846	18
Harberman	2	306	3	Vertebral-2C	2	310	6
Heart	2	270	13	Vertebral-3C	3	310	6
Hepatitis	2	155	19	Waveform1	3	5000	21
segment	7	2310	19	Waveform2	3	5000	40
Ionosphere	2	351	34	Wine	3	178	13
Iris	3	150	4	Wine quality(red)	6	1599	11
Lymphography	4	296	18	Yale	15	165	1024

ensemble methods. Gini-impurity which is defined as

$$Gini(t) = \frac{n_t^l}{n_t} \left[1 - \sum_i^c \left(\frac{n_{w_i}^l}{n_t^l} \right)^2 \right] + \frac{n_t^r}{n_t} \left[1 - \sum_i^c \left(\frac{n_{w_i}^r}{n_t^r} \right)^2 \right], \quad (3.13)$$

where n_t is the number of data samples reaching this node, n_t^l , n_t^r is the number of data samples that reach the left and right child nodes of the current node, respectively, $n_{w_i}^l$, $n_{w_i}^r$ represents the number of samples of class w_i in the left and right nodes, respectively, is employed as the impurity criterion in all cases.

All methods are tested on 44 real-world benchmark classification datasets from various research fields. Two face recognition data sets (ORL, YALE) are chosen here. Two bio-informatics data sets (AFP-Pred [108] and BLProt [109]) are also selected. The remaining data sets are from UCI repository [15]. We follow the works of [110, 111] to perform feature extraction from the face images. After this step, the ORL has a size of 400×59 and Yale has a size of 165×14 . For each dataset, 10 times 3-fold cross validations are carried out. But for some dataset where the data partition was already performed, then we just use the training set to train the model and put all the testing set on the model. This procedure was conducted 10 times. The details of those dataset are summarized in Table 3.3.

The simulations with all datasets presented in Table 3.3 are carried out in Matlab R2010b with Intel (R) Core(TM) i5, 3.20-GHz CPU and 4-GB RAM. For all ensemble methods, the ensemble size M can be regarded as a hyperparameter. Generally, the performances of deci-

sion tree ensemble will not become worse if we increase M . Considering the computational complexity, we conduct all experiments with $M = 50$ in this section. The parameter “ $mtry$ ” in Random Forest, which represents the number of features randomly selected and controls the randomness of the algorithm, is set to be $round(\sqrt{m})$ where m is the dimensionality of a the feature vector. The Tikhonov regularization term λ is set to be 0.01 for all experiments in this work. One may argue that it may be optimized by either a separate validation dataset or by out-of-bag data. However, it may become computationally infeasible in that case since there are a lot of nodes in the ensemble especially for very large number of training samples.

Results for Oblique Random Forest

The details results of those dataset are summarized in Table 3.4. The boldface indicates that the best result.

From Table 3.4, we can see in most cases, the MPSVM based Random Forest outperforms the standard Random Forest or at least achieves comparable performance. More specifically, MPSVM based RaF with Tikhonov regularization outperforms standard Random Forest in 32 out of 44 datasets. MPSVM based RaF with axis-parallel split regularization outperforms standard Random Forest in 31 out of 44 cases. However, for the NULL space based oblique Random Forest, we do not observe any accuracy advantage.

Results for Oblique Rotation Forest

Similarly, we evaluate the proposed MPSVM based Rotation Forest with the same dataset. Results can be found in Table 3.5.

From Table 3.5, we can see none of the proposed method outperforms the baseline Rotation Forest. MPRoF-T loses 40 times, MPRoF-P loses 23 times, and MPRoF-N only wins in 9 cases. The reason is that in the induction of a CART in Rotation Forest, the dimensions of matrix G and H in Eqn (3.8) are always $(m + 1) \times (m + 1)$ (where m is the dimension of the feature vector. For RaF, this dimension turns out to be $(mtry + 1) \times (mtry + 1)$). As the tree grows up, the number of the data samples reaching a given node will be smaller than m . It is easy to observe that in Rotation Forest, there are more nodes to be regularized than the Random Forest. The eigenvalue problem is somehow sensitive to the Tikhonov regularization and NULL space approaches [103]. In our case, the problem tends to be more serious as more and more decision nodes need to be regularized.

In this case, we use random subspace approach to reduce the size of matrix G and H . We name the resulting method as Random Rotation Forest (RRoF). In each node, the optimization of hyperplane is conducted on a randomly selected sub feature set. In this case, RaF and

Chapter 3. Oblique Decision Tree Ensemble Via MPSVM

Table 3.4 Classification Accuracy of Random Forest and its MPSVM based variants.

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N
Adult	84.37	83.40	84.33	82.06
AFP-pred	74.00	73.43	72.78	78.93
Australian	87.01	86.42	86.90	86.25
Balance scale	84.77	89.02	88.19	89.10
Banknote	99.17	99.91	99.91	99.89
Biodeg	86.31	86.68	86.52	85.78
Blood transfusion	77.09	78.05	77.49	77.50
BLProt	81.22	81.27	83.85	70.40
W-breast cancer	96.45	96.72	96.72	96.92
Breast tissue	68.01	68.87	67.55	60.94
Climate model	92.28	92.06	91.85	91.59
DNA	94.24	91.30	93.92	80.07
Ecoli	84.55	85.46	84.76	84.40
Fertility	87.70	87.90	88.20	88.00
Glass	96.92	94.21	96.64	59.67
Harberman	70.39	72.39	70.29	71.34
Heart	81.93	83.74	82.96	83.07
Hepatitis	82.97	83.61	83.94	79.48
segment	95.33	94.42	95.51	89.55
Ionosphere	93.62	92.68	93.96	89.74
Iris	94.60	97.60	95.67	97.40
Lymphography	93.10	95.37	93.01	89.26
Mam-masses	82.09	82.24	81.93	82.38
Orl	77.15	94.10	88.20	93.45
Ozone	97.07	97.12	97.10	97.13
Page Block	97.33	97.28	97.35	95.37
Parkinsons	90.21	91.23	90.82	86.36
Pima-diabetes	75.81	75.91	75.98	76.77
Planning relax	69.62	70.49	70.66	71.21
Seeds	92.14	94.19	93.14	92.43
Sonar	79.13	82.12	78.99	81.44
Spambase	94.60	93.68	94.68	93.77
statlog-segment	97.38	97.13	97.48	94.89
Teaching assistant	54.37	55.10	55.56	53.38
Twonorm	96.51	97.47	97.51	97.45
User Knowledge	91.51	91.20	91.82	90.15
Vehicle	74.86	76.30	76.34	70.76
Vertebral-2C	82.58	84.61	84.58	84.10
Vertebral-3C	83.35	83.61	83.77	82.48
Waveform1	84.34	85.70	85.50	85.70
Waveform2	84.47	85.44	85.32	85.34
Wine	97.30	97.64	98.26	96.24
Wine quality(red)	66.62	67.22	67.10	59.42
Yale	56.44	80.00	64.56	78.22

	MPRaF-T	MPRaF-P	MPRaF-N		RaF	MPRaF-T	MPRaF-P	MPRaF-N
win-tie-lose	32-0-12	33-0-11	22-0-22	mean accuracy	84.80	86.24	85.72	83.63

“MPRaF-T”, “MPRaF-P”, “MPRaF-N” means MPSVM based Oblique Random Forest with Tikhonov, axis-parallel split, and NULL space regularization, respectively. The “win-tie-lose” value means the number of times that this method wins, ties and loses to RaF, respectively. “mean accuracy” stands for the overall accuracy across all the datasets for each method.

Table 3.5 Classification Accuracy of Rotation Forest and its MPSVM based variants.

Datasets	RoF	MPRoF-T	MPRoF-P	MPRoF-N
Adult	81.83	83.55	81.83	83.81
AFP-pred	79.79	64.16	79.96	61.09
Australian	85.58	80.71	83.30	81.91
Balance scale	86.90	88.00	87.81	83.98
Banknote	99.92	99.74	99.85	99.83
Biodeg	86.87	81.92	87.08	86.03
Blood transfusion	74.80	73.30	75.03	76.32
BLProt	81.20	68.70	81.29	70.81
W-breast cancer	96.47	93.73	96.01	96.05
Breast tissue	67.36	54.53	66.42	49.06
Climate model	92.24	90.91	92.59	91.98
DNA	93.72	90.81	93.68	94.41
Ecoli	84.55	79.32	84.73	82.59
Fertility	86.10	78.80	85.70	85.10
Glass	97.57	93.36	97.76	46.82
Harberman	68.30	67.75	66.96	67.81
Heart	81.30	76.26	78.78	79.96
Hepatitis	80.90	76.97	80.39	79.35
segment	95.08	86.00	94.98	66.72
Ionosphere	94.30	84.37	94.47	85.76
Iris	94.73	95.00	95.33	94.93
Lymphography	94.19	92.91	94.32	91.62
Mam-masses	77.39	76.87	76.74	79.49
Orl	89.90	84.50	90.00	81.00
Ozone	97.08	96.26	97.13	96.34
Page Block	97.22	95.98	97.08	95.76
Parkinsons	90.10	81.69	90.41	75.64
Pima-diabetes	74.44	71.05	72.02	73.26
Planning relax	67.86	58.35	63.90	66.32
Seeds	92.86	91.14	92.05	88.86
Sonar	82.26	73.89	77.16	71.49
Spambase	94.94	89.52	94.52	93.88
statlog-segment	98.12	92.63	98.20	81.27
Teaching assistant	57.35	55.63	57.22	57.28
Twonorm	97.49	95.93	97.39	97.17
User Knowledge	91.32	87.36	92.64	91.40
Vehicle	77.62	77.04	79.69	68.96
Vertebral-2C	85.23	79.81	85.52	77.23
Vertebral-3C	84.54	78.19	84.65	75.03
Waveform1	85.61	80.63	85.06	84.83
Waveform2	85.30	80.53	85.02	84.71
Wine	94.10	92.36	94.55	92.25
Wine quality(red)	66.74	57.10	67.03	53.42
Yale	73.56	78.89	74.78	76.67

	MPRoF-T	MPRoF-P	MPRoF-N		RoF	MPRoF-T	MPRoF-P	MPRoF-N
win-tie-lose	4-0-40	21-0-23	9-0-35	mean accuracy	85.58	81.28	85.25	79.95

“MPRoF-T”, “MPRoF-P”, “MPRoF-N” means MPSVM based Oblique Rotation Forest with Tikhonov, axis-parallel split, and NULL space regularization, respectively. The “win-tie-lose” value means the number of times that this method wins, ties and loses to RoF, respectively. “mean accuracy” stands for the overall accuracy across all the datasets for each method.

RRoF differ in the way that they perturb the data: the former employs bagging to create a data subset to inject diversity and latter uses different rotation matrices for each decision tree. Experiments was carried out to evaluate random subspace concept in Rotation Forest. The results are summarized Table 3.6.

Several observations can be find in Table 3.6. Firstly, the RRoF achieves better performance than RoF. This is aligned with the work in [32] as the decision tree in RRoF is more unstable than those in RoF. Moreover, the MPSVM can improve the performance of RRoF. We can see MPRRoF-T outperforms RRoF in 25 out of 44 cases and MPRRoF-P can even win 33 times. We still can not observe any accuracy advantage of Null space based methods.

Statistical point of view

A Friedman test [112] was carried out to find out the statistical significance of the results. Friedman test is reported to be more robust than other approach [82, 112]. It gives the algorithms a rank for each data set separately. The lowest rank is associated with the best performing algorithm. Denoted by r_i^j the rank of the j th of k algorithms on the i th of N data sets. The Friedman test compares the average ranks of algorithms, $R_j = \frac{1}{N} \sum_i r_i^j$. Under the null-hypothesis, which states that all the algorithms are equivalent and so their ranks R_j should be equal, the Friedman statistic

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right], \quad (3.14)$$

is distributed according to χ_F^2 with $k-1$ degrees of freedom, when N and k are big enough. In that case, Friedman's χ_F^2 is undesirably conservative and derived a better statistic:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1) - \chi_F^2}, \quad (3.15)$$

which is distributed according to the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom. If the null-hypothesis is rejected, a post-hoc test, namely the Nemenyi test [113], can be used to check whether the performance of two among k classifiers is significantly different. If the corresponding average ranks differ by at least the critical difference

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}, \quad (3.16)$$

where critical values q_α are based on the Studentized range statistic divided by $\sqrt{2}$.

Table 3.6 Classification Accuracies of the proposed Random Rotation Forest and its MPSVM based variants.

Datasets	RRoF	MPRRoF-T	MPRRoF-P	MPRRoF-N
Adult	83.31	83.95	83.38	81.91
AFP-pred	76.41	73.16	74.11	77.56
Australian	86.42	86.42	86.61	86.49
Balance scale	86.66	89.30	88.70	89.81
Banknote	99.88	100.00	100.00	100.00
Biodeg	86.92	87.18	86.65	87.55
Blood transfusion	75.76	77.59	76.14	77.63
BLProt	82.19	81.92	82.57	66.94
W-breast cancer	96.74	96.39	96.67	96.80
Breast tissue	69.53	68.87	70.19	65.85
Climate model	92.15	92.30	91.26	91.76
DNA	94.38	91.29	93.68	90.88
Ecoli	85.51	85.24	85.80	84.38
Fertility	86.90	88.00	87.70	88.10
Glass	96.68	94.95	97.43	69.86
Harberman	68.86	71.21	70.13	69.67
Heart	81.85	83.11	82.37	83.59
Hepatitis	82.45	84.71	84.13	83.74
segment	95.75	95.62	96.01	94.31
Ionosphere	94.36	93.39	94.70	90.77
Iris	94.40	96.47	95.40	96.33
Lymphography	95.03	95.41	95.20	93.99
Mam-masses	78.68	81.48	78.61	81.38
Orl	89.40	94.55	91.95	85.85
Ozone	97.10	97.12	97.10	97.10
Page Block	97.22	97.20	97.24	96.78
Parkinsons	91.33	92.15	92.15	88.41
Pima-diabetes	75.51	74.87	75.26	75.81
Planning relax	70.04	71.26	71.10	71.14
Seeds	93.19	94.71	95.00	93.62
Sonar	81.54	83.46	80.77	83.17
Spambase	95.08	94.51	95.04	94.42
statlog-segment	97.93	97.88	98.04	97.24
Teaching assistant	56.42	54.70	56.29	57.02
Twonorm	97.34	97.52	97.56	97.47
User Knowledge	89.19	88.91	89.42	89.65
Vehicle	77.25	77.02	77.81	72.85
Vertebral-2C	85.19	85.26	85.45	84.97
Vertebral-3C	84.90	85.10	85.06	84.26
Waveform1	85.66	85.67	85.63	86.00
Waveform2	85.53	85.53	85.74	85.70
Wine	97.02	97.98	98.09	97.36
Wine quality(red)	67.08	66.90	67.50	60.46
Yale	72.11	82.33	75.11	78.89

	MPRRoF-T	MPRRoF-P	MPRRoF-N
win-tie-lose	25-2-17	33-0-11	24-0-20

	RRoF	MPRRoF-T	MPRRoF-P	MPRRoF-N
mean accuracy	85.84	86.42	86.27	84.72

“MPRRoF-T”, “MPRRoF-P”, “MPRRoF-N” means MPSVM based Random Rotation Forest (subspace involved in each node) with Tikhonov, axis-parallel split, and NULL space regularization, respectively. The “win-tie-lose” value means the number of times that this method wins, ties and loses to RRoF, respectively. “mean accuracy” stands for the overall accuracy across all the datasets for each method.

Chapter 3. Oblique Decision Tree Ensemble Via MPSVM

Thus, we first rank the performance of RaF and RoF and RRoF on each data set and take average across all the datasets. After simple calculations, the average ranks for RaF, RoF and RRoF are 2.24, 2.07, 1.69, respectively. Then

$$\begin{aligned}\chi_F^2 &= \frac{12 \times 44}{3 \times 4} (2.24^2 + 2.07^2 + 1.69^2 - \frac{3 \times 4 \times 4}{4}) \\ &= 6.98; \\ F_F &= \frac{43 \times 6.98}{44 \times 2 - 6.98} = 3.71,\end{aligned}\tag{3.17}$$

as there are 44 data sets and three algorithms, F_F is distributed according to the F –distribution with $3-1=2$ and $(3-1)(44-1)=86$ degrees of freedom. The critical value of $F(2,86)$ with $\alpha=0.05$ is within $[2.68, 2.76]$. In this case, the null-hypothesis is rejected. Based on The Nemenyi test, the critical value here is $CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} = 2.343 * \sqrt{3 * 4 / (6 * 44)} \simeq 0.5$ with $\alpha = 0.05$. Since 2.24 and 1.69 are the only pair of average rank whose difference is larger than 0.5, we can conclude that the RRoF is significantly better than RaF. Though RRoF works slightly better than RoF, we do not observe statistical differences.

Similarly, the average ranks for RaF, MPRaF-T, MPRaF-P, MPRaF-N are 2.95, 1.97, 2.11, 2.97 respectively. Then

$$\begin{aligned}\chi_F^2 &= \frac{12 \times 44}{4 \times 5} (2.95^2 + 1.97^2 + 2.11^2 + 2.97^2 - \frac{4 \times 5 \times 5}{4}) \\ &= 22.61, \\ F_F &= \frac{43 \times 22.61}{44 \times 3 - 22.61} = 8.89,\end{aligned}\tag{3.18}$$

With 4 algorithms and 44 data sets, F_F is distributed according to the F –distribution with $4-1=3$ and $(4-1) \times (44-1) = 129$ degrees of freedom. The critical value of $F(3,129)$ for $\alpha = 0.05$ is smaller than 2.68, so we reject the null-hypothesis. In this case, the critical value is $CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} = 2.569 * \sqrt{4 * 5 / (6 * 44)} \simeq 0.71$ for $\alpha = 0.05$. So, it is obvious that the MPRaF-T and MPRaF-P perform significantly better than standard Random Forests and MPRaF-N. Table 3.7 summarizes the statistical test for Random Forest.

Table 3.7 Significance of the difference for Random Forest.

Method	RaF(2.95)	MRaF-T (1.97)	MRaF-P(2.11)	MRaF-N(2.97)
RaF(2.95)				
MRaF-T(1.97)	yes			yes
MRaF-P(2.11)	yes			yes
MRaF-N(2.97)				

The numbers in the bracket represent the average rank for the algorithm, *yes* means the method of the row is significantly better than the method of of the column. Empty entry means there is no significant difference between the method of row and the method of the column.

Similarly, the average ranks for the RRoF, MPRRoF-T, MPRRoF-P, MPRRoF-N are 2.91, 2.19, 2.15, 2.75, respectively. Then

$$\begin{aligned}\chi_F^2 &= \frac{12 \times 44}{4 \times 5} (2.91^2 + 2.19^2 + 2.15^2 + 2.75^2 - \frac{4 \times 5 \times 5}{4}) \\ &= 11.86, \\ F_F &= \frac{43 \times 11.86}{44 \times 3 - 11.86} = 4.25,\end{aligned}\tag{3.19}$$

We reject the null-hypothesis. Thus, similar conclusion can be drawn that MPRRoF-T and MPRRoF-P lead to significantly better performance than RRoF and MPRRoF-N as shown in Table 3.8.

Table 3.8 Significance of the difference for Random Rotation Forest

Method	RRoF(2.91)	MRRoF-T (2.19)	MRRoF-P(2.15)	MRRoF-N(2.75)
RRoF(2.91)				
MRRoF-T(2.19)	yes			
MRRoF-P(2.15)	yes			
MRRoF-N(2.75)				

The number in the bracket represent the average rank for the algorithm, *yes* means the method of the row is significantly better than the method of column. Empty entry means there is no significant difference between the method of row and the method of the column.

3.4.3 Comprehensive Evaluation with Other 179 Methods on 121 Datasets

To the best of author’s knowledge, the most extensive and comprehensive evaluation in terms of both number of classifiers and number of datasets is reported in [6] which involves 121 datasets with 179 classifiers arising from 17 families. Results indicate that RaF achieves the top performance and RoF gets the rank of 15 among 179 classifiers. In order to benchmark our previous work, MPSVM based decision tree ensembles are evaluated in this framework and the results are summarized in this section.

Evaluation on 121 datasets

We choose MPRAF-P as a variant of RaF since it has less parameters to tune compared with MPRAF-T. For the Rotation Forest, we choose MPRRoF-P for the same reason.

All 121 datasets are from the UCI repository [15], the details of the datasets are presented in Table A.1. We normalize each feature by removing the mean and dividing by its variance, as done in [6]. Randomized stratified sampling is employed to make sure one training and one test set are generated (each with 50% of the available patterns), where each class has the same number of training and test patterns. Parameter tuning is performed on this couple of sets, which favors those parameters with best performance on the test set [6]. We only tune

Chapter 3. Oblique Decision Tree Ensemble Via MPSVM

one parameter in present work which is the number of features randomly selected at each node. Then, with the selected values for this parameter, a 4-fold cross validation is developed using the whole data [6]. However, for some data sets, where the training-testing partition is already available (such as annealing and audiology-std, among others), the classifier is trained and tested on the respective data sets. In this case, the test result is calculated on the test set. All the data partitions are the same as in [6].

Classification Results of Oblique Decision Tree ensemble on 121 datasets

The results are presented in the Table 3.9 for all datasets (we refer the readers to Table A.1 in Appendix A for the information of the datasets), where the values corresponding to “RF” are copied from the results of *parRF_t* in [6]. It is considered as the top 1 method in [6]. One may argue that the most intuitive way to compare classifiers is to compute the average accuracies over all datasets. However, their averages are meaningless if the results on different datasets are not comparable. Moreover, averaging of accuracies is also susceptible to outliers. They allow a classifier’s excellent performance on one dataset to compensate for poor performance. Further, a total failure on one problem can submerge fair results on most others [112]. Hence, in this study, we follow the method in [112], and use the rank of each classifier to reflect its performance [6]. This approach ranks the algorithms for each data set separately, the best performing algorithm getting the rank of 1, the second best rank 2..., average ranks are assigned in case of ties.

Following the procedure in [6], we firstly calculate the mean accuracy across all the 121 datasets. Results indicate that both the MPSVM based decision tree ensemble outperforms the top method in [6]. The Friedman ranking was also computed to sort the classifiers. The rank is increasing with the classifier error. Lower rank indicates better performance.

The mean rank for each method also shows better performance of MPSVM based decision tree ensembles. After plugging in our results with two MPSVM based decision tree ensemble, we have summarized the top 10 classifiers in Table 3.10. For the statistical test, we refer the reader to Chapter 6 for more details. Our MPSVM based Rotation Forest achieves the second best performance in this work. Note for this approach, only one parameter is tuned in the present work in order to significantly reduce the time for parameter tuning. Other parameters such as the dimension of the subspace when calculating the rotation matrix and the number of features randomly selected at each node are kept to be the default values as suggested in [75]. It is highly possible to further improve the result here by tuning these parameters for each dataset.

Table 3.9 Detailed results of the evaluations.

dataset	MPRaF-P	MPRRoF-P	RF	dataset	MPRaF-P	MPRRoF-P	RF
abalone	65.49	65.4	65.1	monks-1	61	61.4	61.1
ac-inflam	100	100	100	monks-2	66.44	67.13	65.7
acute-nephritis	100	100	100	monks-3	52.78	55.79	53.7
adult	85.96	84.93	86.1	mushroom	100	100	100
annealing	76.38	96.73	94	musk-1	88.45	88.87	89.9
arrhythmia	76.99	73.67	77.4	musk-2	96.92	96.59	97.7
audiology-std	82.56	83.14	72	nursery	99.59	99.54	99.7
balance-scale	90.87	90.71	85.3	OccMerl4D	82.25	82.45	81.6
balloons	81.25	81.25	66.7	occMerl2F	92.55	92.25	92.8
bank	90.13	89.38	89.9	oocTris2F	83.77	84.21	81.4
blood	77.41	76.47	74.9	OccTris5B	92.54	93.42	92.5
breast-cancer	74.65	72.18	73.1	optical	96.93	97.43	97
bc-wisc	97.57	97.86	96.6	ozone	97.16	97.2	97.1
bc-wisc-diag	97.36	97.36	96.8	page-blocks	97.37	97.15	97.5
bc-wisc-prog	80.1	85.2	80.9	parkinsons	89.29	91.33	89.2
breast-tissue	72.12	73.08	71.7	pendigits	96.12	96.89	95.5
car	98.32	97.34	98.3	pima	75.52	74.63	76.6
ctg-10clases	94.96	93.13	88.1	pb-MATERIAL	92.31	94.23	84
ctg-3clases	86.91	85.36	94.3	pb-REL-L	77.88	74.04	74.8
chess-krvk	83.83	79.11	86.4	pb-SPAN	63.04	71.74	65.1
chess-krvkp	99.53	99.22	99.4	pb-T-OR-D	89	89	88.2
conn-voting	61.47	61.01	62.5	pb-TYPE	68.27	69.23	64.9
conn-ben-sonar	83.17	84.62	82.7	planning	71.11	71.67	71.4
conn-ben-vowelg	99.03	100	99.4	plant-margin	85.38	84.44	86.2
connect-4	86.9	83.91	87.4	plant-shape	66.38	74.19	63.9
contrac	53.4	50.27	55.1	plant-texture	82.5	83.19	83.4
credit-approval	87.65	86.92	87.5	post-operative	72.73	64.77	61.2
cylinder-bands	80.86	81.64	80.3	primary-tumor	55.18	53.35	49.4
dermatology	98.08	98.08	97.8	ringnorm	92.43	88.24	96.6
echocardiogram	86.36	84.85	84.7	seeds	94.23	93.75	94.3
ecoli	87.8	87.8	87.2	semeion	93.59	94.6	94.1
energy-y1	96.09	96.48	97.8	soybean	36.7	42.09	89.6
energy-y2	89.32	90.23	92.1	spambase	94.87	95.61	95.1
fertility	90	88	88	spect	69.22	65.59	55.1
flags	68.23	62.5	70.1	spectf	91.98	91.98	92
glass	74.06	73.11	78	st-australian-credit	68.02	64.24	67.7
haberman-survival	73.03	70.72	67.7	st-german-credit	78.9	76	77.1
hayes-roth	87.5	84.82	92.9	st-heart	86.57	85.82	83
heart-cleveland	58.55	56.58	59.4	st-image	97.8	98.09	97.8
heart-hungarian	83.56	85.62	82.3	st-landsat	95.09	90.88	91
heart-switzerland	46.77	46.77	41.4	st-shuttle	99.97	99.93	100
heart-va	37.5	36.5	37.4	st-vehicle	78.32	78.44	75.8
hepatitis	82.69	81.41	82.6	st-plates	77.94	79.18	79
hill-valley	49.42	68.32	55.3	synthetic-control	99	99.33	98.3
horse-colic	60.29	60.29	88.2	teaching	62.5	60.53	62.1
ilpd-indian-liver	73.12	72.09	68.4	thyroid	98.95	98.92	98.9
image-segmentation	14.52	26.73	28.6	tic-tac-toe	97.8	98.43	99.1
ionosphere	94.02	94.89	93.2	titanic	78.68	78.95	79.1
iris	96.62	97.3	96.7	trains	87.5	87.5	91.7
led-display	72.8	71.7	71.7	twonorm	97.85	97.89	97.2
lenses	83.3	83.3	83.3	vc-2clases	84.09	85.71	85.2
letter	96.78	96.55	96.4	vc-3clases	85.06	84.09	86.4
libras	82.78	89.72	82.5	wall-following	99.7	95.6	99.6
low-res-spect	91.92	91.73	90.4	waveform	86.24	86.2	85.7
lung-cancer	53.13	59.38	60	waveform-noise	86.48	86.54	85.7
lymphography	84.46	87.16	87.9	wine	98.96	99.43	98.9
magic	87.27	86.55	88	wine-quality-red	68.19	69.13	69
mammographic	81.04	79.39	80.9	wine-quality-white	68.97	69.63	68.6
miniboone	93.38	94.05	93.8	yeast	62.26	62.47	63
molec-biol-promoter	84.62	90.38	88.7	zoo	99	99	98.1
molec-biol-splice	94.84	91.78	95.2				

3.5 Conclusion

As an unstable classifier, decision tree can be significantly improved by ensemble methods. State-of-the-art classifier (Random Forest) combines the concepts of bagging and random subspace and employs decision tree as base classifier. However, conventional random forest use orthogonal decision tree where in each node the splitting rule is conducted by exhaustively searching for a “best feature”. These approaches is usually time consuming and lack the ability of capture the geometric structure of the data samples. In this chapter, we propose to use multisurface proximal support vector machine to assist the search process of decision hyperplane for decision tree ensemble. Further, we propose two strategy to address the small sample size problem as the tree grows. Extensive experiments demonstrate the effectiveness of the proposed methods.

Table 3.10 Summarization of the top 10 classifiers. Methods other than MPRaF-P and MPRRoF-P are from [6].

Rank	Acc	Classifier	Rank	Acc	Classifier
32.20	82.05	MPRaF-P	39.20	81.19	svmPoly_t
33.67	82.35	MPRRoF-P	40.55	81.93	rforest_R
33.96	81.96	parRF_t	40.64	81.96	elm_kernel_m
34.13	82.30	rf_t	41.43	81.43	svmRadialCost_t
37.85	81.81	svm_C	43.64	81.04	svmRadial_t

We follow the name style of [6] and all the results of other methods are copied from it.

Diverse Random Vector Functional Link Networks Ensemble

Like decision tree, neural networks are also unstable classifiers. Typically, training algorithms of neural network aims at minimizing a particular loss function. Conventional approaches usually optimize the parameters based on the derivatives of the loss function. However, much of the power of neural network comes from the nonlinear function in the hidden units used to model the nonlinear mapping between the input and output. In this case, the learning problem turns out to be nonlinear least square problems which are usually solved iteratively. In this case, the error signal has to be propagated backwards to serve as a guidance for tuning the parameters [83]. Those methods are reported to be very slow [84] and may not converge to a single global minimum because there exist many local minima [85, 86] and also the resulting neural network is very weak in the real world noisy situations.

An alternative of the above methods is the randomization based methods. In this chapter we bring back the machine learning community a very classical methods : Random Vector Functional Link Networks. Firstly we conduct a very comprehensive evaluations and figure out the key component of RVFL. We also propose an ensemble RVFL methods which show comparative performance to the state-of-the-art methods.

We first refer the reader to Section 2.3 for a detailed introduction of RVFL. The core tenet of RVFL remains untouched though there are many RVFL variants in the literature. In this work, We choose the closed-form based RVFL and the following issues are investigated by using 121 UCI datasets as done in [6]

1. Effect of direct links from the input layer to the output layer.
2. Effect of the bias in the output neuron.

Chapter 4. Diverse Random Vector Functional Link Networks Ensemble

3. Performance of 6 commonly used activation functions as summarized in Table 4.1.
4. Performance of Moore-Penrose pseudoinverse and ridge regression (or regularized least square solutions) for the computation of the output weights.
5. Effect of range for randomly generated parameters in hidden neurons.

Table 4.1 Activation functions used in this work. s and y are the inputs and outputs, respectively.

activation function	formulation
sigmoid	$y = \frac{1}{1+e^{-s}}$
sine	$y = \text{sine}(s)$
hardlim	$y = (\text{sign}(s) + 1)/2$
tribas	$y = \max(1 - s , 0)$
radbas	$y = \exp(-s^2)$
sign	$y = \text{sign}(s)$

4.1 Evaluation Protocol

Exactly the same the pipeline as in Section 3.4.3 is used here. Randomized stratified sampling is employed to make sure one training and one test set are generated (each with 50% of the available patterns), where each class has the same number of training and test patterns. Parameter tuning is performed on this couple of sets to identify parameters with the best performance on the test set. There are two parameters in the present work. One is the number of hidden neurons, which is tuned over 3: 203 with a step-size of 20 [6]. The other one is λ in ridge regression in Eq. (2.4), which is set to be 2^{-C} and C is $-5 : 1 : 14$ [6]. Then, with the selected values for the tunable parameters, a 4-fold cross validation is developed using the whole data. However, for some datasets where the training-testing partition is already available (such as annealing and audiology-std, among others), the classifier is trained on the predefined training set and evaluated on the test set. In this case, the test result is calculated on the test set [6]. Each input feature is normalized by removing the mean value and dividing by its l_2 norm.

4.2 Comprehensive Evaluation of RVFL Variants

For the simplicity of demonstration, the detailed results on the total 24 RVFL variants is summarized in Table B.1, Table B.2, Table B.3 and Table B.4. The overall performances of the variants based on their rank on each dataset are summarized in Table 4.2 .

Table 4.2 Average rank values ¹ based on classification accuracies of RVFL variants.

solution & activation function		-bias, -link ²	+bias, -link ³	-bias, +link ⁴	+bias, +link ⁵	<i>F</i> value ⁶
Ridge Regression	sigmoid	2.7975	2.7025	2.3306	2.1694	6.7829
	sine	2.5868	2.6033	2.4174	2.3926	0.8843
	hardlim	3.0785	3.0702	1.9091	1.9421	43.0537
	tribas	2.6901	2.6736	2.2810	2.3554	3.3333
	radbas	2.6612	2.6405	2.3182	2.3802	2.2775
	sign	3.0455	3.0372	1.9545	1.9628	36.7475
Moore-Penrose pseudoinverse	sigmoid	2.5868	2.5620	2.3926	2.4587	0.5639
	sine	2.5702	2.6612	2.2769	2.4917	1.9702
	hardlim	3.0785	3.0785	1.9732	1.9298	43.8826
	tribas	2.5620	2.6116	2.4008	2.4256	0.7646
	radbas	2.5000	2.7066	2.4008	2.3926	1.5576
	sign	3.0950	3.0289	1.9404	1.9256	40.6768

¹ Each row represents a distinct comparison. Lower value indicates higher accuracy.

² RVFL without bias and without direct link.

³ RVFL with bias and without direct link.

⁴ RVFL without bias and with direct link.

⁵ RVFL with bias and with direct link.

⁶ Statistic value derived from Eq. (3.14).

4.2.1 Importance of Bias and Direct Link

The effect of the direct link and bias in various RVFL configurations are firstly investigated. In order to have fair comparisons, other issues (the activation functions and the solution for the output weights) are fixed . In Table 4.2, each column represents an RVFL variant based on direct link and bias. Interestingly, we find that the bias is not a key factor for the performance. Most importantly, the direct link plays a much more important role than the bias term. It can be easily demonstrated that the performances in 5th – 6th columns are better

Chapter 4. Diverse Random Vector Functional Link Networks Ensemble

than those in $3^{rd} - 4^{th}$ columns. The direct link from the input layer to the output layer can serve as a regularization for the randomization thereby making the RVFL to achieve overall better performance than the RVFL variants without the direct link.

Table 4.3 Result of statistical test on RVFL variants.

(a) Ridge regression with “sigmoid”

	-b,-d	+b,-d	-b,+d	+b,+d
-b,-d			--	--
+b,-d				--
-b,+d	++			
+b,+d	++	++		

(b) Ridge regression with “hardlim”

	-b,-d	+b,-d	-b,+d	+b,+d
-b,-d			--	--
+b,-d			--	--
-b,+d	++	++		
+b,+d	++	++		

(c) Ridge regression with “sign”

	-b,-d	+b,-d	-b,+d	+b,+d
-b,-d			--	--
+b,-d				--
-b,+d	++			
+b,+d	++	++		

(d) M-P pseudoinverse with “hardlim”

	-b,-d	+b,-d	-b,+d	+b,+d
-b,-d			--	--
+b,-d				--
-b,+d	++			
+b,+d	++	++		

(e) M-P pseudoinverse with “sign”

	-b,-d	+b,-d	-b,+d	+b,+d
-b,-d			--	--
+b,-d				--
-b,+d	++			
+b,+d	++	++		

Statistically significant differences are marked with -- or ++. “+”, “-”, “b” and “d” have the same meaning as in Table 4.2. “--” means the method in the row is statistically significantly worse than the method in the column. “++” means the method in the row is statistically significantly better than the method in the column. “M-P” stands for Moore-Penrose.

Follow the same procedure in Section 3.4.3, the critical values for the F distribution with 3 and 360 degrees of freedom is 2.6 and CD for the Nemenyi test is 0.4264. Hence, for those rows with F value larger than 2.6, their difference is statistically significant. Further in those rows, the pair of columns whose ranks differ by more than 0.4264 are considered as statistically significantly different.

In Table 4.3, we summarize the statistical significance of the rank differences for those rows where the F value is larger than 2.6.

4.2.2 Effect of Activation Function

A detailed analysis of the effect from each activation functions is presented in this section. It is easy to get the critical value for F distribution with 5 and 600 degrees of freedom as 2.21 and the CD for the following Nemenyi test as 0.6196.

4.2 Comprehensive Evaluation of RVFL Variants

Table 4.4 Statistical significance test for different activation functions.

(a) -bias, -link, Ridge regression

	sigmoid (3.18)	sine (2.67)	hardlim (4.83)	tribas (2.95)	radbas (2.48)	sign (4.89)
sigmoid (3.18)			✓		✓	✓
sine (2.67)			✓			✓
hardlim (4.83)	✓	✓		✓	✓	
tribas (2.95)			✓			✓
radbas (2.48)	✓		✓			✓
sign (4.89)	✓	✓		✓	✓	

(b) +bias, -link, Ridge regression

	sigmoid (3.16)	sine (2.69)	hardlim (4.80)	tribas (2.98)	radbas (2.49)	sign (4.88)
sigmoid (3.16)			✓		✓	✓
sine (2.69)			✓			✓
hardlim (4.80)	✓	✓		✓	✓	
tribas (2.98)			✓			✓
radbas (2.49)	✓		✓			✓
sign (4.88)	✓	✓		✓	✓	

(c) -bias, +link, Ridge regression

	sigmoid (3.45)	sine (2.91)	hardlim (4.36)	tribas (2.98)	radbas (2.76)	sign (4.55)
sigmoid (3.45)			✓		✓	✓
sine (2.91)			✓			✓
hardlim (4.36)	✓	✓		✓	✓	
tribas (2.98)			✓			✓
radbas (2.76)	✓		✓			✓
sign (4.55)	✓	✓		✓	✓	

(d) +bias, +link, Ridge regression

	sigmoid (3.37)	sine (2.98)	hardlim (4.37)	tribas (3.02)	radbas (2.71)	sign (4.54)
sigmoid (3.37)			✓		✓	✓
sine (2.98)			✓			✓
hardlim (4.37)	✓	✓		✓	✓	
tribas (3.02)			✓			✓
radbas (2.71)	✓		✓			✓
sign (4.54)	✓	✓		✓	✓	

(e) -bias, -link, MP pseudoinverse

	sigmoid (2.79)	sine (2.82)	hardlim (4.68)	tribas (3.36)	radbas (2.60)	sign (4.75)
sigmoid (2.79)			✓			✓
sine (2.82)			✓			✓
hardlim (4.68)	✓	✓		✓	✓	
tribas (3.36)			✓		✓	✓
radbas (2.60)			✓	✓		✓
sign (4.75)	✓	✓		✓	✓	

(f) +bias, -link, MP pseudoinverse

	sigmoid (2.73)	sine (2.96)	hardlim (4.68)	tribas (3.34)	radbas (2.59)	sign (4.69)
sigmoid (2.73)			✓			✓
sine (2.96)			✓			✓
hardlim (4.68)	✓	✓		✓	✓	
tribas (3.34)			✓		✓	✓
radbas (2.59)	✓		✓			✓
sign (4.69)	✓	✓		✓	✓	

(g) -bias, +link, MP pseudoinverse

	sigmoid (3.10)	sine (2.95)	hardlim (4.19)	tribas (3.56)	radbas (2.93)	sign (4.27)
sigmoid (3.10)			✓			✓
sine (2.95)			✓			✓
hardlim (4.19)	✓	✓		✓	✓	
tribas (3.56)			✓			✓
radbas (2.93)			✓			✓
sign (4.27)	✓	✓		✓	✓	

(h) +bias, +link, MP pseudoinverse

	sigmoid (3.17)	sine (3.05)	hardlim (4.08)	tribas (3.58)	radbas (2.99)	sign (4.12)
sigmoid (3.17)			✓			✓
sine (3.05)			✓			✓
hardlim (4.08)	✓	✓		✓	✓	
tribas (3.58)			✓			✓
radbas (2.99)			✓			✓
sign (4.12)	✓	✓		✓	✓	

The values in bracket stands for their average rank. Lower values stands for better performance. The mark ✓ indicates that these two methods are statistically significantly different.

Chapter 4. Diverse Random Vector Functional Link Networks Ensemble

From Table 4.4, we can see that the *radbas* activation function always achieves better performance in all cases. *hardlim* and *sign* activation functions lead to the penultimate worst and the worst performances, respectively. For Ridge Regression based methods, there is a clear pattern

$$radbas > sine > tribas > sig > hardlim > sign \quad (4.1)$$

where “>” means that the method on the left performs better than the method on the right. However, for the Moore-Penrose pseudoinverse based methods, *tribas* always achieves the 4th rank. We also find that there is no clear superiority between *sigmoid* and *sine* activation functions.

4.2.3 Regularized Least Square Versus Moore-Penrose Pseudoinverse

Both the Moore-Penrose pseudoinverse and regularized least square (also named as ridge regression) can lead to closed-form solution for RVFL. In order to investigate the effect of these two methods on different RVFL variants, we keep the activation functions and the network structure (bias, direct link) to be the same and compare the performances. Hence, it leads to 24 pairs of comparisons in total. Sign test [114] is employed to test the statistical significance since each comparison only involves two classifiers.

If the two algorithms compared are equivalent as assumed under the null-hypothesis, each should win on approximately $N/2$ out of N datasets. The number of wins is distributed according to the binomial distribution. For a greater number of datasets, the number of wins is under the null-hypothesis distributed according to $N(N/2, \sqrt{N}/2)$, which allows for the use of z -test: if the number of wins is at least $N/2 + 1.96\sqrt{N}/2$ (or, for a quick rule of a thumb, $N/2 + \sqrt{N}$), the algorithm is significantly better with $p < 0.05$. In this case, if one algorithm wins more than 71.28 times on 121 datasets, then it is considered as statistically significantly better than the other one. These cases are highlighted in Table 4.5. Table 4.5 summarizes the results for each pair of comparisons. The entry in each column represents the number of times ridge regression (pseudoinverse) is better than pseudoinverse (ridge regression) for the same activation function. For example, the first column means ridge regression is better than Moore-Penrose pseudoinverse in 59 of 121 datasets and worse in 57 of 121 datasets. Generally, ridge regression leads to a better performance for almost all cases.

4.2 Comprehensive Evaluation of RVFL Variants

Table 4.5 Comparisons between ridge regression and Moore-Penrose pseudoinverse. The entry in each column represents the number of times ridge regression (pseudoinverse) is better than pseudoinverse (ridge regression) for the same activation function. Statistically significant columns are highlighted.

(a) -bias, -link,

	sigmoid	sine	hardlim	tribas	radbas	sign
Ridge Regression	59	61	80	78	63	80
Moore-Penrose pseudoinverse	57	56	36	38	54	37

(b) +bias, -link,

	sigmoid	sine	hardlim	tribas	radbas	sign
Ridge Regression	57	70	81	77	63	81
Moore-Penrose pseudoinverse	58	48	36	40	54	37

(c) -bias, +link,

	sigmoid	sine	hardlim	tribas	radbas	sign
Ridge Regression	64	57	68	76	58	63
Moore-Penrose pseudoinverse	52	56	46	42	59	51

(d) +bias, +link,

	sigmoid	sine	hardlim	tribas	radbas	sign
Ridge Regression	68	61	67	78	63	63
Moore-Penrose pseudoinverse	49	56	47	39	53	51

4.2.4 Overall Comparison

In this section, we present an overall comparison of the RVFL variants. Since we have already found that *hardlim* and *sign* activation functions consistently performed poorly, RVFL with these two activation functions will be excluded in this comparison. Hence, an overall comparison with 36 RVFL variants is presented in Table 4.6. In the same way, pairs of methods whose ranks differ by more than 4.5589 are statistically significantly different.

Chapter 4. Diverse Random Vector Functional Link Networks Ensemble

Table 4.6 Overall Comparison based on overall ranks of 32 RVFL variants ¹. Lower values in rank reflects better performance.

Method	RR, -b,-d,sigmoid,	RR, +b,-d,sigmoid,	RR, -b,+d,sigmoid,	RR, +b,+d,sigmoid
Rank	18.3182	18.0579	17.0000	16.6860
method	RR, -b,-d,sine,	RR,+b,-d,sine,	RR,-b,+d,sine,	RR,+b,+d,sine
Rank	15.1157	15.2603	14.5455	14.9421
method	RR, -b,-d,tribas,	RR,+b,-d,tribas,	RR,-b,+d,tribas,	RR,+b,+d,tribas
Rank	16.7810	16.7479	15.1446	15.5702
method	RR, -b,-d,radbas,	RR,+b,-d,radbas,	RR,-b,+d,radbas,	RR,+b,+d,radbas
Rank	13.9545	14.0083	13.2686	13.4256
Method	MP, -b,-d,sigmoid,	MP, +b,-d,sigmoid,	MP, -b,+d,sigmoid,	MP, +b,+d,sigmoid
Rank	17.0124	16.7149	16.5124	16.9298
method	MP, -b,-d,sine,	MP,+b,-d,sine,	MP,-b,+d,sine,	MP,+b,+d,sine
Rank	17.3223	18.2066	15.7107	16.3678
method	MP, -b,-d,tribas,	MP,+b,-d,tribas,	MP,-b,+d,tribas,	MP,+b,+d,tribas
Rank	20.8636	20.9876	19.7934	20.1612
method	MP, -b,-d,radbas,	MP,+b,-d,radbas,	MP,-b,+d,radbas,	MP,+b,+d,radbas
Rank	15.4504	16.5785	15.0496	15.5124

¹ RR and MP stand for ridge regression and Moore-Penrose pseudoinverse, respectively. “+”, “-”, “b” and “d” are the same meaning as in Table 4.2.

4.2.5 Range of the Random Parameters

In [93], the authors indicate that the performance of the RVFL may depend on the ranges of uniformly distributed random weights. However, this issue has been untouched in the literature to the best of the authors’ knowledge. In this work, we investigate this issue by introducing one scaling factor S to control the ranges of the randomization. This process can also shed light on the effect of saturation of hidden neurons in RVFL. Based on the performance in previous section, we choose ridge regression based RVFL with *radbas* activation function because it achieves the best performance among all variants.

In previously section, the random weights are generated with uniform distribution in $[-1,1]$, as done exactly in [87], while the biases are in $[0,1]$. In this section, the random weights and biases are generated with uniform distribution in $[-S, S]$ and $[0, S]$ respectively, where S is a positive scaling factor. In this work, we set $S = 2^t$, t is set to be $-5:0.5:5$. The

4.2 Comprehensive Evaluation of RVFL Variants

Figure 4.1 Performance of RVFL based for different ranges of randomization. Smaller rank indicates better accuracy and less number of hidden neurons. N stands for the number of hidden neurons corresponding to the testing accuracy used in the ranking.

performances 21 RVFL configurations with direct links and bias are summarized in Fig. 4.1. It is obvious that all RVFL variants perform poorly when the range of the random parameters becomes either too large or too small. Another interesting conclusion is the commonly adopted approach that $S = 1$ for the randomization may not lead to the optimal performance.

For RVFL without direct link, setting $t > 0$ for scaling factor S to increase the discrimination power of the features in the hidden neurons may make more neurons to saturate. This can be compensated by either having more hidden neurons or the direct link from the input layer to the output layer. On the other hand, setting $t < 0$ for scaling factor S to reduce the possibility of neuronal saturation may reduce the discrimination power of the features in the hidden neurons. Again, this can be compensated to some degree by having more hidden neurons or the direct link from the input layer to output layer. In Table 4.7 and Table 4.8, we also present rank values based on accuracies and number of hidden neurons of different RVFL variants for different scaling factor S with $t = -1.5 : 0.5 : 1.5$.

Chapter 4. Diverse Random Vector Functional Link Networks Ensemble

Table 4.7 Average rank values of RVFL based on accuracies for different scale factor values over 121 datasets. Each column stands for a distinct comparison. Lower rank means higher accuracy. All RVFLs use ridge regression solution and *radbas* activation function.

Method	$S = 2^{-1.5}$	$S = 2^{-1}$	$S = 2^{-0.5}$	$S = 1$	$S = 2^{0.5}$	$S = 2^1$	$S = 2^{1.5}$
-b,-d	2.6446	2.5992	2.5992	2.6612	2.7231	2.6322	2.6653
+b,-d	2.6529	2.5785	2.6446	2.6405	2.6736	2.6860	2.7355
-b,+d	2.4298	2.4752	2.3140	2.3182	2.3430	2.4215	2.3182
+b,+d	2.2727	2.3471	2.4421	2.3802	2.2603	2.2603	2.2810

Table 4.8 Average rank values of RVFL based on the number of hidden neurons for different scale factor values over 121 datasets. Each column stands for a distinct comparison. Lower rank means less number of hidden neurons. All RVFLs use ridge regression solution and *radbas* activation function.

Method	$S = 2^{-1.5}$	$S = 2^{-1}$	$S = 2^{-0.5}$	$S = 1$	$S = 2^{0.5}$	$S = 2^1$	$S = 2^{1.5}$
-b,-d	2.6818	2.6529	2.6322	2.5868	2.6653	2.6736	2.6942
+b,-d	2.6488	2.7025	2.5785	2.5785	2.6488	2.6570	2.6281
-b,+d	2.3760	2.3140	2.3760	2.4091	2.3636	2.3926	2.3099
+b,+d	2.2934	2.3306	2.4132	2.4256	2.3223	2.2769	2.3678

Results in Table 4.7 are consistent with the previous subsections which clearly indicate the advantage of the direct link. Moreover, RVFL with direct link achieves better accuracy with less number of hidden neurons. The direct link from input layer to output layer serves as a standing out regularization and make a high chance for RVFL to achieve a better performance with high possibility than those without direct link. That is, with smaller number of random hidden neurons, direct link in RVFL leads to a thinner and simpler model than those without. For a given set of observations or data, there is always an infinite number of possible hypotheses fit the same data. According to the Occam's Razor principle, one should choose from a set of otherwise equivalent models of a given phenomenon the simplest one. For example, it is possible for us to further increase the performance of RVFL if we enlarger the number of random hidden neurons. "Super flat" RVFL models (i.e. with a large number of hidden neurons) are more likely to overfit the available data. This is also in line with the statistical learning theory [115] that advocates for learning with lower complexity

models. Hence, the tuning range for the number of hidden neurons can be relatively narrower for RVFL with direct links.

4.3 RVFL Ensemble with Diverse Parameters

In this section, we evaluate the feasibility of ensemble learning to improve the performance of a single RVFL. Though any theoretical ensemble learning can be applied here, we adopt the simplest method in order to use the results from previous section. More specifically, following the pipeline in [6], we generate 500 basic RVFL models. We permute the parameter for each base model by adding a small noise based the tuned parameter obtained from previous section. We choose the RVFL with direct link and with bias as it shows satisfactory results. The details of those parameters are presented in Table 4.9.

Table 4.9 Parameters setting for base RVFL model.

Parameters	Tuned Result	Randomized Parameter
Number of Hidden Neurons	N	$Round((1 + \delta) * N)$
Regularization Parameter	λ	$(1 + \delta) * \lambda$
Scale Parameter	S	$(1 + \delta) * S$

The second column represents the best value for each parameter tuned from previous section. The third column stands for the parameter for each RVFL model. δ is a random variable drawn from $[0, 1]$ with uniform distribution.

4.3.1 Comprehensive Evaluation with Other 179 Methods on 121 Datasets

Detailed results of the RVFL ensemble is summarized in Table 4.10. According to section 4.2.3, if there are in total 2 methods and N datasets, one method which wins at least $N/2 + \sqrt{N}$ times are considered as statistically significantly better than the other. In this case, we can see RVFL ensemble wins 74 ($>60+11=71$) times. Hence we consider the proposed diverse RVFL ensemble is statistically significantly better than single RVFL model.

4.4 Conclusion

As an alternative to conventional back-prop based neural network, randomized neural network has been proposed a long time ago. In this chapter, we bring back the machine learning

Chapter 4. Diverse Random Vector Functional Link Networks Ensemble

community a very classical method—Random vector functional link network (RVFL). Extensive and comprehensive evaluation of variants of RVFL with closed-form solution is present in this chapter and basic conclusions can be summarized as:

1. The effect of the direct links from the input layer to the output layer. It turns out that the direct links lead to better performance than those without in all cases.
2. The effect of the bias in the output layer. It turns out that the bias term in the output neurons only has mixed effects on the performance, as it may or may not improve performance. Hence, bias can be a tunable network configuration depending on the specific problem.
3. Effect of scaling the randomization range of input weights and biases. We show scaling down the randomization range of input weights and biases to avoid saturating the neurons may risk at degenerating the discrimination power of the random features. However, this can be compensated by having more hidden neurons or direct link. Scaling the randomization range of input weights and biases up to enhance the discrimination power of the random features may risk saturating the neurons. Again, this can be compensated by having more hidden neurons or combining with the direct link from the input to the output layer. However, for reasons explained in Subsection 4.2.5, we prefer lower model complexity.
4. The performance of 6 commonly used activation functions summarized in Table 4.1. It turns out that *radbas* function always leads to a better performance. *hardlim* and *sign* activation functions lead to penultimate worst and worst performances, respectively.
5. The performance of Moore-Penrose pseudoinverse and ridge regression (or regularized Least Square) solutions for the output weights. It turns out that with one more parameter (λ in Eq. (2.4)) to tune, ridge regression based RVFL shows better performance than the Moore-Penrose pseudoinverse based RVFL.

We also propose a very simple RVFL ensemble, where each base model is trained with randomized parameters. Experimental results demonstrate the effectiveness of the proposed method.

Table 4.10 Details Results of RVFL ensemble.

Dataset	Ensemble	Single Model	Dataset	Single model	Ensemble
abalone	66.5	65.71	molec-biol-splICE	83.97	81.73
ac-inflam	100	98.33	monks-1	56.02	59.38
acute-nephritis	100	100	monks-2	66.2	66.67
adult	84.96	84.86	monks-3	45.14	51.68
annealing	50	34.25	mushroom	100	100
arrhythmia	70.13	68.14	musk-1	90.13	84.24
audiology-std	64	62	musk-2	96.42	95.51
balance-scale	91.83	91.27	nursery	93.53	92.85
balloons	93.75	87.5	oocMerl2F	85.29	85.61
bank	89.76	89.89	oocMerl4D	93.82	93.43
blood	76.87	76.64	oocTris2F	86.29	84.59
breast-cancer	73.24	74.65	oocTris5B	94.74	95.07
bc-wisc	97.29	96.71	optical	97.22	95.2
bc-wisc-diag	97.01	96.13	ozone	97.08	96.92
bc-wisc-prog	81.12	79.21	page-blocks	95.54	95.56
breast-tissue	70.19	63.94	parkinsons	88.27	89.54
car	94.39	93.11	pendigits	96.37	95.59
ctg-10classes	81.54	79.66	pima	76.43	76.73
ctg-3classes	91.43	90.7	pb-MATERIAL	92.31	92.31
chess-krvk	39.03	37.02	pb-REL-L	72.12	77.88
chess-krvkp	98.25	95.47	pb-SPAN	69.57	71.74
congress-voting	61.7	62.44	pb-T-OR-D	89	87
conn-bench-sonar	84.62	81.25	pb-TYPE	74.04	68.27
conn-bench-vowel	99.57	97.94	planning	71.67	71.81
connect-4	76.3	75.95	plant-margin	82.31	77.14
contrac	54.08	53.84	plant-shape	64.06	57.77
credit-approval	86.34	87.28	plant-texture	83.75	79.45
cylinder-bands	77.54	76.46	post-operative	72.73	72.44
dermatology	97.53	97.18	primary-tumor	55.79	56.1
echocardiogram	85.61	85.98	ringnorm	96.66	95.5
ecoli	87.8	87.95	seeds	95.67	95.43
energy-y1	94.01	93.55	semeion	89.95	88.51
energy-y2	90.76	90.36	soybean	91.56	50.2
fertility	87	86.75	spambase	91.7	90.47
flags	51.04	50	spect	61.83	58.74
glass	67.45	68.99	spectf	91.98	84.22
haberman-survival	73.36	75.16	st-aus-credit	65.41	67.11
hayes-roth	85.71	83.04	st-german-credit	78.2	76.83
heart-cleveland	60.53	61.43	st-heart	88.06	87.31
heart-hungarian	85.27	84.08	st-image	94.84	94.2
heart-switzerland	49.19	43.95	st-landsat	86.4	85.3
heart-va	34	33.75	st-shuttle	97.95	97.02
hepatitis	85.26	79.97	st-vehicle	82.58	83.06
hill-valley	49.34	49.34	steel-plates	76.6	74.9
horse-colic	79.41	69.49	synthetic-control	98.33	97.08
ilpd-indian-liver	72.6	73.89	teaching	55.92	54.61
image-segmentation	27.95	26.98	thyroid	93.96	84.4
ionosphere	94.89	91.48	tic-tac-toe	97.59	98.04
iris	98.65	98.65	titanic	77.77	78.23
led-display	74.8	75	rains	87.5	78.13
lenses	83.33	83.33	twonorm	97.88	97.89
letter	82.43	79.59	vc-2classes	85.39	85.31
libras	85.56	81.32	vc-3classes	83.77	83.77
low-res-spect	91.35	91.35	wall-following	86.62	82.74
lung-cancer	56.25	64.84	waveform	86.64	86.65
lymphography	85.81	85.98	waveform-noise	86.96	86.03
magic	85.24	84.96	wine	96.59	100
mammographic	81.56	82.08	w-qua-red	60.88	59.56
miniboone	90.58	90.26	w-qua-white	56.17	55.12
molec-biol-promoter	88.46	86.54	yeast	61.12	60.26
			zoo	98	95.5

Co-trained Kernel Ridge Regression Ensemble

A typical linear regression can be formulated as the following problem:

$$\min_w \sum_i (x_i w - y_i)^2 + \lambda \|w\|^2; \quad (5.1)$$

The λ is a user-defined regularization parameter that controls the model complexity. This problem can lead to an elegant closed-form solution:

$$w = (X^T X + \lambda I) X^T Y; \quad (5.2)$$

where the data matrix X has one sample per row x_i , and each element of the vector Y is the output target y_i for x_i . I is an identity matrix.

Kernel ridge regression (KRR) extend the linear regression nonlinear and high dimension through the kernel trick. Based on the Representer Theorem, the solution of w can be formulated as a linear combination of the samples in the feature space $\phi(x)$ as: $w = \sum_i \alpha_i \phi(x_i)$. The problem is then formulated as:

$$\min_{\alpha} \|Y - K\alpha\|^2 + \lambda \alpha^T K \alpha; \quad (5.3)$$

Similarly, the solution is given in a closed-form manner as:

$$\alpha = (K + \lambda I)^{-1} Y; \quad (5.4)$$

By applying the kernel trick, the kernel matrix K can be obtained by $K_{ij} = k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$. One can simply employ the commonly used kernel functions such as Gaussian kernel, RBF kernel, linear kernel without explicitly knowing the form the feature transformation function $\phi(x)$.

Classification can be posed as regression with class labels. One alternative is to define the output target Y with 0 – 1 coding [116]. More specifically, if there are K classes with n samples, the output Y should be an $n \times K$ matrix which can be generated by the following equation:

$$Y_{ij} = \begin{cases} 1 & \text{If } i^{th} \text{ sample belongs to the } j^{th} \text{ class} \\ 0 & \text{Otherwise} \end{cases} \quad (5.5)$$

5.1 Co-trained KRR

The kernel matrix in Eqn. (5.5) plays a central role in KRR. Actually different kernels may correspond to different notions of similarity, and using a specific kernel may lead to a risk of bias and result in the classification result. One may use an ensemble of kernels to tackle this problem. However, the time complexity of problem is around $O(n^3)$, where n is the number of data samples. When n is large, too many kernels may become computational prohibitive. In this work, we propose to train an ensemble of KRR jointly. We show with 2 kernels, the proposed method, which is named as Co-KRR, performs competitively against the state-of-the-art Random Forest Classifier.

More specifically, suppose that we have 2 kernels, then the proposed method aims at solving the following problem:

$$\begin{aligned} T(\alpha) = \min_{\alpha} \sum_{i=1}^2 (\|Y - K_i \alpha_i\|^2 + \lambda \alpha_i^T K_i \alpha_i) \\ + \beta \sum_{i,j=1}^2 \|K_i \alpha_i - K_j \alpha_j\|^2 \end{aligned} \quad (5.6)$$

The first term in Eqn (5.6) forces each individual model to have minimal squared error with respect to the desired output y . The second term $(\alpha_i^T K_i \alpha_i)$ denotes the Tikhonov regularization in a Reproducing Kernel Hilbert Space. The third term regularizes each model in a pair-wise fashion which weights the influence of pairwise disagreements. It allows high performance to be achieved since it becomes unlikely for compatible classifiers trained on independent kernels/features to agree on an incorrect label. When $\beta = 0$, the proposed method is equivalent to 2 independent kernel ridge regression problems. The proposed

Chapter 5. Co-trained Kernel Ridge Regression Ensemble

method can also be regarded as a purely supervised version of the co-regularized multiview learning [117].

It is straight-forward to obtain the solution of Eqn (5.6) by setting the derivation of T with respect to α_i to be zero:

Let

$$G_i = (2\beta + 1)K_i K_i + \lambda K_i, \quad (5.7)$$

we can obtain the solution for Eqn (5.6)

$$\begin{aligned} \alpha_1 &= (G_1 - 4\beta^2 K_1 K_2 G_2^{-1} K_2 K_1)^{-1} \\ &\quad * (K_1 Y + 2\beta K_1 K_2 G_2^{-1} K_2 Y), \\ \alpha_2 &= (G_2 - 4\beta^2 K_2 K_1 G_1^{-1} K_1 K_2)^{-1} \\ &\quad * (K_2 Y + 2\beta K_2 K_1 G_1^{-1} K_1 Y), \end{aligned} \quad (5.8)$$

Let $v = 2\beta + 1$ and notice

$$G_i^{-1} = (\beta K_i + \lambda I)^{-1} * K_i^{-1}, \quad (5.9)$$

Substituting Eqn (5.7) into Eqn (5.8), we have

$$\begin{aligned} \alpha_1 &= ((vK_1 + \lambda I) - 4\beta^2 K_2 (vK_2 + \lambda I)^{-1} K_1)^{-1} * (Y + 2\beta K_2 (vK_2 + \lambda I)^{-1} Y), \\ \alpha_2 &= ((vK_2 + \lambda I) - 4\beta^2 K_1 (vK_1 + \lambda I)^{-1} K_2)^{-1} * (Y + 2\beta K_1 (vK_1 + \lambda I)^{-1} Y), \end{aligned} \quad (5.10)$$

5.2 Comprehensive Evaluation with Other 179 Methods on 121 Datasets

We follow the pipeline in previous chapter to conduct comprehensive evaluations of the KRR and the proposed Co-KRR. The regularization parameter λ in Eqn (5.1) is set to be 2^C and C is set to be -14:5. We use Gaussian kernel for all cases and the kernel parameter is set to be 2^g and g is set to be -16:8. For the Co-KRR, we firstly generate one extra feature set by a linear transformation of the original features. More specifically, we generate a random projection matrix W , where each entry is sampled from $N(0, 1)$ normal distribution. The size of W is $m \times 2m$, where m is the dimensionality of the original feature set. We tune these two base models separately within the same pipeline. After that, we fix these parameter in Eqn (5.6) and further tune the parameter β . The range of β is set to be exactly the same as λ .

The detailed results are summarized in Table 5.1.

5.2 Comprehensive Evaluation with Other 179 Methods on 121 Datasets

Table 5.1 Detailed results of Co-KRR and other KRR variants.

Data	Co-KRR-avg	Co-KRR-max	2KRR-avg	2KRR-max	KRR-O	KRR-T	KRR-C
abalone	66.67	66.76	66.55	66.5	66.64	66.16	66.09
ac-inflam	100	100	100	100	100	100	100
acute-nephritis	100	100	100	100	100	100	100
adult	85.09	85.09	85.12	85.12	84.74	85.11	85.19
annealing	48	48	57	47	57	55	51
arrhythmia	72.12	71.9	100	100	70.13	100	100
audiology-std	64	72	64	72	68	76	64
balance-scale	91.35	91.35	91.35	91.35	91.51	91.35	91.51
balloons	68.75	68.75	81.25	75	87.5	68.75	75
bank	89.93	89.93	89.98	89.98	89.38	89.87	89.89
blood	78.74	78.74	78.74	79.01	78.61	80.21	78.61
breast-cancer	74.65	74.65	73.94	73.94	73.94	73.59	73.94
bc-wisc	97.57	97.57	97.43	97.43	97.57	97.14	97.43
bc-wisc-diag	97.18	97.18	96.83	96.83	97.54	96.83	97.01
bc-wisc-prog	82.65	82.65	84.18	84.18	79.59	81.12	81.63
breast-tissue	69.23	67.31	68.27	66.35	70.19	65.38	66.35
car	97.8	98.03	98.38	98.15	98.44	97.8	99.13
ctg-10classes	84.32	84.27	84.18	83.85	84.65	83.8	84.27
ctg-3classes	92.94	93.13	93.13	93.36	92.94	91.62	92.84
chess-krvk	77.79	78.18	45.07	44.12	61.18	42.02	47.44
chess-krvkp	99.22	99.22	99.16	99.16	99.25	99.16	99
congress-voting	61.7	61.7	61.7	61.7	61.24	61.47	61.47
conn-bench-sonar	87.02	87.02	87.02	87.02	84.62	87.5	87.5
conn-bench-vowel	100	100	100	100	100	100	100
connect-4	78.6	78.6	82.69	82.69	84.9	75.58	86.45
contrac	54.48	55.23	54.14	54.62	53.26	53.74	53.74
credit-approval	86.34	86.34	86.05	86.05	86.48	85.9	86.48
cylinder-bands	80.08	80.08	79.3	79.1	78.32	77.73	77.54
dermatology	97.8	97.53	97.53	97.53	97.53	97.53	97.53
echocardiogram	85.61	85.61	85.61	85.61	85.61	85.61	84.85
ecoli	87.2	87.5	86.31	86.9	87.5	86.9	87.8
energy-y1	94.27	94.27	94.66	94.66	94.01	92.58	91.93
energy-y2	90.49	90.63	90.76	90.63	90.23	90.1	90.76
fertility	88	88	88	88	86	87	89
flags	50	50	49.48	47.92	54.17	45.83	54.17
glass	65.57	63.68	66.51	68.4	69.34	64.62	62.74
haberman-survival	74.01	74.01	75	75	72.37	72.37	73.03
hayes-roth	78.57	78.57	82.14	82.14	85.71	75	78.57
heart-cleveland	59.87	59.87	60.53	60.86	60.2	60.86	60.53
heart-hungarian	86.64	86.64	85.62	86.3	85.27	85.27	85.96
heart-switzerland	33.06	33.87	33.06	36.29	34.68	37.9	41.13
heart-va	29.5	31	26.5	27.5	33.5	26.5	29
hepatitis	84.62	84.62	83.33	84.62	83.97	83.33	83.97
hill-valley	60.4	60.23	62.87	60.73	50.83	60.89	59.57
horse-colic	76.47	76.47	76.47	76.47	58.82	75	82.35
ilpd-indian-liver	73.29	73.29	72.95	72.95	73.12	73.46	72.95
image-segmentation	29.19	29.14	29.1	29.14	29.14	28.67	27.76
ionosphere	92.33	92.61	93.18	93.18	92.61	92.33	94.6
iris	97.97	97.97	97.97	97.97	97.3	97.3	98.65
led-display	75.2	74.6	74.1	74.1	74.9	73.2	75
lenses	87.5	87.5	87.5	87.5	87.5	79.17	79.17
letter	97.35	97.31	97.25	97.32	97.73	96.79	97.28
libras	85.83	85.83	86.11	85.83	85.28	85.56	86.11
low-res-spect	90.04	90.23	90.04	90.79	90.41	90.23	89.85
lung-cancer	56.25	56.25	50	53.13	56.25	53.13	50
lymphography	85.14	85.14	85.81	85.14	87.84	85.14	79.73

Continued on next page

Chapter 5. Co-trained Kernel Ridge Regression Ensemble

Table 5.1 – continued from previous page

Data	Co-KRR-avg	Co-KRR-max	2KRR-avg	2KRR-max	KRR-O	KRR-T	KRR-C
magic	86.88	86.88	84.37	84.4	86.02	83.79	84.84
mammographic	82.19	82.19	82.5	82.5	82.08	81.35	81.15
miniboone	89.4	89.4	89.29	89.29	92.83	89.38	91.88
molec-biol-promoter	83.65	83.65	77.88	81.73	78.85	77.88	85.58
molec-biol-splice	86.2	86.45	86.73	86.57	85.88	83.88	86.01
monks-1	66.44	66.44	65.97	66.44	64.35	66.2	68.29
monks-2	64.81	64.81	64.58	65.51	65.28	64.12	65.74
monks-3	42.59	42.59	42.36	42.59	57.64	45.6	57.41
mushroom	100	100	100	100	100	100	100
musk-1	90.55	90.55	90.55	90.34	90.55	89.5	89.08
musk-2	99.68	99.68	99.7	99.7	99.68	99.68	99.68
nursery	99.48	99.48	99.45	99.44	99.47	99.21	99.39
oocMerl2F	85	85	85	85	85.39	84.61	85.29
oocMerl4D	94.41	94.41	94.41	94.31	94.12	94.02	94.41
oocTris2F	86.62	86.62	87.17	87.17	85.42	87.06	86.84
oocTris5B	94.63	94.52	94.3	94.41	94.3	94.08	93.86
optical	98.39	98.33	98.33	98.33	98.5	98.16	98.39
ozone	97	97	97.08	97.08	97.16	97.08	97.08
page-blocks	96.44	96.42	96.36	96.42	96.49	96.36	96.33
parkinsons	94.39	94.39	94.39	94.39	94.39	92.86	91.84
pendigits	97.77	97.77	97.88	97.77	97.54	97.68	97.63
pima	76.3	76.3	76.43	76.3	76.04	76.17	76.43
pb-MATERIAL	89.42	89.42	90.38	88.46	91.35	88.46	92.31
pb-REL-L	72.12	73.08	73.08	73.08	72.12	73.08	73.08
pb-SPAN	71.74	71.74	75	75	69.57	73.91	72.83
pb-T-OR-D	88	88	88	88	88	87	87
pb-TYPE	67.31	67.31	64.42	65.38	67.31	65.38	67.31
planning	72.22	72.22	72.22	72.22	71.11	72.22	72.22
plant-margin	84.75	84.81	83.69	84	84.75	82.56	83.38
plant-shape	72.13	72.19	72.38	72.31	72.06	72.31	72.13
plant-texture	86	86	86.56	86.38	86.56	85.69	85.13
post-operative	71.59	71.59	71.59	71.59	72.73	71.59	71.59
primary-tumor	55.79	56.1	53.96	53.05	56.71	50.91	56.71
ringnorm	98.2	98.2	98.09	98.09	98.45	97.58	97.89
seeds	95.67	96.15	95.67	95.19	96.15	95.67	95.67
semeion	96.17	96.17	96.36	96.17	91.58	96.17	96.11
soybean	92.86	92.53	92.86	92.86	61.97	93.18	92.86
spambase	93.24	93.24	93.02	93	92.26	92.89	93.7
spect	62.37	62.37	66.13	66.67	54.84	67.2	69.35
spectf	91.98	91.98	91.98	91.98	91.98	91.98	91.98
st-aus-credit	65.84	65.84	64.97	64.97	66.28	64.97	67.44
st-german-credit	77.7	77.7	77.8	77.8	76.7	77.4	77.4
st-heart	88.06	88.06	88.06	88.06	88.43	87.69	86.57
st-image	97.4	97.31	97.18	97.14	93.63	96.58	96.92
st-landsat	91.75	91.5	91.75	91.7	92.05	91.25	91.6
st-shuttle	99.89	99.89	99.89	99.89	99.9	99.89	99.89
st-vehicle	84.12	84.24	84.36	84.6	77.13	83.89	84.72
steel-plates	76.24	76.34	76.75	76.65	76.8	76.39	76.6
synthetic-control	98.67	98.83	99.17	99	99.17	99.17	99.17
teaching	53.29	54.61	52.63	56.58	61.84	57.24	63.16
thyroid	95.13	94.81	95.07	94.95	96.44	94.37	94.69
tic-tac-toe	100	100	97.91	100	100	97.91	99.9
titanic	78.95	78.95	78.95	78.95	78.95	78.95	78.95
rains	87.5	87.5	75	87.5	87.5	75	87.5
twonorm	97.89	97.89	98.03	98.03	97.91	97.93	97.93
vc-2classes	86.36	86.36	86.36	86.36	86.04	86.69	86.04
vc-3classes	81.82	81.82	83.44	83.12	84.09	81.82	81.17

Continued on next page

Table 5.1 – continued from previous page

Data	Co-KRR-avg	Co-KRR-max	2KRR-avg	2KRR-max	KRR-O	KRR-T	KRR-C
wall-following	92.12	92.21	92.06	91.99	92.16	91.51	92.08
waveform	86.68	86.78	86.42	86.52	86.86	86.16	86.5
waveform-noise	86.8	86.86	86.84	86.34	86.82	85.96	86.72
wine	99.43	99.43	100	100	97.73	98.3	100
w-quar-red	62.06	62.44	62.06	62.44	62	62.06	61
w-quar-white	65.3	65.22	65.38	65.67	65.09	65.34	65.05
yeast	60.51	59.84	60.18	59.64	60.38	59.5	60.38
zoo	99	100	98	96	96	98	99

The output of two models can be obtained by either averaging the results or choosing the one with the maximum response. In Table 5.1, the second and third column denote the average result or the one with maximum response of the Co-KRR, respectively. The fourth column, denoted as “2KRR-avg”, represents the average results of the 2 KRR. This can be regarded as setting the parameter β in Eqn (5.6) to be 0. In this case, 2 kRR models are trained independently with its own features. Similarly, the “2KRR-max” means the max result of the 2 KRR models. The last three column represent the single KRR model. “KRR-O”, “KRR-T” stand for the single KRR model trained on original features and transformed features, respectively. In order to show the advantage of the proposed method, we also train a KRR model on the concatenation of the two feature sets, which is named as “KRR-C” in Table 5.1.

We first summarize the Friedman rank of each method in Table 5.2. We can see both for Co-KRR and 2KRR methods, selecting the one with maximum response lead to better performance. The Co-KRR methods all outperform the 2KRR methods, which indicates the effectiveness of the proposed pairwise constraints in Eqn (5.6). Moreover, using the same statistical test we found the KRR-T is statistically significantly worse than all other methods.

Table 5.2 Friedman rank of different KRR methods.

Method	Co-KRR-avg	Co-KRR-max	2KRR-avg	2KRR-max	KRR-O	KRR-T	KRR-C
Rank	3.6942	3.6364	3.7893	3.7479	3.8554	5.2025	4.0744

5.3 Conclusion

Kridge ridge regression (KRR) admits a simple closed-form solution and can achieve performance which is close to more sophisticated methods such as SVM. In this chapter, we propose a method which can train 2 KRR models jointly and simultaneously. These 2 models can be trained with different kernels, different parameters and even different features. We conduct extensive experiments on the same 121 datasets. Experimental results show that the proposed method outperforms the single KRR model.

Summary of Part I

With the concept of “perturb and combine”, ensemble methods are shown to improve the performance of a single predictor and have been applied in many fields such as machine learning, computer vision, speech recognition and so on. The rationale of ensemble methodology is to build a predictive model (either classifier or regressor) by integrating multiple models.

In the first part of thesis, we work on ensemble methods for general pattern classification methods. In Chapter 2, we give literature review on ensemble learning methods. In chapter 3, we propose an efficient oblique decision tree forest. This work is based on the MPSVM which can directly capture the geometric structure of the data in each internal node of the decision tree. In Chapter 4, we bring back the machine learning community a very classical method—Random Vector Functional Link Networks (RVFL). Without using conventional back-prop method, RVFL can be optimized in a single learning step. Moreover, we further propose a diverse RVFL ensemble to improve the performance of RVFL. In Chapter 5, we propose a Co-trained Kernel ridge regression (KRR) methods which can train 2 KRR models jointly. In each chapter, we conduct a comprehensive evaluation of the proposed method on 121 UCI datasets. This is the most exhaustive evaluation existing in the literature regarding the number of datasets. The experimental results demonstrate the effectiveness of the proposed methods.

6.1 Overall Comparison of the Methods in Part I

In this section, we compare the performance of MPRaF-P, MPRRoF-P, RVFL ensemble, Co-KRR-max, together with the 179 classifiers in [6]. Thus, in total there are 183 classifiers and 121 datasets. For clarity, we summarize the overall top 20 methods in Table 6.1.

6.1 Overall Comparison of the Methods in Part i

Table 6.1 Top 20 out of 183 methods.

Method	Friedman Rank	Mean Accuracy
MPRaF-P	33.10	82.05
CoKRR-max	34.13	81.79
MPRRoF-P	34.56	82.35
parRF_t	34.97	81.96
rf_t	35.16	82.30
svm_C	36.78	81.77
svmPoly_t	40.36	80.31
elm_kernel_m	41.65	81.52
rforest_R	41.67	81.94
svmRadialCost_t	42.47	80.54
RVFL ensemble	44.27	80.94
svmRadial_t	44.78	80.17
C5.0_t	45.22	80.56
avNNet_t	46.54	79.37
nnet_t	47.99	79.52
BG_LibSVM_w	49.44	80.77
pcaNNet_t	49.61	78.65
mlp_t	49.82	80.25
RotationForest_w	50.20	80.61
RRF_t	52.64	80.92

We follow the name style of [6] and all the results of other methods are copied from it.

In Table 6.1, the Friedman ranking for the best three classifiers are 33.10, 34.13, 34.56 respectively, being MPRaF-P slightly better and MPRRoF-P slightly worse. The Co-trained KRR method proposed in Chapter 5 achieves the second best performance. Therefore, although the *No – Free – lunch* theorem indicates that no classifier can be always the best, in practice, our proposed methods can be very near to achieving the best performance for almost all the data sets. The proposed RVFL method in Chapter 4, with Friedman rank of 44.27, achieves the 11th ranking. It outperforms all other neural network methods, which indicates the feasibility of the randomized neural network.

According to [112], sign test [114] is employed to test the statistical significance for each pair of classifiers for the top 20 methods. If the two algorithms compared are equivalent as assumed under the null-hypothesis, each should win on approximately $N/2$ out of N datasets. The number of wins is distributed according to the binomial distribution. For a greater number of datasets, the number of wins is under the null-hypothesis distributed according to $N(N/2, \sqrt{N}/2)$, which allows for the use of z -test: if the number of wins is at least $N/2 + 1.96\sqrt{N}/2$ (or, for a quick rule of a thumb, $N/2 + \sqrt{N}$), the algorithm is significantly better with $p < 0.05$. In this case, if one algorithm wins more than 71.28 times on 121 datasets, then it is considered as statistically significantly better than the other one. The result of the statistical result is presented in Table 6.2. The empty entry indicates that no statistical significance is observed. The + means the method in the row is statistically significant better than the method in the column. Similarly, – indicates the method in the row is statistically significant worse than the method in the column. We can see among the 6

Chapter 6. Summary of Part i

top classifiers, no statistical significance is observed. MPRAF-P is statistically significant better than the method in the range of 11-20. The results for CoKRR-max and MPRRoF-P are almost the same.

Table 6.2 Statistical comparisons of the top 20 methods.

	MPRAF-P	CoKRR-max	MPRRoF-P	parRF_t	rf_t	svm_C	svmPoly_t	elm_kernel_m	rforest_R	svmRadialCost_t	RVFL ensemble	svmRadial_t	C5.0_t	avNNet_t	nnet_t	BG_LibSVM_w	pcaNNet_t	mlp_t	RotationForest_w	RRF_t
MPRAF-P							+		+		+	+	+	+	+	+	+	+	+	+
CoKRR-max							+		+	+	+	+	+	+	+	+	+	+	+	+
MPRRoF-P											+	+	+	+	+	+	+	+	+	+
parRF_t													+		+		+	+	+	+
rf_t													+		+		+	+	+	+
svm_C											+				+		+	+	+	+
svmPoly_t	-	-														+	+	+	+	+
elm_kernel_m															+		+	+	+	+
rforest_R	-	-															+	+	+	+
svmRadialCost_t		-															+	+	+	+
RVFL ensemble	-	-	-			-											+			+
svmRadial_t	-	-	-																+	
C5.0_t	-	-	-	-	-															
avNNet_t	-	-	-																	
nnet_t	-	-	-	-	-			-												
BG_LibSVM_w	-	-	-			-														
pcaNNet_t	-	-	-	-	-		-	-		-	-									
mlp_t	-	-	-	-	-	-		-	-											
RotationForest_w	-	-	-	-	-	-	-	-	-	-	-	-								
RRF_t	-	-	-	-	-	-		-	-	-	-									

Part II

Ensemble Tracking

Literature review for Visual Tracking

In this chapter we first give a brief introduction to the background of visual tracking, followed by a comprehensive review of the state-of-the-art visual tracking methods.

7.1 Visual Tracking

As a fundamental task in computer vision, visual tracking has received a fast-growing interest due to applications such as video surveillance, human computer interaction, medical imaging and so on. As demonstrated in Fig 7.1, despite significant progress made recently, it remains challenging due to factors such as lighting and background variations, occlusions, motion blur and so on. In Fig 7.1, the first row shows that the object being tracked may be heavily occluded. In the second row, there exists motion blur in the video sequence. In the last row, the illumination may change significantly. Although visual tracking can be formulated in different settings according to different applications, most researchers focus on the *one – pass model – free single object tracking* setting, which is the most fundamental case. More specifically, without any appearance model, it only provides the bounding box of one single object in the first frame. Given this single (labeled) instance, the goal is to predict the location of the object in an online manner.

Modern trackers consist of 2 sub-models. One is motion model and the other is the appearance model. The motion model aims at estimating the posteriori probability of the target states. For the motion model, particle filter has long been regarded as the dominant approach. On the statistical side, particle filters are regarded as sequential Monte Carlo importance sampling methods for estimating the latent state of the dynamical systems. If we denote by s^t the latent state and y^t the observation variables in t^{th} frame, visual tracking aims

Figure 7.1 Challenging scenarios in visual tracking.

to find the most likely state in the t^{th} frame of the video based on the observations up to the $(t - 1)^{th}$ frame according to the following relationship:

$$\begin{aligned} s^t &= \argmax p(s^t | y^{1:t-1}) \\ &= \argmax \int p(s^t | s^{t-1}) p(s^{t-1} | y^{1:t-1}) ds^{t-1}. \end{aligned} \quad (7.1)$$

When y^t is available at t^{th} frame, the tracker updates the posterior distribution of the state variable in a Bayesian manner,

$$p(s^t | y^{1:t}) = \frac{p(y^t | s^t) p(s^t | y^{1:t-1})}{p(y^t | y^{1:t-1})}. \quad (7.2)$$

Particle filter estimates the true posterior state distribution $p(s^t | y^{1:t})$ by a set of n particles, $\{s_i^t\}_{i=1}^n$ with corresponding normalized weights $\{W_i^t\}_{i=1}^n$. The weight are updated as follows:

$$W_i^t = W_i^{t-1} * \frac{p(y^t | s_i^t) p(s_i^t | p(s_i^{t-1}))}{q(s^t | s^{1:t-1}, y^{1:t})}. \quad (7.3)$$

For simplicity, in practice the importance distribution is set to be a first-order Markov process thereby the weight are simply updated as

$$W_i^t = W_i^{t-1} * p(y^t | s_i^t) \quad (7.4)$$

Note after each update, resampling is needed when the sum of new weights is too small and normalization is needed when the sum of weights is too large. Recent work in [118] shows

Chapter 7. Literature review for Visual Tracking

that there does not exist significant differences among various motion models such as particle filter, sliding window and radius sliding window.

Appearance model plays the central role in modern tracking systems. Based on appearance model, most trackers can be divided into generative tracker [119–125, 125, 126] or discriminative tracker [127–133] or hybrid of them [134]. Generative trackers formulate the tracking problem as searching for the regions most similar to the target model. The rationale behind generative models for tracking is based on the reasonable assumption that the targets being tracked can be described by some generative processes. These methods are either based on template matching [121–125] or subspace learning [119, 120] or learning more powerful descriptors for the image patch [126]. The appearance model also needs to be updated dynamically to adapt to the target appearance variations. A template update method was proposed in [125] which could reduce the drifting problem by aligning with the first template to reduce drifts. Incremental subspace learning was proposed in [120]. In order to cover a wide range of pose and illumination variations, appearance model in [123] is decomposed into several basic appearance models. Recently, sparse representation has been proposed in visual tracking [135–138]. It works by solving the l_1 norm based optimization and the candidates with the lowest reconstruction error is selected as the tracking result. Sparse prototype was proposed in [139] which exploits classic principal component analysis (PCA) algorithm with recent sparse representation schemes for learning effective appearance models.

Discriminative trackers solve tracking problem by an online classification problem which distinguishes the target from the background. Discriminative trackers utilize the information from both the object and the background. Classical example include ensemble tracking [127], where each base tracker is a least-square classifier which works directly on the image pixels. Online boosting was proposed in [128] to online update discriminative features. Extensions of online boosting to semi-supervised versions were addressed in [129] to prevent drift. Multiple instance learning [130], which puts all ambiguous positive and negative samples into bags to learn a discriminative model for tracking, aims at alleviating the “butterfly effect” caused by inaccurate update of the classifier. In [132], online discriminative feature selection was proposed for visual tracking. It optimized the objective function in the steepest ascent direction with respect to the positive samples while in the steepest descent direction with respect to the negative ones. The concept of “superpixel” was proposed in [140] which facilitates the tracker to distinguish between the target and background. Partial least square was employed in [133] to learn a set of appearance models for adaptive discriminative object representation. Self-paced learning and online solvers for SVMs were proposed in [141] for visual tracking. In [142], a simple convolutional neural network based tracker was proposed

and updated with backpropagation to extracted discriminant features to distinguish the object from background. Deep learning and transfer learning based tracking was proposed in [143, 144] where the author pre-trained a deep neural network with a huge amount of labeled images and finetune in the process of tracking. Similar idea can be found in [145] where visual prior from generic real-world images can be learned and transferred for representing objects in a scene. Deep learning without pre-training can be found in [146].

According to the comprehensive results on recent surveys [147], discriminative methods show their superiority over generative models. Hence, in this thesis, we mainly focus on discriminative tracker based on the concepts proposed in the first part.

7.2 Performance Evaluation of Visual Tracking

A wealth of performance measures have been proposed for single-object model free target tracking, among which the most commonly used are based on center error [120] and region overlap [148]. The mean center error measures the average distance (in pixel) from the center of the tracking result and the ground for each frame. The overlap score measures the average overlap between the ground truth bounding box and the tracking result. Given the ground truth bounding box r_a , once we got the tracking result, i.e., the tracked bounding box r_t , the overlap score can be evaluated by: $S = \frac{|r_a \cap r_t|}{|r_a \cup r_t|}$, where $\cap$ and $\cup$ represent the intersection and union of two regions, respectively, and $|\cdot|$ stands for the number of pixels in the region.

Although widely used, those performance criteria have their shortcomings. The average location error may impose an arbitrary penalty on lost trackers which depend on chance factors because the tracking result could be uncertain once it lose the target. On the other hand, the bounding box overlap as the disadvantage of heavily penalizing trackers that do not track across scale, even if the target position is otherwise tracked perfectly.

Following the work in [7], we choose the increasingly popular criterion—precision curve to evaluate the trackers in this thesis. A frame is regarded as correctly tracked if the center of the tracking result is within a distance threshold of ground truth. Precision curves simply represents the percentage of correctly tracked frames for a range of distance thresholds. No parameters are required to plot the precision curve, which makes the curves unambiguous and easy to interpret. On one hand, a higher precision score at low thresholds indicates the tracker is more accurate. On the other hand, a lost target will prevent it from achieving perfect precision for a very large threshold range. When a representative precision score is needed, the chosen threshold is 20 pixels, as done in previous works [7].

Robust Visual Tracking via Incremental Oblique Random Forest

A discriminative classifier, tasked with distinguishing between the target and the surrounding environment, plays a central role in modern visual tracking system. Given the ground bounding box from the first frame, a discriminative classifier is trained where usually, the image patches whose overlap ratio upon a threshold are posed as positive class and those patch whose overlap ratio are below another threshold are regarded as negative class. Given the target location in previous frame, the tracker samples several image patches nearby. Those query image patches are fed into the classifier, the most confident positive result is regarded as the tracking result.

8.1 Ensemble Tracking

Ensemble methods [1, 2] have led to a great success in machine learning tasks. Theoretical explanations can be found in strength-correlation [5], bias-variance decomposition [149] and so on. Motivated by this, ensemble methods also saw heavy use in visual tracking. The pioneering work in [127] formulated each base tracker as a least-square classifier which worked directly on the image pixels. Other classical examples include boosting [128] and semi-supervised boosting [129] and random forest [150]. Randomized ensemble tracker was proposed in [151] to model the time-varying appearance of an object for visual tracking. Instead of updating classifier ensembles with the commonly used methods in tracking-by-detection pipeline, the weight vector to combine weak classifiers is posed as a random variable. The posterior distribution for the weight vector is estimated in a Bayesian manner. In [152] the authors proposed the visual tracker sampler that can work robustly in challenging

scenarios. The proposed tracking algorithm accurately tracks a target by searching for appropriate trackers in each frame. A novel tracking algorithm that combines complementary tracking modules with a new object representation model to balance between stability and adaptivity was proposed in [153]. To reduce the update error of online tracking, three complementary modules (a stable module, a soft stable module, and an adaptive module) were proposed. The authors fused them by using a biased multiplicative criterion. In [154] the authors studied the visual tracking problem in which the unknown data to be estimated is in the form of a sequence of bounding boxes representing the trajectory of the target object being tracked. As a result, they proposed a factorial hidden Markov model (FHMM) for ensemble-based tracking by learning jointly the unknown trajectory of the target and the reliability of each tracker in the ensemble. In [155], the role of each key component of the visual tracking system is analyzed and ensemble post-processing is found to boost the tracking performance significantly.

8.2 Incremental Oblique Random Forest for Visual Tracking

Motivated by Random Forest's high computational efficiency during both training and prediction while achieving state-of-the-art results, the work in [150] proposed a novel on-line random forest algorithm for visual tracking. They combined on-line bagging, random feature selection and a novel tree-growing method that allows for on-line building of orthogonal decision trees. However, due to the limited generalization ability of orthogonal decision, this method was outperformed by other tracking methods.

In this chapter, we propose a novel incremental oblique random forest classifier to address the visual tracking problem. Motivated by the success of oblique random forest classifier in Chapter 3, we employ a proximal support vector machine [156] to separate all the data sample within each internal node into 2 parts. Denoted by A ($n \times m$) be the stacked version of n training samples. The feature dimensionality of each datum is m . D is a $n \times n$ diagonal matrix which indicates the class label of the data sample.

$$D_{ii} = \begin{cases} 1 & \text{If } i^{th} \text{ sample comes from the positive class} \\ -1 & \text{If } i^{th} \text{ sample comes from the negative class} \end{cases} \quad (8.1)$$

Let e be the column vector of 1, w, γ be the coefficients and the scalar of the hyperplane. Let ε be the nonnegative slack vector and constant $\lambda > 0$ is the user-defined regularization parameter. The identity matrix is denoted by I .

Chapter 8. Robust Visual Tracking via Incremental Oblique Random Forest

The PSVM [156] aims to find the “proximal” ($x'w - r = \pm 1$ for class ± 1) planes. Data points of each class are clustered around each hyperplane and are pushed as far apart as possible. The solution with linear kernel is given by the following quadratic program:

$$\begin{aligned} \min_{(w, \gamma, \varepsilon) \in R^{n+m+1}} & \frac{1}{2} \|\varepsilon\|^2 + \lambda \frac{1}{2} (w'w + \gamma^2) \\ \text{s.t.} & D(Aw - e\gamma) + \varepsilon = e. \end{aligned} \quad (8.2)$$

According to [156], the solutions for the hyperplane can be achieved in a closed-form manner:

$$[w, \gamma]^T = (\lambda I + H^T H)^{-1} H^T D e, \text{ where } H = [A, -e]. \quad (8.3)$$

The matrix inversion in Eqn (8.3) is around $o(m^3)$, while commonly used SVM has time complexity of $o(n^3)$. This can lead to significant complexity reduction when $m \ll n$. We now describe a method to learn the PSVM incrementally. It is straightforward to see that the solution of Eqn (8.3) can be reformulated as:

$$\underset{W}{\text{minimize}} \quad \|HW - De\|^2 + \lambda \|W\|^2; \text{ where } W = [w, \gamma]^T. \quad (8.4)$$

Let $De = Y$ and suppose at time t , we have the solution $W_t = [w_t, \gamma_t]^T$. We can calculate the weight W_{t+1} at time $t + 1$ with new available data recursively from W_t without directly solving Equation (8.3). The key problem for updating the weight is the calculation of $(H^T H + \lambda I)^{-1}$. At time $t + 1$, suppose the features corresponding to the new available data is H_{t+1} , then the problem becomes

$$\underset{W_{t+1}}{\text{minimize}} \quad \left\| \begin{bmatrix} H_t \\ H_{t+1} \end{bmatrix} W_{t+1} - \begin{bmatrix} Y_t \\ Y_{t+1} \end{bmatrix} \right\|^2 + \lambda \|W_{t+1}\|^2, \quad (8.5)$$

Then, it is straightforward to observe that

$$\begin{aligned} W_{t+1} &= \left(\begin{bmatrix} H_t \\ H_{t+1} \end{bmatrix}^T \begin{bmatrix} H_t \\ H_{t+1} \end{bmatrix} + \lambda I \right)^{-1} \begin{bmatrix} H_t \\ H_{t+1} \end{bmatrix}^T \begin{bmatrix} Y_t \\ Y_{t+1} \end{bmatrix} \\ &= T_{t+1}^{-1} \begin{bmatrix} H_t \\ H_{t+1} \end{bmatrix}^T \begin{bmatrix} Y_t \\ Y_{t+1} \end{bmatrix}, \end{aligned} \quad (8.6)$$

where

$$T_{t+1} = \begin{bmatrix} H_t \\ H_{t+1} \end{bmatrix}^T \begin{bmatrix} H_t \\ H_{t+1} \end{bmatrix} + \lambda I, \quad (8.7)$$

Then,

$$T_{t+1} = \begin{bmatrix} H_t \\ H_{t+1} \end{bmatrix}^T \begin{bmatrix} H_t \\ H_{t+1} \end{bmatrix} + \lambda I = T_t + H_{t+1}^T H_{t+1}, \quad (8.8)$$

And,

$$\begin{aligned} \begin{bmatrix} H_t \\ H_{t+1} \end{bmatrix}^T \begin{bmatrix} Y_t \\ Y_{t+1} \end{bmatrix} &= H_t^T Y_t + H_{t+1}^T Y_{t+1}, \\ &= T_t T_t^{-1} H_t^T Y_t + H_{t+1}^T Y_{t+1}, \\ &= T_t W_t + H_{t+1}^T Y_{t+1}, \\ &= (T_{t+1} - H_{t+1}^T H_{t+1}) W_t + H_{t+1}^T Y_{t+1}, \\ &= T_{t+1} W_t - H_{t+1}^T H_{t+1} W_t + H_{t+1}^T Y_{t+1}. \end{aligned} \quad (8.9)$$

From Equation. 8.6,

$$\begin{aligned} W_{t+1} &= T_{t+1}^{-1} \begin{bmatrix} H_t \\ H_{t+1} \end{bmatrix}^T \begin{bmatrix} Y_t \\ Y_{t+1} \end{bmatrix} \\ &= T_{t+1}^{-1} (T_{t+1} W_t - H_{t+1}^T H_{t+1} W_t + H_{t+1}^T Y_{t+1}) \\ &= W_t + T_{t+1}^{-1} H_{t+1}^T (Y_{t+1} - H_{t+1} W_t). \end{aligned} \quad (8.10)$$

From matrix inversion lemma, we have

$$\begin{aligned} T_{t+1}^{-1} &= (T_t + H_{t+1}^T H_{t+1})^{-1} \\ &= T_t^{-1} - T_t^{-1} H_{t+1}^T (I + H_{t+1} T_t^{-1} H_{t+1}^T)^{-1} H_{t+1} T_t^{-1}, \end{aligned} \quad (8.11)$$

To summarize, let $K_t = T_t^{-1}$, the online update of the weight can be achieved by the following operation:

$$\begin{aligned} K_{t+1} &= K_t - K_t H_{t+1}^T (I + H_{t+1} K_t H_{t+1}^T)^{-1} H_{t+1} K_t \\ W_{t+1} &= W_t + K_{t+1} H_{t+1}^T (Y_{t+1} - H_{t+1} W_t). \end{aligned} \quad (8.12)$$

The term $Y_{t+1} - H_{t+1} W_t$ is also named the innovation. If innovation is zero, then the updated solution equals to the previous solution. The proposed incremental oblique random forest is straightforward. We firstly train an oblique random forest from scratch. Suppose at each internal node m_1 features are randomly selected to train a PSVM. Only $K(m_1 \times m_1)$ and $W((m_1 + 1) \times 1)$ are needed to saved. In each node, the PSVM is updated with Eqn (8.12).

8.3 Experiments

For the visual tracking task, we adopt the commonly used particle filter framework. There are six commonly used affine parameters for particle filter based trackers. More specifically, those parameters are:

$$\Psi = [t_x, t_y, S, r, a, s]; \quad (8.13)$$

The first two parameters (t_x, t_y) stand for the translations in x and y dimensions, respectively. Parameters S, r, a, s represent the scale variation, rotation angle, aspect ratio and skew angle, respectively. These parameters dictate the kind of motion of interest to a tracker and this generic dynamical model assumes the variance of each affine parameter does not change over time. They are set to be $[4, 4, 0.01, 0, 0.005, 0]$ for all the experiments in this chapter. We set the number of particles to be 400. In frame t , 400 samples x_t are sampled from previous results x_{t-1} with the following probability:

$$P(x_t | x_{t-1}) = N(x_t; x_{t-1}, \Psi); \quad (8.14)$$

We first resize the target size to be 32×32 . For the first frame, we first sample 100 positive samples by adding a tiny noise on the target location. 100 negative examples are collected from the background at a short distance from the object. There are 100 trees in the random forest. Each tree in the forest will vote for the positive class. The image patch with the most votes are considered as the tracking results. Suppose there are M votes for the results, the tracking confident is set to be $M/100$. The oblique random forest is updated when the confidence is less than 0.85. We sample 20 positive and 20 negative samples for updating when it is necessary. The proposed tracker is named as RaF. Our current implementation of the RaF runs at around 5 fps in an Intel i7 CPU.

8.3.1 Overall Performance

In order to avoid the bias in dataset selection, we evaluate the proposed RaF tracker on the recently proposed benchmark [147]. It consists of 29 state-of-the-art trackers which are evaluated on 51 challenging video sequences. The challenging factors are summarized in Table 8.1.

Figure 8.1 Overall performance of the proposed RaF tracker compared with other 29 state-of-the-art methods.

Table 8.1 Challenging Attributes in the benchmark

Attribute	Description
IV	Illumination Variation - the illumination in the target region is significantly changed.
SV	Scale Variation - the ratio of the bounding boxes of the first frame and the current frame is significantly changed.
OCC	Occlusion - the target is partially or fully occluded.
DEF	Deformation - non-rigid object deformation.
MB	Motion Blur - the target region is blurred due to the motion of target or camera.
FM	Fast Motion - the motion of the ground truth is larger than 20 pixels.
IPR	In-Plane Rotation - the target rotates in the image plane
OPR	Out-of-Plane Rotation - the target rotates out of the image plane
OV	Out-of-View - some portion of the target leaves the view
BC	Background Clutters - the background near the target has the similar color or texture as the target.
LR	Low Resolution - the number of pixels inside the ground-truth bounding box is less than 400 pixels.

Firstly, we summarize the results over all 51 video sequences in Figure 8.1. For the clarity of demonstration, only top 10 trackers are plotted. The proposed RaF tracker outperforms the current leader Struck [157] and SCM [134] tracker on the benchmark, which are more complicated to implement, with an average relative improvement of around 13%.

8.3.2 Attribute-based Evaluation

We further analyze the RaF tracker under different challenging attributes listed in Table 8.1. Fig 8.2 and Fig 8.3 shows the precision plot for all the 11 challenging factors. Firstly, our proposed RaF tracker outperforms all others in almost all cases excepted the motion problem and low resolution. This can be attributed by the effectiveness of the proposed oblique random forest classifier. For the cases of motion problem and low resolution, the target cannot be represented properly. Advanced machine learning method such as structured output nonlinear SVM may solve this problem by directly estimating the object transformation

Chapter 8. Robust Visual Tracking via Incremental Oblique Random Forest

between frames without using the tracking by detection. This will be considered in the future work.

Figure 8.2 Precision plot of RaF with under each challenging factor.

Figure 8.3 Precision plot of RaF with under each challenging factor (continued from Figure 8.2).

8.4 Conclusion

Random Forest saw heavy use in computer vision. In visual tracking, Random Forest tracker was proposed a decade ago and outperformed by other trackers soon. In this chapter, we propose an incremental oblique random forest for visual tracking where proximal SVM was served as the decision hyperplane at each node of the decision tree. We further address how to

Chapter 8. Robust Visual Tracking via Incremental Oblique Random Forest

incrementally learn the oblique random forest for visual tracking. We evaluate the proposed RaF tracker on a recently proposed benchmark which consists of 29 state-of-the-art methods and 51 challenging video sequences. Experimental results indicate that the proposed method performs favorably against all other methods.

Robust Visual Tracking via Convolutional Random Vector Functional Link Network

Recently visual tracking community has been witnessing significant performance boost with deep neural networks. The pioneering work proposed by Wang and Yeung [143] adopt a deep denosing autoencoder pretrained on 80M Tiny image dataset [158] to learn generic features robust to variations in object. In [144], ensemble of convolutional neural networks (CNN) pretrained on Imagenet [159] are used to regress the bounding box of the target in the video frame. However, the learning objective of deep denoising autoencoder differs significantly from visual tracking, where the former is to reconstruct the image patches while the latter aims at distinguishing the object from the background. On the other hand, CNN pretrained on Imagenet are designed for distinguishing generic objects. For visual tracking, not all the features are needed and some of them can be regarded as noise.

Almost all existing deep learning based trackers [143, 144, 160] rely heavily on conventional back-prop to refine the model. This fine tuning process make the pretrained model to adapt to the object being tracked. However, the tracking performance is sensitive to hyper-parameters during finetuning such as learning rate, maximum number of epoches, size of the mini-batch and so on. In this work, we addressed these issues in deep learning based tracker by (i) randomly generating the parameters in convolutional layer of CNN and kept them fixed; (ii) proposing an efficient and effective updating rule for the classifiers in CNN. Our approach, named as convolutional random vector functional link network (CRVFL), can be regarded as a marriage of the random vector functional link network (RVFL) and CNN. It is much simpler and more efficient to train and performs favorably against state-of-the-art methods. To summarize, the main contributions of the current work are as follows:

1. We aim at solving visual tracking with convolutional neural network without pretraining and finetuning.
2. We bring back the visual tracking community a very classical method: random vector functional link network (RVFL). We extend RVFL to a convolutional structure and demonstrate its feasibility for visual tracking.
3. As a result, we propose a CRVFL model which is very easy to train. Without any pretraining, the only learning step is involved in the final fully-connected layer which can be solved in a closed-form solution.
4. We propose a very simple strategy based on the recursive least square for online updating of the CRVFL model.
5. For visual tracking without any auxiliary training data, outstanding performance can be achieved by ensemble of CRVFL model on 51 challenging video sequences.

In the following section we will briefly introduce the related background before focusing on the details of the proposed method.

9.1 Tracking by CNNs

Features matter. Hand-crafted features such as SIFT [161] and HOG [162] saw heavy use in last decades but then fell out of fashion with the rise of learning representations [163]. Visual representations extracted from CNNs pretrained on very large auxiliary dataset have led to astonishing performance in various computer vision [164], speech recognition [165] and natural language processing [166] tasks.

We will discuss some relevant trackers before focusing on the details of the proposed methods. Numerous hand-crafted features have been used to represent the target appearance such as subspace representation [120], color features [167, 168], Harr-like features [157] and HOG features [7]. Recently, the visual tracking community has witnessed significant advances with CNNs which are motivated by the pioneering work done by Wang and Yeung [143]. The work in [143] is based on deep denoising autoencoder transferred from 80M Tiny Image dataset [158]. Later they apply the same idea to CNN [144] pretrained on Imagenet [159]. On the other hand, a two-layer neural network trained on a video repository was proposed in [169], where temporal slowness constraints are used for feature learning. A discriminative saliency map learned with pretrained CNN was reported in [170].

The initial motivation for our line of research was the recent success of CNN without pretraining. Visual tracking can be regarded as a self-taught learning problem where there

are very limited training samples. It remains an open problem if the object being tracked is significantly different from the information contained in the auxiliary dataset. We prefer simple models with less number of parameters to alleviate the problem of over-fitting. Along this line, perhaps the most related work is [171], where the authors formulate the tracking problem as a generative model. The tracking results are directly employed as the filters in the convolutional layer of CNN. Moreover, in [160], multiple CNN classifiers on different instances of target objects to rule out noisy sampling in updating was proposed. The so-called DeepTrack [160] also learns two layer CNN classifiers from binary samples and does not require a pre-training procedure. We address the visual tracking problem by moving one step further. The parameters in our convolutional layer are randomly generated and kept fixed. The online updating of the model is effective in a recursive fashion without introducing any hyper-parameters compared with conventional back-prop methods.

9.2 Convolutional Neural Network

In this section we present detailed information about Convolutional Neural Network (CNN). For RVFL, which is the other key component of the proposed method, we refer the readers to Chapter 2 Chapter 4.

Biologically inspired by visual cortex, Convolutional Neural Network (CNN) learns some discriminant features that generalizes well on many tasks such as image classification [164, 172], speech recognition [173, 174], object detection [175], image segmentation [176] and so on. Typically, a CNN may consider the following basic structures.

Convolutional Layer In the convolutional layer of CNN, each neuron is locally connected to parts of the neurons in the previous layer. As demonstrated in Fig. 9.1, a common implementation is to convolve the input with a pre-defined filter. Mathematically speaking, denote by X the input data with size $m \times n \times k$ (where m, n, k correspond to the height, width and depth of the input, respectively) and by W with size $j \times k \times l$ filter bank. The output of the convolutional layer is:

$$Y = g(\text{conv}(X, W)) \quad (9.1)$$

And g is some non-linear function, among which, $\text{Relu}(y = \max(x, 0))$ and $\text{Square}(y = x^2)$ are commonly used. Unshared weights in convolutional layer is reported in [177], where different spatial location in the input map may be convolved with different filters. Recent studies show that better performance can be achieved with over-complete feature maps in convolutional layer.

Figure 9.1 A convolutional node in CNN.

pooling layer Pooling layer in CNN is usually followed by convolutional layer to achieve some degree to shift invariance, playing the same role as the complex cells in models of visual perception. Different kinds of pooling operators are proposed in the literature such as max pooling, average pooling, stochastic pooling [178], $l - p$ norm pooling [179], square root pooling [177].

Normalization Layer Another important CNN building block is channel-wise normalization. After pooling layer, a normalization layer can be followed to perform feature-wise sliding window normalization which is defined by the following equation:

$$y_{ijk} = x_{ijk} (\kappa + \alpha \sum_{t \in G(k)} x_{ijt}^2)^{-\beta} \quad (9.2)$$

where $G(k)$ is $[\kappa - \lfloor \rho/2 \rfloor, \kappa + \lceil \rho/2 \rceil] \cap \{1, 2, \dots, K\}$, K is the number of feature maps before the normalization layer.

Fully Connected Layer Finally, after several convolutional, pooling layers and normalizing layer, the high-level reasoning in the neural network is done via fully connected layers. A fully connected layer takes all neurons in the previous layer, be it fully connected, pooling, or convolutional, and connects it to every single neuron it has.

CNN is often used in image and speech based recognition applications. Designing of CNN is usually task-dependent. Interested readers can refer [180] for more information about how to choose the basic structures.

Figure 9.2 Basic structure of the CNN used in this work.

9.3 Convolutional Random Vector Functional Link Neural Network

In [181], the authors pointed out that certain convolutional pooling architectures could be inherently frequency selective and translation invariant, even with random weights. Also in our previous work [182], we found the direct link from input to output neuron may serve as a good guidance for a randomized neural network. Motivated by these, we propose Convolutional Random Vector Functional Link Neural Network (CRVFL) in this work. CRVFL can be regarded as a marriage of RVFL and CNN, where the filters in the convolutional layer are randomly generated and fixed. We use this simple method to simplify modern state-of-the-art deep learning based visual tracking system.

As demonstrated in Fig. 9.2, the input features of the last fully connected layer is a concatenation of the original input feature and the penultimate feature maps. “conv”, “pool”, “norm” and “Fc” stand for convolutional layer, pooling layer, normalization layer and fully connected layer, respectively. The weights in the fully connected layer can be solved by minimizing the square errors of the output and the targets. Denoted by H the “combined” features in the fully connected layer and by Y the desired target, the weights W can be derived by simply $W = H^+Y$ with MP pseudo-inverse or regularized least square.

Compared with conventional gradient based approach which needs many iterations and heuristic approaches to tune the parameters (such as learning rate and weight decay) for convergence, this method is easy to train because of its closed-form solution. However, in some applications where the data comes in a sequential manner. In this case, we can extend

Chapter 9. Robust Visual Tracking via CRVFL

the CRVFL based on the recursive least square [183–186] approach to make the proposed approach be updated online. The weights in the fully connected layer is solved by

$$W = (H^T H + \lambda I)^{-1} H^T Y, \quad (9.3)$$

where λ is a user-defined parameter. Suppose at time t , we have the solution W_t . The RLS approach calculate the weight W_{t+1} at time $t + 1$ with new available data recursively from W_t without directly solving Equation 9.3. The key problem for updating the weight is the calculation of $(H^T H + \lambda I)^{-1}$. At time $t + 1$, suppose the features corresponding to the new available data is H_{t+1} , we can follow the exactly pipeline proposed in Chapter 8 to recursively update the solution.

To summarize, let $K_t = T_t^{-1}$, the online update of the weight can be achieved by the following operation:

$$\begin{aligned} K_{t+1} &= K_t - K_t H_{t+1}^T (I + H_{t+1} K_t H_{t+1}^T)^{-1} H_{t+1} K_t, \\ W_{t+1} &= W_t + K_{t+1} H_{t+1}^T (Y_{t+1} - H_{t+1} W_t). \end{aligned} \quad (9.4)$$

The term $Y_{t+1} - H_{t+1} W_t$ is also named the innovation. If innovation is zero, then the updated solution equals to the previous solution. In the fully connected layer, we concatenate the input features with the output features in the penultimate layer. The number of output neurons equals to the number of classes of the specified task. If the concatenated version of feature is H and the output is Y and the weights in the fully-connected layer is W , then this layer aims to solve the the problem from Equation 9.3. The basic structure is demonstrated in Fig 9.2. It has 6 layers in total. We have two normalization layer after the pooling layer and before the fully connected layer as we empirically find normalization layer slightly boosts the performance since it provides extra robustness to drastic illumination variations. The input are the image patches cropped from the positive class (which comes from the object) as well as the negative class (which is the background). The second layer is the convolutional layer. It convolves the input with a set of random filter banks. The third layer is the 2D pooling layer, followed by 2 normalization layer which normalize the features in a feature-wise sliding window manner as defined by following equation:

$$y_{ijk} = x_{ijk} (\kappa + \alpha \sum_{t \in G(k)} x_{ijt}^2)^{-\beta} \quad (9.5)$$

where $G(k)$ is $[\kappa - \lfloor \rho/2 \rfloor, \kappa + \lceil \rho/2 \rceil] \cap \{1, 2, \dots, K\}$, K is the number of feature maps before the normalization layer. The last layer is the fully connected layer which fully connects the

concatenation of the input and the output of the last normalization layer with the output neuron.

9.4 Experiments

9.4.1 Implementation Details

There are 100 filter banks in the convolution layer. Each of them has the size of $4 \times 4 \times k$, where k is the number of channels of the input patch. They are initialized with a zero mean and unit variance Gaussian distribution normalized by a constant of 100. The convolution is performed with stride 1 and a padding of 2 at the boundary of the image. The pooling size of the pooling layer is 2×2 with stride 2 and without padding. Averaging operator is employed in this layer. After pooling layer, two normalization layers are followed to perform feature-wise sliding window normalization which is defined by the following equation: We set $\kappa = 1$, $\alpha = 1/16$, $\rho = 4$, $\beta = 0.75$ in the first normalization layer and $\kappa = 1$, $\alpha = 1/4$, $\rho = 2$, $\beta = 0.5$ in the second normalization layer. All these parameters are searched through a small number of randomly selected video sequences and all these parameters of our tracker are fixed throughout the experiments.

In the implementation of the CNN tracker, only translation is considered for efficiency. Search for the target is conducted on a Cartesian grid of unit step within a radius of $\sqrt{wh}$ of the previous prediction, where (w, h) is the template size. The input image patches are rescaled such that $w \times h$ is less than 144. The predicted position gives the positive sample, and the local image patches that do not significantly overlap the prediction (overlap ratio less than 0.5) are cropped as the negative ones. The output scores of the CRVFL model are transformed to the probability form by a Gaussian cumulative distribution with mean of 0.5 and STD of 1. The tracker will be updated only when this probability is less than 0.9.¹

9.4.2 Results and Discussion

In this section, we report an extensive evaluation of the proposed online CRVFL model for visual tracking denoted as CRVFL, in comparison with other state-of-the-art trackers. We follow the same protocol to evaluate the proposed methods as done in Chapter 8. Instead, we adopt the criterion of precision curve, which is now considered as an increasingly popular alternative of the previously mentioned evaluation criteria [7, 130, 147, 187]. A frame may be considered correctly tracked if the predicted target center is within a distance threshold of

¹Source code will be available.

Chapter 9. Robust Visual Tracking via CRVFL

Table 9.1 Performance (Mean precision (20 px)) comparisons of proposed methods with conventional back propagation based baseline

Method	overall	IV	SV	Occ	Def	MB	FM	IR	OR	OV	BC	LR
CNN-bp	72.5	66.2	74.3	72.6	67.2	72.9	60.6	69.0	71.0	57.2	73.4	40.2
CRVFL	79.0	79.9	81.9	77.0	78.1	75.9	79.7	77.2	78.6	72.6	73.6	61.8

Some keys are: IV=Illumination variation, SV=Scale variation, Occ=Occlusion, Def=Deformation, MB=Motion Blur, FM=Fast Motion, IR=In-Plane Rotation, OR=Out-of-Plane Rotation, OV=Out of View, LR=Low Resolution.

ground truth. Precision curves simply show the percentage of correctly tracked frames for a range of distance thresholds. No parameters are required when plotting the precision for all thresholds, which makes the curves unambiguous and easy to interpret. It is obvious that a higher precision at low thresholds means the tracker is more accurate, while a lost target will prevent it from achieving perfect precision for a very large threshold range. As done in [7, 130, 147, 187], we set the threshold to be 20 pixels.

Figure 9.3 Overall performance of a single CRVFL model.

9.4.3 Overall Performance

Firstly, we summarize the results over all 51 video sequences in Fig 9.3. Our single CRVFL model achieves a score of 79.0% for the precision above threshold 20. We also implement a baseline tracker (named as “CNN-bp”) which is trained and updated with conventional back propagation in order to demonstrate the superiority of the proposed method. More specifically, for training and tuning of the baseline model, we set the batch size to be 10 and learning rate to be 0.001. The maximum number of epochs is 5. We search these parameters on a small number of frames (same as the those used in CRVFL) and keep them fixed throughout all the experiments. We summarize the comparison in Table 9.1.

It is widely accepted that ensemble based integration of multiple trackers can boost the performance [118]. Hence, we also conduct another set of experiments to evaluate the feasibility of an ensemble of CRVFL models. In this study, we adopt the entropy minimization [188] strategy to integrate CRVFL and its historical snapshots as done in [167], where the best CRVFL model is selected to restore the current tracker when needed based on a minimum entropy criterion. It aims at maximizing the log posterior probability of the model parameterized by θ ,

$$C(\theta, \beta; L) = L(\theta; x, z) - \beta H_{emp}(y|x, z; \theta); \quad (9.6)$$

where $L(\theta; x, z)$ is the log likelihood of the model parameters θ and $H_{emp}(y|x, z; \theta)$ are the empirical conditional entropy of class labels conditioned on the training data and the possible label sets. x and z are the training data and label sets. β is the trade-off parameter. The entropy regularization term favors models with low ambiguity. For example, when a label set contains two possible labels, then a model giving equally high confidence scores to both labels is less favored than a model giving a high confidence score to one label and a low confidence score to the other. We refer the reader to [167, 188] for more details. Under this framework, the maximum number of experts N and the time interval for saving a snapshot of CRVFL are set to 4 and 30, respectively. The trade-off parameter β in Equation 9.6 is set to be 100. We choose the best CRVFL snapshot based on Equation 9.6 with the past 4 frames. All other parameters are set to be the default value in [167]. Hereby, we name this approach as CRVFLEn and its overall performance is summarized in Fig 9.4.

Figure 9.4 Overall performance of a ensemble of CRVFL model.

The rank in Fig 9.3 and Fig 9.4 is valid only when the score is smaller than 20 pixels. Therefore, there exists some possibilities that excellent performance on one video sequence to compensate for poor performance on many others. Thus, to better demonstrate the feasibility

Chapter 9. Robust Visual Tracking via CRVFL

of the proposed CRVFL model for visual tracking, total number of sequences that fulfill this criteria are listed in Table 9.2. It can be seen that the proposed tracker has successfully

Table 9.2 Number of successful sequences and overall performance

Ranking	Method	Number in total	Precision score(%)
1	CRVFLEn	43	86.3
2	MEEM [167]	39	83.6
3	CRVFL	37	79.0
4	Struck [157]	29	65.6
5	SCM [134]	28	64.9
6	TLD [189]	21	60.8
7	DLT [143]	23	58.7
8	VTD [123]	21	57.6
9	VTs [152]	22	57.5
10	CXT [190]	21	57.5

tracked 43 sequences with mean precision is below 20 pixels, which account for around 84.31% of the total number of sequences in the whole benchmark, which outperforms the second best one by 4 more sequences. This demonstrates that tracking with CRVFLEn enables a base tracker to be capable of handling more challenging situations. We also observe that the base tracker, CRVFL, can successfully track 37 sequences, which lead to a significant improvement over the Struck tracker.

9.4.4 Attribute-based Evaluation

The attributes annotated with the video sequence in the benchmark describe the challenge which a tracker will face in each sequence. It enable practitioners to diagnose and characterize the behavior of trackers without having to analyze each individual video. In Fig. 9.5 and Fig. 9.6 we report the results for each attribute. It can be easily observed that CRVFLEn model achieve the best results in almost all cases. In order to have a better demonstration about the CRVFLEn model, we summarized the improvement over MEEM in Table 9.3.

9.4.5 Comparison with Existing Deep Learning Methods

In this section we provide further comparisons with more state-of-the-art deep learning methods proposed recently. Perhaps one reason that hinders the application of deep neural network and CNN to visual tracking is the lack of properly labeled training data. One straightforward approach to address this issue is to pretrain a deep neural network/CNN offline and then transferring the rich feature hierarchies learned to online tracking. In [143], a deep autoencoder is offline trained with 80 million tiny image dataset [158] then updated online for visual tracking task. However, the offline training of autoencoder which aims at

Figure 9.5 Attribute-based Evaluation.

Figure 9.6 Attribute-based Evaluation (continued from Figure 9.5).

Table 9.3 Improvement of the mean precision (over 20 pixels) over MEEM for each attributes

Attributes	Improvement over MEEM[167]
All attributes	2.7%
Illumination variation	3.8%
Scale variation	4.1%
Occlusion	6.1%
Deformation	2.9%
Motion blur	7%
Fast motion	3.7%
In-plane rotation	2.4%
Out-of-plane rotation	2.6%
Out of view	0.1%
Background clutter	2.7%
Low resolution	12.4%

reconstruction of the input image may not be very suitable for tracking. Features that are effective for tracking should be able to distinguish the object from the background. In [144], the author propose to use CNN which is pre-trained on the Imagenet dataset [159]. The CNN model aims at generating a probability map of the bounding box instead of producing a simple class label. During updating, some augmentation of the tracking result are involved to ensure some scale and translation variants. The same idea can be found in [169], where the authors offline learn features robust to diverse motion patterns from auxiliary video sequences with a two-layer convolutional neural network. In [169], a CNN is trained offline on auxiliary video sequences, where the temporal slowness constraint in the stacked architecture is involved to make the learned features robust to complicated motion transformations.²

However, there may exist large discrepancy between the source domain (the auxiliary data) and the target domain (video sequence for tracking). Almost all the existing deep learning methods employ finetuning to address this issue with the conventionally used back propagation, which is very sensitive to a wide range of parameters such as the number of epochs, size of mini-batch, learning rate and momentum when the training samples are extremely limited.

The CRVFL model proposed in this work can be regarded as a semi-randomized CNN with direct link from the input layer to the output layer which is much simpler to train and update for online applications. Without any auxiliary data, this model outperforms those deep learning model with only one single convolutional layer, which is one of the computational bottleneck of CNN. Most importantly, the main learning step is only involved in calculating the weights in the fully connected layer. Suppose the concatenated features have a dimension of p . The step that dominates the computational cost for batch learning in Equation 9.3 is the

²Result of this method is obtained by personal communication with the authors.

matrix inversion which takes $o(p^3)$ flops. For the online update in Equation 9.4, the matrix inversion takes $o(n^3)$ flops and the overall time complexity of matrix multiplication is at $o(np^2)$, where n is the number of new available data and in our case, $n \ll p$

In [171], the author proposed to use CNN to extract a robust representation in a generative manner. The filters in CNN were directly cropped from the tracking result. During updating, only the filters were updated incrementally to accommodate appearance changes over time for robust visual tracking with a simple temporal low-pass filtering method, which may also be regarded in a closed-form solution. This approach is also very simple to train and update. However, it operates in the generative manner and turns out to be less accurate in challenging environments.

We have summarized those methods whose results are publicly available in Table 9.4. In order to check the importance of direct link in Fig 9.2, we also report one variant of the proposed method, named as 'CRVFLEn-d', which represents the same model as CRVFLEn without direct link. The results are also inline with our previous study [182], which also indicates that the input feature helps to regularize the random features in the model. Our current implementation of the CRVFLEn runs at around $1.5 \sim 2$ fps with a single thread in a Intel i7 CPU, which is comparable to the deep learning approaches. It is possible to significantly reduce the computational time by advanced GPU/GPGPU computing [191].

Table 9.4 Results of deep learning method

Method	Mean precision (20 px)	Pre-training	Updating
DLT [143]	58.7%	Yes	Back-prop with SGD
SO-DLT [144]	81.9%	Yes	Back-prop with SGD
CNT [171]	61.2%	No	Closed-form
CNN-SVM [170]	85.2%	Yes	Incremental SVM
LHF [169]	81.2%	Yes	Iterative with L-BFGS
CRVFLEn	86.3%	No	Closed-form
CRVFLEn-d	83.3%	No	Closed-form

CRVFLEn-d is CRVFLEn without direct link.

9.5 Conclusion

Recent advances in visual tracking rely heavily on deep convolutional neural networks pretrained with extremely large auxiliary dataset such as ImageNet. However, as there exist significant discrepancy between the auxiliary dataset and the object being tracked, those models need to be finetuned through back propagation. In order to simplify the visual tracking system, we propose convolutional random vector functional link neural network, where the filters in the convolutional layer are randomly initialized and kept fixed and only the parameters in the fully connected layers need to be learned. We further propose to update

those parameters with recursive least squares. Extensive experimental results show that the proposed algorithm performs favorably against the state-of-the-art methods in the recently proposed visual tracking benchmark.

Robust Visual Tracking via Co-trained Correlation Filters

Though many efforts have been devoted in recent years and significant improvement can be found in the literature, visual tracking remains challenging for real-life application. On one hand, single classifier may be not powerful enough to handle complex background and significant variations of target appearance, and ensemble methods seem to be a promising approach to significantly improve the tracking performance [118]. Ensemble learning can be either integrated in the learning phase [151] or as a post-processor to integrate multiple base trackers [154]. On the other hand, modern trackers suffer from relatively large computational complexity, which is a major hurdle that hinders the ensemble based visual tracking. In this work, in order to strike a good trade-off between the robustness and speed in visual tracking, we embark on KCF tracker [7], which is very efficient in training and updating the binary classifier. We jointly train an ensemble of base trackers in a closed-form fashion. We show that the realization of the proposed methods are able to outperform the KCF tracker by a larger margin and perform favorably against other state-of-the-art trackers.

10.1 Correlation Filters for Tracking

In this section we review the correlation filter based visual tracking. Other related work such as ensemble tracking, we refer the readers to previous chapters for details.

Correlation filters belong to the well-known tracking-by-detection framework. Since the pioneering work in [192], correlation filters based tracking has attracted considerable attention due to its high computational efficiency via fast Fourier transform. They have been proven to be competitive with far more complicated tracking methods [7, 187] while

operating at real-time speed. Recent work shows that KCF tracker, a kernelized version of correlation filters, outperforms other trackers in this family [7]. In the Fourier domain the training data, which is presented in a circulant matrix to avoid hard negative mining, is made diagonal [193]. Except for DFT and inverse DFT, which bounds the cost at a nearly-linear $O(n \log n)$, the cost of the element-wise operation in the Fourier domain is $O(n)$, which is much smaller than the conventional method ($O(n^3)$) [7]. In [7], without hard negative mining, the authors assume that the cyclically shifted version of the base sample is able to approximate the dense samples over the base sample. For notational simplicity, we choose one dimensional signal to demonstrate the idea of KCF. The results generalize to two dimensional multi-channel signals [7]. Given an $1 \times n$ vector $x = [x_1, x_2, \dots, x_n]$ representing a data sample, a cyclic shift of x can be represented as $Px = [x_n, x_1, x_2, \dots, x_{n-1}]$. The data matrix $X = C(x)$ turns out to be a matrix which is a concatenation of all the cyclic shift visual samples, $\{P^u x | u = 0 \dots n-1\}$. Thus, X is called circulant matrix and can be made diagonal by DFT matrix:

$$X = F \text{diag}(\hat{x}) F^H \quad (10.1)$$

where F is known as the DFT matrix which transforms the data into Fourier domain $\hat{x}$ and F^H is the Hermitian transpose of F . The learning problem is formulated by ridge regression [194], which aims at minimizing the squared error over the output of sample x_i and its desired target y_i with a function $f(z) = w^T z$.

$$\min_w \sum_i^n (f(x_i) - y_i)^2 + \lambda \|w\|^2 \quad (10.2)$$

where λ is the regularization parameter. x_i is the i^{th} row of the data matrix X . The regression labels¹ of the training sample are generated in a Gaussian function, which equals to 1 for the centered target and smoothly decays to 0 for the cyclic shifts.

It is easy to obtain the solution for the ridge regression as $W = (X^T X + \lambda I)^{-1} X^T y$. Combining with Eqn. (10.1), we have the solution $\hat{w} = \frac{\hat{x}^* \odot \hat{y}}{\hat{x}^* \odot \hat{x} + \lambda}$, where $*$ denotes the complex-conjugate. The solution can also be generated for the non-linear case with kernel trick,

$$f(z) = w^T z = \sum_{i=1}^n \alpha_i \kappa(z, x_i), \quad (10.3)$$

¹We follow the notions in [7]. Regression with class label can be regarded as classification. Hence, these two terms are used interchangeably in this paper.

Chapter 10. Robust Visual Tracking via Co-trained Correlation Filters

Most commonly used kernel functions lead to a circulant matrix [7] and the dual space coefficients α can be learnt as below:

$$\hat{\alpha} = \frac{\hat{y}}{\hat{k}^{xx} + \lambda} \quad (10.4)$$

where k^{xx} is called as the kernel correlation in [7] and can be computed as

$$k_i^{xx} = \kappa(x, P^{i-1}x). \quad (10.5)$$

Since the work in [7], there exist numerous extensions for KCF trackers. In [195], part based KCF was proposed to address occlusion. Reliable Patch Trackers (RPT) can be found in [196]. Specifically, the authors in [196] presented a tracking reliability metric to measure how reliably a patch can be tracked, where a probability model is proposed to estimate the distribution of reliable patches under a sequential Monte Carlo framework. Correlation filter was proposed in [197] to model the spatio-temporal context information. In [198], a scale pyramid representation by learning discriminative correlation filters was proposed. Multiple kernel for correlation filter based tracking can be found in [199]. The so-called MKCF tracker fully takes advantage of the invariance-discriminative power spectra of various features to further improve the performance. In [200], an online random fern classifier was employed as re-detection component for long-term tracking. A biology-inspired framework where short-term processing and long-term processing are cooperated with each other [201] in correlation filter based trackers. In [8], the authors hierarchically infer the maximum response of ensemble of correlation filters trained on the features extracted from different layers of ConvNets. Given the set of correlation response maps f_l , targets location can be hierarchically inferred. More specifically, let $(\bar{m}, \bar{n}) = \operatorname{argmax}_{m,n} f_l(m, n)$ indicate the location of the maximum values on the l^{th} layer, the optimal location of the target in the $(l-1)^{th}$ layer can be inferred as:

$$\begin{aligned} & \operatorname{argmax}_{m,n} f_{l-1}(m, n) + \gamma f_l(m, n), \\ & s.t. \quad |m - \bar{m}| + |n - \bar{n}| \leq r \end{aligned} \quad (10.6)$$

The problem defined above indicates that only the $r \times r$ neighbouring region of $(\bar{m}, \bar{n})$ are searched in the $(l-1)^{th}$ correlation response map. The target location is finally estimated by hierarchically maximizing Eqn (10.6).

10.2 Co-trained Correlation Filters

Our approach builds on the aforementioned related methods. Single classifier may be not powerful enough to handle complex background and significant variations of target appearance. Thus, some attempts have been made to employ multiple models in visual tracking. Classical approach is ensemble tracking [127]. However, ensemble tracking suffers from high computational time which makes them unsuitable for real-time applications. Thus, in this work, we strike a good trade-off between the robustness and speed in visual tracking. As a result, we embark on KCF tracker, which is very efficient in training and updating classifiers, and jointly train an ensemble of base trackers in a closed-form fashion. Extensive experiments on recently proposed benchmark dataset indicate that the proposed algorithm performs favorably against state-of-the-art methods.

10.2.1 Jointly Training of Ensemble Correlation Filters

The kernel matrix in Eqn. (10.4) plays a central role in KCF tracker. Actually different kernels may correspond to different notions of similarity, and using a specific kernel may lead to a risk of bias and result in drift in tracking. Multiple kernel learning (MKL) tackles this problem by using a combination of different kernels and is reported to consistently improve the performance of kernel methods in machine learning [202] and computer vision [203–205]. However, though MKL or other ensemble methods can improve the stability of the tracker, they usually significantly reduce the tracking speed due to high computational complexity in the optimization process. (An example can be found in [199], where a two kernel correlation filters is proposed in the MKL framework. With the same feature, our method achieves almost the same performance with 10 times faster in fps.) Hence, in this study, we propose a simple method which jointly trains a pool of classifiers, and most importantly, inherits the efficiency of KCF tracker in the Fourier domain.

Suppose that we have M kernels, then the proposed method aims at solving the following problem:

$$T(\alpha) = \min_{\alpha} \sum_{i=1}^M (\|y - K_i \alpha_i\|^2 + \lambda \alpha_i^T K_i \alpha_i) + \beta \sum_{i,j=1}^M \|K_i \alpha_i - K_j \alpha_j\|^2 \quad (10.7)$$

The first term in Eqn (10.7) forces each individual model to have minimal squared error with respect to the desired output y . The second term $(\alpha_i^T K_i \alpha_i)$ denotes the Tikhonov

Chapter 10. Robust Visual Tracking via Co-trained Correlation Filters

regularization in a Reproducing Kernel Hilbert Space. The third term regularizes each model in a pair-wise fashion which weights the influence of pairwise disagreements. It allows high performance to be achieved since it becomes unlikely for compatible classifiers trained on independent kernels/features to agree on an incorrect label. When $\beta = 0$, the proposed method is equivalent to M independent kernel ridge regression problems. The proposed method can also be regarded as a purely supervised version of the co-regularized multiview learning [117]. It is worth noting that the proposed method can have many realizations. If we generate each kernel by using different kernel functions such as Gaussian, polynomial and linear kernel or with different kernel parameters, the resulting tracker can be regarded as an MKL tracker [204, 205]. One can also generate each kernel with different features such as HOG [162], SIFT [206], Haar-like [207], or LBP [208]. Another approach could be a part-based tracker if each kernel is generated with only fixed part of the object patch.

In the present work, we simply generate two Gaussian kernels K_1, K_2 with the same parameters and with different features, respectively. This approach is named as Co-trained KCF (COKCF) tracker. We have conducted two sets of experiments, one is based on handcrafted features and the other is with features extracted from Convolutional Neural Network (ConvNets) [209].

It is straight-forward to obtain the solution of Eqn (10.7) by setting the derivation of T with respect to α_i to be zero:

Let

$$G_i = (2\beta + 1)K_i K_i + \lambda K_i, \quad (10.8)$$

we can obtain the solution for Eqn (10.7)

$$\begin{aligned} \alpha_1 &= (G_1 - 4\beta^2 K_1 K_2 G_2^{-1} K_2 K_1)^{-1} \\ &\quad * (K_1 y + 2\beta K_1 K_2 G_2^{-1} K_2 y), \\ \alpha_2 &= (G_2 - 4\beta^2 K_2 K_1 G_1^{-1} K_1 K_2)^{-1} \\ &\quad * (K_2 y + 2\beta K_2 K_1 G_1^{-1} K_1 y), \end{aligned} \quad (10.9)$$

Let $v = 2\beta + 1$ and notice

$$G_i^{-1} = (\beta K_i + \lambda I)^{-1} * K_i^{-1}, \quad (10.10)$$

Substituting Eqn (10.8) into Eqn (10.9), we have

$$\begin{aligned} \alpha_1 &= ((vK_1 + \lambda I) - 4\beta^2 K_2 (vK_2 + \lambda I)^{-1} K_1)^{-1} \\ &\quad * (y + 2\beta K_2 (vK_2 + \lambda I)^{-1} y), \end{aligned} \quad (10.11)$$

By applying Eqn (10.1), we have

$$\begin{aligned}\hat{\alpha}_1 = & \text{diag}(\mathbf{v}k_1 + \lambda - 4\beta^2 \frac{k_2 \odot k_1}{\mathbf{v}k_2 + \lambda})^{-1} \\ & * \text{diag}(1 + \frac{2\beta k_2}{\mathbf{v}k_2 + \lambda}) * \hat{y},\end{aligned}\tag{10.12}$$

or better yet,

$$\begin{aligned}\hat{\alpha}_1 = & \left(\frac{(1 + 4\beta)k_2 + \lambda}{(1 + 4\beta)k_1 \odot k_2 + \lambda(2\beta + 1)(k_1 + k_2) + \lambda^2} \right) \odot \hat{y} \\ = & \frac{\hat{y}}{k_1 + \lambda + \frac{2\beta\lambda(k_1 - k_2)}{(1 + 4\beta)k_2 + \lambda}}\end{aligned}\tag{10.13}$$

where k_1, k_2 can be computed in the same way as Eqn (10.5). The above matrix multiplication/division can be efficiently computed in an element-wise manner since they are all diagonal matrices. Similar solution can be derived for α_2 .

Similar to KCF, the pipeline for the COKCF tracker is intentionally simple and does not include any heuristic for failure detection or motion modelling. In the first frame, we generate two kernels K_1, K_2 with the same Gaussian kernel and with HOG and color attributes, respectively. For each new frame, detection is performed at the previous position.

10.2.2 Model Update

We adopt a similar approach to update the models for visual tracking. More specifically, the solution of the correlation filters for each base model are updated with moving average:

$$\alpha_t = (1 - \eta)\alpha_{t-1} + \eta\alpha_t;\tag{10.14}$$

where t is the frame index and η is the learning rate.

10.2.3 An Insight into the Methodology

As we mentioned above, the first term in Eqn (10.7) forces each individual model to have the minimal squared error with respect to the desired output y . The second term $(\alpha_i^T K_i \alpha_i)$ denotes the commonly used Tikhonov regularization in a Reproducing Kernel Hilbert Space. The third term regularizes each model in a pair-wise fashion which weights the influence of pairwise disagreements. It allows high performance to be achieved since it becomes unlikely for compatible classifiers trained on independent kernels/features to agree on an incorrect label. For example, if one model becomes less accurate for a new arrival datum, the last constraint will force the output of this model to be as accurate as other models

Chapter 10. Robust Visual Tracking via Co-trained Correlation Filters

in the ensemble. At first glance, the proposed method looks contradictory to the existing

Table 10.1 Performance (Mean precision (20 px)) comparisons of *CoKCF_HC* with *KCF*

Methods	Overall	IV	SV	Occ	Def	MB	FM	IR	OR	OV	BC	LR
CoKCF_HC	77.5	72.0	76.2	68.5	77.7	79.8	66.0	57.9	73.2	62.9	70.2	38.8
KCF	74.3	73.3	73.3	67.9	75.4	74.8	60.9	61.0	72.9	65.0	75.3	38.1

Some keys are: IV=Illumination variation, SV=Scale variation, Occ=Occlusion, Def=Deformation, MB=Motion Blur, FM=Fast Motion, IR=In-Plane Rotation, OR=Out-of-Plane Rotation, OV=Out of View, LR=Low Resolution.

Negative Correlation Learning (NCL) approach [210]. In NCL, each base model is forced to be different to each other in order to have decorrelated errors. However, NCL was originally proposed to regularize neural network ensembles. Without negative correlation constraints, all base neural network model may converge to the same local minimum solution. However, the proposed method can be regarded as the positive correlation learning. It is effective because here each kernel is constructed with either different source of features or different kernel functions. It is unlikely for those inherent diversity of kernels to have the same solutions for a given problem. The positive correlation constraint can force the output of one “poor” model to be as accurate as other “good” models in the ensemble. We also have conducted experiments with NCL and found the results to be much worse than the proposed methods here, as we expected.

10.3 Experiment

10.3.1 Overall Performance

Firstly, we evaluate our tracker by using the benchmark which has 51 video sequences [147]. The details of the bechmark dataset can be found in Section 8.3.1.

CoKCF_HC (CoKCF with Handcrafted Features):

In this section, we implement the CoKCF tracker based on two handcrafted features: HOG [162] and color features [211]. We name the proposed method with these two features as CoKCF_HC. For the HOG feature, we adopt the same configurations used in [7] which sets the cell size to be 4 and number of orientations to be 9. We use the same color features in [211]. For this part, the two base trackers operate in a competitive manner, which means the one with larger peak-to-sidelobe ratio (PSR) will give the tracking result as the position of detection with the maximum value based on Eqn (10.3). Finally, we update the tracker by linearly interpolating the resulting α and x with the ones from the previous frame. The

learning rate is set as $0.002 * PSR$. The regularization parameter λ and β are set to be e^{-4} and 0.05 respectively and the width parameter for the Gaussian kernel is 0.5 (the same as KCF) throughout all the experiments. As mentioned earlier, the overall complexity of each base KCF tracker is dominated by the DFT, which is $O(n \log n)$. Thus, the overall complexity of the whole ensemble tracker is around $O(Mn \log n)$, where M is the number of the base trackers. In our regular PC with Intel-i7-3770 CPU (3.4 GHz) and 32 GB RAM, KCF tracker operates at an average fps of around 260 while CoKCF runs at around 130 fps (including feature extraction).²

We firstly summarize the results over 51 video sequences in Table 10.1. We can see that the proposed method demonstrates its superiority over the KCF tracker in overall performance and most of the challenging scenarios.

CoKCF_CNN (CoKCF with Features extracted from ConvNets):

Table 10.2 Performance (Mean precision (20 px)) comparisons of CoKCF_CNN with KCF [7] and the KCF_CNN [8]

Methods	Overall	IV	SV	Occ	Def	MB	FM	IR	OR	OV	BC	LR
<i>KCF</i>	74.3	73.3	73.3	67.9	75.4	74.8	60.9	61.0	72.9	65.0	75.3	38.1
KCF_CNN	89.2	84.6	87.0	88.0	87.9	88.3	84.8	79.2	87.0	69.5	88.5	89.7
CoKCF_CNN	88.9	83.4	86.2	88.9	87.1	84.7	81.3	76.4	86.4	70.2	89.2	93.8

Some keys are: IV=Illumination variation, SV=Scale variation, Occ=Occlusion, Def=Deformation, MB=Motion Blur, FM=Fast Motion, IR=In-Plane Rotation, OR=Out-of-Plane Rotation, OV=Out of View, LR=Low Resolution.

Handcrafted features, such as HOG [162], SIFT [206] and LBP [208], saw heavy use in the last decade, but then fell out of fashion with the rise of deep ConvNets trained on large-scale datasets [164]. In [164], the authors rekindled interest in ConvNets by demonstrating significantly better image classification accuracy on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [212].

In [213], the authors show generic descriptors extracted from the ConvNets are very powerful for a wide range of recognition tasks. Motivated by this work, Chao Ma et al. propose to train correlation filters with the features extracted from different layers of ConvNets. They hierarchically infer the maximum response of each layer to locate targets [8]. We name this method as KCF_CNN. In this section, we follow the same pipeline as done in [8]. More specifically, we use the VGG-19 model [209] to encode target appearance. We train two sets of correlations filters with features extracted from *conv4* – 4 and *conv5* – 4

²Source code will be available at the authors' homepage.

layers (which correspond to the 28th and 37th layers of the VGG-19 model [209]). We resize those two output maps with bilinear interpolation to be the same as the window size (e.g., 1.8 times of the target size). The regularization parameters λ and β are set to be 10^{-5} and 10^{-3} , respectively. We find that the tracking performance is not sensitive to these settings. Linear kernel is used. Moreover, the learning parameter is set to be 0.01 for these two models. We name the proposed method on features extracted from ConvNets as CoKCF_CNN to differentiate from the previous section. The current implementation of the CoKCF_CNN tracker runs at around 9 fps at our K40 GPU, while the work in [8] operates at about 6 fps on the same machine.

In the same way, we summarize the detailed results of the proposed method in Table 10.2. We also summarize the detailed result of KCF_CNN in [8]. Table 10.2 clearly demonstrates the superiority of tracking with ConvNet features. The proposed method achieves competitive performance compared with [8]. Note that the proposed method is both conceptually simpler and computational cheaper (around 1.5 times faster) than the method in [8]. In [8], the location of target is hierarchically inferred from the output of 3 sets of different correlation filters. However, in CoKCF_CNN, the response can be easily obtained by only 2 sets of correlation filters.

10.3.2 Attribute-based Evaluation

Similarly, in this section we give the overall precision plot for each challenging attribute [147] in Fig 10.1 and Fig 10.2. We can see the proposed method, with simpler concept and less computational complexity, can perform competitively against the state-of-the-art method in all cases.

10.4 Conclusion

Robustness and speed have been regarded as a dilemma in visual tracking system for a long time. On one hand, single classifier may not be powerful enough to handle complex backgroundings and significant variations of target appearance, and ensemble methods seem to be a promising approach to significantly improve the tracking performance [118]. On the other hand, modern trackers suffer from relatively large computational time, which is a major hurdle that hinders the ensemble based visual tracking.

Figure 10.1 The precision plots of the results.

Figure 10.2 The precision plots of the results (continued from Figure 10.1).

This work extends the recently developed KCF tracker and makes a good trade-off between the robustness and speed. It presents a novel and effective joint training approach for ensemble tracking. We choose CoKCF, which is a simple realization of the proposed method with only 2 base trackers, to demonstrate our point. We demonstrate the effectiveness of the proposed method with handcrafted features and features extracted from ConvNets.

Summary of Part II

In the second part of the thesis, we apply the concepts in Part I to solve visual tracking problems, which is a fundamental task in computer vision with applications such as video surveillance, human computer interaction, medical imaging and so on. Visual tracking can be posed as an online classification task. In this thesis, we focus on the *one – pass model – free single object tracking* setting, which is the most fundamental case. More specifically, without any appearance model, it only provides the bounding box of one single object in the first frame. Given this single (labeled) instance, the goal is to predict the location of the object in an online manner.

In Chapter 8, we propose an incremental oblique random forest tracker. In each node of the decision tree, we employ a proximal support vector machine to partition the data into 2 groups. We also address how to incrementally learn the oblique random forest classifier, which corresponds to the updating process of the visual tracking. We evaluate the proposed RaF tracker on a recently proposed benchmark which consists of 51 challenging video sequences and compare 29 state-of-the-art trackers published in Top Venues (CVPR/ICCV, TPAMI, etc). Experimental results show that the proposed methods achieved the best overall performance.

Recently, deep learning based methods achieved astonishing performance in visual tracking. However, almost all existing deep learning based trackers rely heavily on conventional back-prop to train and refine the model. They firstly train the deep models on some auxiliary dataset such as Imagenet. Then the pre-trained models are finetuned during the tracking process. This fine tuning process make the pretrained model to adapt to the object being tracked. However, the tracking performance is sensitive to hyper-parameters during fine-tuning such as learning rate, maximum number of epoches, size of the mini-batch and so on. In Chapter 9, we addressed these issues in deep learning based tracker by (i) randomly

generating the parameters in convolutional layer of CNN and kept them fixed; (ii) proposing an efficient and effective updating rule for the classifiers in CNN. Our approach, named as convolutional random vector functional link network (CRVFL), can be regarded as a marriage of the random vector functional link network (RVFL) and CNN. It is much simpler and more efficient to train and performs favorably against state-of-the-art methods.

Recent advances in visual tracking have witnessed the importance of discriminative classifiers tasked with distinguishing the target from the background. However, a single classifier may fail to cope with complex surrounding environment and large appearance variations of the target. In Chapter 10, we apply the Co-KRR method in Chapter 5 and propose a framework to train a pool of discriminative classifiers jointly in a closed-form fashion. It poses an extra regularization term in ridge regression which interacts with other base trackers in the ensemble. Through a simple realization of this approach, we show co-trained kernelized correlation filters (COKCF) which consist of two KCF trackers, are able to outperform the KCF tracker by a larger margin and perform favorably against other state-of-the-art trackers on 51 benchmark video sequences.

Our observation is in line with the conclusions in the computer vision community: features from ConvNets are effective and outperforms conventional handcrafted features. Though the proposed RaF tracker in Chapter 8 achieves the top performance with the overall score of 74.1%, it is far from satisfactory compared with the trackers based on features from ConvNets. However, feature extraction from ConvNets are time consuming and may heavily depend on advanced GPU hardware. From this point of view, RaF naturally fit distributed or parallel computing environment, which makes it also suitable for real life applications.

Recommendation for Future Work

The research on pattern classification will be more demanding in the future. The data size and dimensions of the real world applications will increase dramatically and therefore effective and efficient classification methods need to be investigated. In general, several topics deserving more research:

1. Feature learning for ensemble methods: Generally speaking, Feature selection can be addressed into two different group of methods . The first one is model-dependent ("wrapper" and "embedded" methods). Wrapper approaches search the space of feature subsets, usually in an exhaustive manner. They evaluate subsets of variables which allows, unlike filter approaches, to detect the possible interactions between variable. The utility for a candidate subset can be obtained via the training/validation accuracy of a particular learning model. This may deliver significant advantages in generalisation, however, at the expense of a significant larger computational complexity. Moreover, it may produce subsets that are overly specific to the learning model used. As a result, any change in the learning model is likely to render the feature set suboptimal. Embedded methods, on the other hand, usually exploit the structure of specific classes of learning models to guide the feature selection process. While the defining component of a wrapper method is simply the search procedure, the defining component of an embedded method is a criterion derived through fundamental knowledge of a specific class of functions. Thus, we plan to investigate how to jointly optimized feature learning and classification in the ensemble framework based on embedded feature learning methods.
2. Large-Scale ensemble learning: "Big Data" [214] has attracted considerable attention recently. Existing approaches for "Big Data" is either based on very deep structure or extend current methods to large scale online learning. Hence, it is worthy to investigate

Chapter 12. Recommendation for Future Work

the benefits of both Large-Scale and deep structure ensemble learning for solving big data problems.

3. Combination of ensemble methods and deep learning: It is straightforward to see that ensemble methods consistently improve the single deep model in large-scale computer vision problems [212]. However, those approach usually employ simple average. Thus, how to develop an effective framework to combine ensemble learning and deep learning deserves more research. Can we design better model to combine the results from several deep models in a joint framework?
4. Applications of ensemble learning: In this thesis we only consider visual tracking to demonstrate the feasibility of the proposed ensemble methods. In the future, we plan to investigate the performance of ensemble methods on other challenging tasks such as object detection, face recognition and so on.

List of Publications

Journal Articles:

- **L. Zhang**, P. N. Suganthan, “Random forests with ensemble of feature spaces,” in *Pattern Recognition*, vol. 47, no. 10, pp. 3429–3437, 2014.
- **L. Zhang**, P. N. Suganthan, “Oblique Decision Tree Ensemble via Multisurface Proximal Support Vector Machine,” in *IEEE Transaction on Cybernetics*, vol.45, no.10, pp.2165-2176, 2015.
- **L. Zhang***, Y. Ren*, and P. N. Suganthan, “Ensemble Classification and Regression – Recent Developments, Applications and Future Directions,” *IEEE Computational Intelligence Magazine*, vol. 11, no. 1, pp. 41-53, Feb. 2016 (* indicates equal contribution).
- **L. Zhang**, P. N. Suganthan, “A comprehensive evaluation of random vector functional link networks,” *Information Sciences, September, 2015. ISSN 0020-0255*.
- **L. Zhang**, P. N. Suganthan, “A Survey of Randomized Algorithms for Training Neural Networks,” *Information Sciences*, 2016 (Accepted).
- **L. Zhang**, P. N. Suganthan, “Convolutional Random Vector Functional Link Neural Network,” *IEEE Transaction on Cybernetics*, (Accepted).
- **L. Zhang**, P. N. Suganthan, “Role of sparsity: A case study with visual tracking,” *Experts Systems with Applications*, (Submitted).
- **L. Zhang**, P. N. Suganthan, “Robust Visual Tracking via Co-trained Kernelized Correlation Filters,” *IEEE Transaction Circuits and Systems for Video Technology*, (Submitted).

- **L. Zhang**, Jagannadan Varadarajan, P.N. Suganthan, Pierre Moulin, “Robust Visual tracking with incremental oblique random forest,” *International Journal of Computer Vision*, (Submitted).

Conference Papers:

- **L. Zhang**, Y. Ren and P. N. Suganthan, “Instance based random forest with rotated feature space,” in: *Proc. IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL’13)*, Singapore, Apr. 2013.
- Y. Ren, **L. Zhang**, P. N. Suganthan, “K-nearest neighbor based bagging SVM pruning,” in: *Proc. IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL’13)*, Singapore, Apr. 2013.
- **L. Zhang**, Y. Ren, P. N. Suganthan, “Towards Generating Random Forests via Extremely Randomized Trees,” in *Proc. IEEE World Congress on Computational Intelligence (WCCI’14)*, Beijing, Jul. 2014.
- X. Qiu, **L. Zhang**, Y. Ren, P. N. Suganthan and G. Amaratunga, “Ensemble Deep Learning for Regression and TimeSeries Forecasting,” in: *Proc. IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL’14)*, Orlando, US, Dec. 2014.
- **L. Zhang**, P. N. Suganthan, “Visual Tracking With Convolutional Neural Network,” in: *IEEE International Conference On Systems, Man and Cybernetics (SMC’2015)*, HK, Oct. 2015.

Bibliography

- [1] Ye Ren, Le Zhang, and P. N. Suganthan. Ensemble classification and regression – recent developments, applications and future directions. *IEEE Computational Intelligence Magazine*, doi: 10.1109/MCI.2015.2471235.
- [2] Thomas G Dietterich. Ensemble methods in machine learning. In *Multiple classifier systems*, pages 1–15. Springer, 2000.
- [3] Ron Kohavi, David H Wolpert, et al. Bias plus variance decomposition for zero-one loss functions. In *ICML*, pages 275–283, 1996.
- [4] Robert E Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of statistics*, pages 1651–1686, 1998.
- [5] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [6] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15(1):3133–3181, 2014.
- [7] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.
- [8] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. In *The IEEE International Conference on Computer Vision*, December 2015.
- [9] Zhi-Hua Zhou, Fabio Roli, and Josef Kittler. *Multiple Classifier Systems: 11th International Workshop, MCS 2013, Nanjing, China, May 15-17, 2013. Proceedings*. Springer, 2013.
- [10] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.
- [11] Robert E. Schapire and Yoav Freund. Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26:322–330, 1998.
- [12] Yi Lin and Yongho Jeon. Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, 101(474):578–590, 2006.

Bibliography

- [13] Hongying Jiang, Youping Deng, Huann-Sheng Chen, Lin Tao, Qiuying Sha, Jun Chen, Chung-Jui Tsai, and Shuanglin Zhang. Joint analysis of two microarray gene-expression data sets to select lung adenocarcinoma marker genes. *BMC bioinformatics*, 5(1):81, 2004.
- [14] Kai-Quan Shen, Chong-Jin Ong, Xiao-Ping Li, Zheng Hui, and Einar PV Wilder-Smith. A feature selection method for multilevel mental fatigue eeg classification. *IEEE Transactions on Biomedical Engineering*, 54(7):1231–1237, 2007.
- [15] M. Lichman. UCI machine learning repository, 2013. URL <http://archive.ics.uci.edu/ml>.
- [16] Naresh Manwani and PS Sastry. Geometric decision tree. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 42(1):181–192, 2012.
- [17] Leo Breiman. *Classification and regression trees*. CRC press, 1993.
- [18] Sreerama K Murthy, Simon Kasif, Steven Salzberg, and Richard Beigel. OC1: A randomized algorithm for building oblique decision trees. In *Proceedings of AAAI*, volume 93, pages 322–327. Citeseer, 1993.
- [19] Witold Pedrycz and Zenon A Sosnowski. Genetically optimized fuzzy decision trees. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 35(3):633–641, 2005.
- [20] Sung-Hyuk Cha and Charles Tappert. A genetic algorithm for constructing compact binary decision trees. *Journal of Pattern Recognition Research*, 4(1):1–13, 2009.
- [21] Erick Cantu-Paz and Chandrika Kamath. Inducing oblique decision trees with evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 7(1):54–68, 2003.
- [22] Jose Maria Pangilinan and Gerrit K Janssens. Pareto-optimality of oblique decision trees from evolutionary algorithms. *Journal of Global Optimization*, 51(2):301–311, 2011.
- [23] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.
- [24] Le Zhang, Ye Ren, and P.N. Suganthan. Towards generating random forests via extremely randomized trees. In *2014 International Joint Conference on Neural Networks*, *IEEE*, pages 2645–2652, 2014.
- [25] Xi-Zhao Wang, Jun-Hai Zhai, and Shu-Xia Lu. Induction of multiple fuzzy decision trees based on rough set technique. *Information Sciences*, 178(16):3188–3202, 2008.
- [26] Xi-Zhao Wang and Chun-Ru Dong. Improving generalization of fuzzy if–then rules by maximizing fuzzy entropy. *IEEE Transactions on Fuzzy Systems*, 17(3):556–567, 2009.
- [27] Le Zhang, , and P.N. Suganthan. Random forests with ensemble of feature spaces. *Pattern Recognition*, 47(10):3429 – 3437, 2014.
- [28] Li Zhang, Wei-Da Zhou, Tian-Tian Su, and Li-Cheng Jiao. Decision tree support vector machine. *International Journal on Artificial Intelligence Tools*, 16(01):1–15, 2007.

-
- [29] Olvi L Mangasarian and Edward W Wild. Multisurface proximal support vector machine classification via generalized eigenvalues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):69–74, 2006.
 - [30] Li-Fen Chen, Hong-Yuan Mark Liao, Ming-Tat Ko, Ja-Chen Lin, and Gwo-Jong Yu. A new LDA-based face recognition system which can solve the small sample size problem. *Pattern recognition*, 33(10):1713–1726, 2000.
 - [31] Xudong Jiang. Linear subspace learning-based dimensionality reduction. *IEEE Signal Processing Magazine*, 28(2):16–26, 2011.
 - [32] Leo Breiman. Bias, variance, and arcing classifiers. 1996.
 - [33] Peter Büchmann and Bin Yu. Analyzing bagging. *Annals of Statistics*, pages 927–961, 2002.
 - [34] Pedro Domingos. Why does bagging work? a bayesian account and its implications. In *KDD*, pages 155–158. Citeseer, 1997.
 - [35] Torsten Hothorn and Berthold Lausen. Double-bagging: Combining classifiers by bootstrap aggregation. *Pattern Recognition*, 36(6):1303–1309, 2003.
 - [36] Thomas G Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157, 2000.
 - [37] Tao Chen and Jianghong Ren. Bagging for gaussian process regression. *Neurocomputing*, 72(7):1605–1610, 2009.
 - [38] Marina Skurichina and Robert PW Duin. Bagging for linear classifiers. *Pattern Recognition*, 31(7):909–930, 1998.
 - [39] Gonzalo Martínez-Muñoz and Alberto Suárez. Pruning in ordered bagging ensembles. In *Proceedings of the 23rd international conference on Machine learning*, pages 609–616. ACM, 2006.
 - [40] G Martinez-Muoz, Daniel Hernández-Lobato, and Almudena Suarez. An analysis of ensemble pruning techniques based on ordered aggregation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):245–259, 2009.
 - [41] Gonzalo Martínez-Muñoz and Alberto Suárez. Using boosting to prune bagging ensembles. *Pattern Recognition Letters*, 28(1):156–165, 2007.
 - [42] Aleksandar Lazarevic and Vipin Kumar. Feature bagging for outlier detection. In *Proceedings of the eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 157–166. ACM, 2005.
 - [43] Yasser Ganjisaffar, Rich Caruana, and Cristina Videira Lopes. Bagging gradient-boosted trees for high precision, low variance ranking models. In *Proceedings of the 34th international ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 85–94. ACM, 2011.
 - [44] Robert E Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.
 - [45] Gunnar Rätsch, Takashi Onoda, and K-R Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001.

Bibliography

- [46] Alexander Vezhnevets and Vladimir Vezhnevets. Modest adaboost-teaching adaboost to generalize better. In *Graphicon*, volume 12, pages 987–997, 2005.
- [47] Shai Avidan. SpatialBoost: Adding spatial reasoning to adaboost. In *Proc. Computer Vision (ECCV'06)*, pages 386–396, 2006.
- [48] Leo Breiman. Arcing classifier (with discussion and a rejoinder by the author). *The annals of statistics*, 26(3):801–849, 1998.
- [49] Carlos Domingo and Osamu Watanabe. MadaBoost: A modification of AdaBoost. In *Proc. Annual Conference on Computational Learning Theory (COLT'00)*, pages 180–189, 2000.
- [50] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The Annals of Statistics*, 28(2):337–407, 2000.
- [51] Leo Breiman. Pasting small votes for classification in large databases and on-line. *Machine Learning*, 36(1-2):85–103, 1999.
- [52] David H Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.
- [53] Li Bo, Li Xinjun, and Zhao Zhiyan. Novel algorithm for constructing support vector machine regression ensemble. *Journal of Systems Engineering and Electronics*, 17(3): 541–545, 2006.
- [54] Kesheng Lu and Lingzhi Wang. A novel nonlinear combination model based on support vector machine for rainfall prediction. In *Proc. IEEE International Joint Conference on Computational Sciences and Optimization (CSO'11)*, pages 1343–1346, 2011.
- [55] Lingzhi Wang and Jiansheng Wu. Application of hybrid RBF neural network ensemble model based on wavelet support vector machine regression in rainfall time series forecasting. In *Proc. IEEE International Joint Conference on Computational Sciences and Optimization (CSO'12)*, pages 867–871, 2012.
- [56] Jui-Sheng Chou and Anh-Duc Pham. Enhanced artificial intelligence for ensemble approach to predicting high performance concrete compressive strength. *Construction and Building Materials*, 49:554–563, December 2013.
- [57] Jiansheng Wu. A novel artificial neural network ensemble model based on k-nearest neighbor nonparametric estimation of regression function and its application for rainfall forecasting. In *Proc. International Joint Conference on Computational Sciences and Optimization (CSO'09)*, volume 2, pages 44–48, 2009.
- [58] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.
- [59] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on Systems, Man, and Cybernetics*, 21(3):660–674, 1991.
- [60] John Mingers. An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, 3(4):319–342, 1989.
- [61] Lior Rokach and Oded Maimon. Top-down induction of decision trees classifiers-a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 35(4):476–487, 2005.

-
- [62] John Mingers. Expert systems-rule induction with statistical data. *Journal of the operational research society*, pages 39–47, 1987.
- [63] J. Ross Quinlan. Simplifying decision trees. *International Journal of Man-machine Studies*, 27(3):221–234, 1987.
- [64] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.
- [65] Bjoern H Menze, B Michael Kelm, Ralf Masuch, Uwe Himmelreich, Peter Bachert, Wolfgang Petrich, and Fred A Hamprecht. A comparison of random forest and its gini importance with standard chemometric methods for the feature selection and classification of spectral data. *BMC bioinformatics*, 10(1):213, 2009.
- [66] Trevor. Hastie, Robert. Tibshirani, and J Jerome H Friedman. *The elements of statistical learning*, volume 1. Springer New York, 2001.
- [67] Torsten Hothorn, Friedrich Leisch, Achim Zeileis, and Kurt Hornik. The design and analysis of benchmark experiments. *Journal of Computational and Graphical Statistics*, 14(3), 2005.
- [68] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.
- [69] Bjoern H Menze, Wolfgang Petrich, and Fred A Hamprecht. Multivariate feature selection and hierarchical classification for infrared spectroscopy: serum-based detection of bovine spongiform encephalopathy. *Analytical and bioanalytical chemistry*, 387(5):1801–1807, 2007.
- [70] Yunming Ye, Qingyao Wu, Joshua Zhexue Huang, Michael K Ng, and Xutao Li. Stratified sampling for feature subspace selection in random forests for high dimensional data. *Pattern Recognition*, 46(3):769–787, 2013.
- [71] Le Zhang, Ye Ren, and P. N. Suganthan. Instance based random forest with rotated feature space. In *Proc. IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL'13)*, pages 31–35, 2013.
- [72] Bjoern H Menze, B Michael Kelm, Daniel N Splitthoff, Ullrich Koethe, and Fred A Hamprecht. On oblique random forests. In *Machine Learning and Knowledge Discovery in Databases*, pages 453–469. 2011.
- [73] Le Zhang and P.N. Suganthan. Random forests with ensemble of feature spaces. *Pattern Recognition*, 47(10):3429–3437, October 2014.
- [74] Le Zhang and P. N. Suganthan. Oblique decision tree ensemble via multisurface proximal support vector machine. *IEEE Transaction on Cybernetics*, pages 2168–2267, November 2014.
- [75] Juan José Rodríguez, Ludmila I Kuncheva, and Carlos J Alonso. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, 2006.
- [76] Kun-Hong Liu and De-Shuang Huang. Cancer classification using rotation forest. *Computers in Biology and Medicine*, 38(5):601–610, 2008.

Bibliography

- [77] Akin Ozcift. Svm feature selection based rotation forest ensemble classifiers to improve computer-aided diagnosis of parkinson disease. *Journal of medical systems*, 36(4):2141–2147, 2012.
- [78] Jun-Feng Xia, Kyungsook Han, and De-Shuang Huang. Sequence-based prediction of protein-protein interactions by means of rotation forest and autocorrelation descriptor. *Protein and Peptide Letters*, 17(1):137–145, 2010.
- [79] Gregor Stiglic and Peter Kokol. Effectiveness of rotation forest in meta-learning based gene expression classification. In *Twentieth IEEE International Symposium on Computer-Based Medical Systems, IEEE*, pages 243–250, 2007.
- [80] Chun-Xia Zhang and Jiang-She Zhang. Rotboost: A technique for combining rotation forest and adaboost. *Pattern Recognition Letters*, 29(10):1524–1536, 2008.
- [81] Koen W De Bock and Dirk Van den Poel. An empirical evaluation of rotation-based ensemble classifiers for customer churn prediction. *Expert Systems with Applications*, 38(10):12293–12301, 2011.
- [82] Ludmila I Kuncheva and Juan José Rodríguez. Classifier ensembles with a random linear oracle. *IEEE Transactions on Knowledge and Data Engineering*, 19(4):500–508, 2007.
- [83] Simon Haykin and Neural Network. *Neural Networks: A comprehensive foundation. Neural Networks*, 2(2004), 2004.
- [84] Robert A Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1(4):295–307, 1988.
- [85] Yoh-Han Pao. *Adaptive Pattern Recognition and Neural Networks*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989. ISBN 0-201-12584-6.
- [86] Marco Gori and Alberto Tesi. On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):76–86, 1992.
- [87] Wouter F Schmidt, Martin A Kraaijveld, and Robert PW Duin. Feedforward neural networks with random weights. In *11th IAPR International Conference on Pattern Recognition, IEEE*, pages 1–4, 1992.
- [88] Hugues Berry and Mathias Quoy. Structure and dynamics of random recurrent neural networks. *Adaptive Behavior*, 14(2):129–137, 2006.
- [89] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
- [90] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [91] Le Zhang and P. N. Suganthan. A survey of randomized algorithms for training neural networks. *Information Sciences*, 2016. Accepted.
- [92] Yoh-Han Pao and Y Takefuji. Functional-link net computing. *IEEE Computer*, 25(5): 76–79, 1992.
- [93] Boris Igelnik and Y-H Pao. Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *IEEE Trans on Neural Networks*, 6(6): 1320–1329, 1995.

-
- [94] Yoh-Han Pao and Stephen M Phillips. The functional link net and learning optimal control. *Neurocomputing*, 9(2):149–164, 1995.
- [95] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.
- [96] Craig Saunders, Alexander Gammerman, and Volodya Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning, Morgan Kaufmann*, pages 515–521, 1998.
- [97] Robert E Banfield, Lawrence O Hall, Kevin W Bowyer, and W Philip Kegelmeyer. A comparison of decision tree ensemble creation techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):173–180, 2007.
- [98] Kolluru Venkata Sreerama Murthy and Steven L Salzberg. *On growing better decision trees from data*. PhD thesis, Citeseer, 1995.
- [99] Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, Lawrence D Jackel, Yann LeCun, Urs A Muller, Edward Sackinger, Patrice Simard, et al. Comparison of classifier methods: a case study in handwritten digit recognition. In *International Conference on Pattern Recognition, IEEE Computer Society Press*, pages 77–77, 1994.
- [100] Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In *Neurocomputing*, pages 41–50. 1990.
- [101] John C Platt, Nello Cristianini, and John Shawe-Taylor. Large margin DAGs for multiclass classification,. In *Advances in Neural Information Processing Systems*, volume 12, pages 547–553, 1999.
- [102] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *arXiv preprint cs/9501101*, 1995.
- [103] Xudong Jiang, Bappaditya Mandal, and Alex Kot. Eigenfeature regularization and extraction in face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(3):383–394, 2008.
- [104] Jose Marroquin, Sanjoy Mitter, and Tomaso Poggio. Probabilistic solution of ill-posed problems in computational vision. *Journal of the American Statistical Association*, 82 (397):76–89, 1987.
- [105] Olvi L Mangasarian and RR Meyer. Nonlinear perturbation of linear programs. *SIAM Journal on Control and Optimization*, 17(6):745–752, 1979.
- [106] Theodoros Evgeniou, Massimiliano Pontil, and Tomaso Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13 (1):1–50, 2000.
- [107] Le Zhang and P.N. Suganthan. Oblique decision tree ensemble via multisurface proximal support vector machine. *IEEE Transactions on Cybernetics*, 45(10):2165–2176, Oct 2015. ISSN 2168-2267. doi: 10.1109/TCYB.2014.2366468.
- [108] Krishna Kumar Kandaswamy, Kuo-Chen Chou, Thomas Martinetz, Steffen Möller, P.N. Suganthan, S Sridharan, and Ganesan Pugalenth. Afp-pred: A random forest approach for predicting antifreeze proteins from sequence-derived properties. *Journal of theoretical biology*, 270(1):56–62, 2011.

Bibliography

- [109] Krishna Kandaswamy, Ganesan Pugalenthi, Mehrnaz Hazrati, Kai-Uwe Kalies, and Thomas Martinetz. Blprot: prediction of bioluminescent proteins based on support vector machine and relief feature selection. *BMC bioinformatics*, 12(1):345, 2011.
- [110] Xiaofei He, Shuicheng Yan, Yuxiao Hu, Partha Niyogi, and Hong-Jiang Zhang. Face recognition using laplacianfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):328–340, 2005.
- [111] Deng Cai, Xiaofei He, Jiawei Han, and Hong-Jiang Zhang. Orthogonal laplacianfaces for face recognition. *IEEE Transactions on Image Processing*, 15(11):3608–3614, 2006.
- [112] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.
- [113] P. Nemenyi. *Distribution-free Multiple Comparisons*. Princeton University, 1963. URL <http://books.google.com.sg/books?id=nhDMtgAACAAJ>.
- [114] David J Sheskin. *Handbook of parametric and nonparametric statistical procedures*. crc Press, 2003.
- [115] Michael J Kearns and Umesh Virkumar Vazirani. *An introduction to computational learning theory*. MIT press, 1994.
- [116] Senjian An, Wanquan Liu, and Svetha Venkatesh. Face recognition using kernel ridge regression. In *2007 IEEE Conference on Computer Vision and Pattern Recognition, IEEE*, pages 1–7, 2007.
- [117] Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. A co-regularization approach to semi-supervised learning with multiple views. In *Proceedings of ICML workshop on learning with multiple views*, pages 74–79. Citeseer, 2005.
- [118] Naiyan Wang, Jianping Shi, Dit-Yan Yeung, and Jiaya Jia. Understanding and diagnosing visual tracking systems. *arXiv preprint arXiv:1504.06055*, 2015.
- [119] Michael J Black and Allan D Jepson. Eigenttracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.
- [120] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3): 125–141, 2008.
- [121] Amit Adam, Ehud Rivlin, and Ilan Shimshoni. Robust fragments-based tracking using the integral histogram. In *2006 IEEE Computer Society Conference on Computer vision and pattern recognition, IEEE*, volume 1, pages 798–805, 2006.
- [122] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003.
- [123] Junseok Kwon and Kyoung Mu Lee. Visual tracking decomposition. In *2010 IEEE Conference on Computer Vision and Pattern Recognition, IEEE*, pages 1269–1276, 2010.
- [124] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

- [125] Iain Matthews, Takahiro Ishikawa, and Simon Baker. The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):810–815, 2004.
- [126] Laura Sevilla-Lara and Erik Learned-Miller. Distribution fields for tracking. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE*, pages 1910–1917, 2012.
- [127] Shai Avidan. Ensemble tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):261–271, 2007.
- [128] Helmut Grabner and Horst Bischof. On-line boosting and vision. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE*, volume 1, pages 260–267, 2006.
- [129] Helmut Grabner, Christian Leistner, and Horst Bischof. Semi-supervised on-line boosting for robust tracking. In *Computer Vision–ECCV 2008*, pages 234–247. Springer, 2008.
- [130] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1619–1632, 2011.
- [131] Ziyang Xiao, Huchuan Lu, and Dong Wang. L2-rls-based object tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(8):1301–1309, 2014.
- [132] Kaihua Zhang, Lei Zhang, and Ming-Hsuan Yang. Real-time object tracking via online discriminative feature selection. *IEEE Transactions on Image Processing*, 22(12):4664–4677, 2013.
- [133] Qing Wang, Feng Chen, Wenli Xu, and Ming-Hsuan Yang. Object tracking via partial least squares analysis. *IEEE Transactions on Image Processing*, 21(10):4454–4465, 2012.
- [134] Wei Zhong, Huchuan Lu, and Ming-Hsuan Yang. Robust object tracking via sparse collaborative appearance model. *IEEE Transactions on Image Processing*, 23(5):2356–2368, 2014.
- [135] Baiyang Liu, Junzhou Huang, Lin Yang, and Casimir Kulikowsk. Robust tracking using local sparse appearance model and k-selection. In *2011 IEEE Conference on Computer Vision and Pattern Recognition, IEEE*, pages 1313–1320, 2011.
- [136] Baiyang Liu, Lin Yang, Junzhou Huang, Peter Meer, Leiguang Gong, and Casimir Kulikowski. Robust and fast collaborative tracking with two stage sparse optimization. In *Computer Vision–ECCV 2010*, pages 624–637. Springer, 2010.
- [137] Xue Mei and Haibin Ling. Robust visual tracking using l1 minimization. In *2009 IEEE 12th International Conference on Computer Vision, IEEE*, pages 1436–1443, 2009.
- [138] Xue Mei, Haibin Ling, Yi Wu, Erik Blasch, and Li Bai. Minimum error bounded efficient l1 tracker with occlusion detection. In *2011 IEEE Conference on Computer Vision and Pattern Recognition, IEEE*, pages 1257–1264, 2011.
- [139] Dong Wang, Huchuan Lu, and Ming-Hsuan Yang. Online object tracking with sparse prototypes. *IEEE Transactions on Image Processing*, 22(1):314–325, 2013.

Bibliography

- [140] Fan Yang, Huchuan Lu, and Ming-Hsuan Yang. Robust superpixel tracking. *IEEE Transactions on Image Processing*, 23(4):1639–1651, 2014.
- [141] James Supancic and Deva Ramanan. Self-paced learning for long-term tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2379–2386, 2013.
- [142] Le Zhang and Suganthan P.N. Visual tracking with convolutional neural network. in 2015 IEEE International Conference On System, Man and Cybernetics.
- [143] Naiyan Wang and Dit-Yan Yeung. Learning a deep compact image representation for visual tracking. In *Advances in Neural Information Processing Systems*, pages 809–817, 2013.
- [144] Naiyan Wang, Siyi Li, Abhinav Gupta, and Dit-Yan Yeung. Transferring rich feature hierarchies for robust visual tracking. *arXiv preprint arXiv:1501.04587*, 2015.
- [145] Qing Wang, Feng Chen, Jimei Yang, Wenli Xu, and Ming-Hsuan Yang. Transferring visual prior for online object tracking. *IEEE Transactions on Image Processing*, 21(7): 3296–3305, 2012.
- [146] Le Zhang and P.N. Suganthan. Visual tracking with convolutional random vector functional link networks. submitted for publication.
- [147] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *2013 IEEE Conference on Computer vision and pattern recognition, IEEE*, pages 2411–2418, 2013.
- [148] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.
- [149] Le Zhang and P.N. Suganthan. Oblique decision tree ensemble via multisurface proximal support vector machine., *IEEE Transactions on Cybernetics*, doi=10.1109/TCYB.2014.2366468,, 45(10):2165–2176, Oct 2015. ISSN 2168-2267.
- [150] Amir Saffari, Christian Leistner, Jakob Santner, Martin Godec, and Horst Bischof. On-line random forests. In *2009 IEEE 12th International Conference on Computer Vision Workshops*, pages 1393–1400, 2009.
- [151] Qinxun Bai, Zheng Wu, Stan Sclaroff, Margrit Betke, and Camille Monnier. Randomized ensemble tracking. In *2013 IEEE International Conference on Computer Vision, IEEE*, pages 2040–2047, 2013.
- [152] Junseok Kwon and Kyoung Mu Lee. Tracking by sampling and integrating multiple trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7): 1428–1441, 2014.
- [153] Huchuan Lu, Shipeng Lu, Dong Wang, Shu Wang, and Henry Leung. Pixel-wise spatial pyramid-based hybrid tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(9):1365–1376, 2012.
- [154] Naiyan Wang and Dit-Yan Yeung. Ensemble-based tracking: Aggregating crowd-sourced structured time series data. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1107–1115, 2014.

- [155] Naiyan Wang, Jianping Shi, Dit-Yan Yeung, and Jiaya Jia. Understanding and diagnosing visual tracking systems. In *The IEEE International Conference on Computer Vision*, December 2015.
- [156] Olvi L Mangasarian and Edward W Wild. Proximal support vector machine classifiers. In *Proceedings KDD-2001: Knowledge Discovery and Data Mining*, 2001.
- [157] Sam Hare, Amir Saffari, and Philip HS Torr. Struck: Structured output tracking with kernels. In *2011 IEEE International Conference on Computer Vision, IEEE*, pages 263–270, 2011.
- [158] Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008.
- [159] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE*, pages 248–255, 2009.
- [160] Hanxi Li, Yi Li, Fatih Porikli, et al. Deeptrack: Learning discriminative feature representations by convolutional neural networks for visual tracking. In *Proceedings of the British Machine Vision Conference, BMVA press*, volume 1, page 3, 2014.
- [161] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.
- [162] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE*, volume 1, pages 886–893, 2005.
- [163] Yoshua Bengio, Aaron Courville, and Pierre Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.
- [164] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [165] Ossama Abdel-Hamid, Abdel-Rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(10):1533–1545, 2014.
- [166] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.
- [167] Jianming Zhang, Shugao Ma, and Stan Sclaroff. MEEM: Robust tracking via multiple experts using entropy minimization. In *Computer Vision—ECCV 2014, Springer*, pages 188–203, 2014.
- [168] Horst Possegger, Thomas Mauthner, and Horst Bischof. In defense of color-based model-free tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2113–2120, 2015.
- [169] Li Wang, Ting Liu, Gang Wang, Kap Luk Chan, and Qingxiong Yang. Video tracking using learned hierarchical features. *IEEE Transactions on Image Processing*, 24(4): 1424–1435, 2015.

Bibliography

- [170] Seunghoon Hong, Tackgeun You, Suha Kwak, and Bohyung Han. Online tracking by learning discriminative saliency map with convolutional neural network. *arXiv preprint arXiv:1502.06796*, 2015.
- [171] Kaihua Zhang, Qingshan Liu, Yi Wu, and Ming-Hsuan Yang. Robust tracking via convolutional networks without learning. *arXiv preprint arXiv:1501.04505*, 2015.
- [172] Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma. Pcanet: A simple deep learning baseline for image classification? *arXiv preprint arXiv:1404.3606*, 2014.
- [173] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10), 1995.
- [174] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, and Gerald Penn. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, pages 4277–4280, 2012.
- [175] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.
- [176] Jose M Alvarez, Theo Gevers, Yann LeCun, and Antonio M Lopez. Road scene segmentation from a single image. In *Computer Vision—ECCV 2012*, pages 376–389. Springer, 2012.
- [177] Jiquan Ngiam, Zhenghao Chen, Daniel Chia, Pang W Koh, Quoc V Le, and Andrew Y Ng. Tiled convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1279–1287, 2010.
- [178] Matthew D Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *2013 International Conference on Learning Representations*, 2013.
- [179] Jiashi Feng, Bingbing Ni, Qi Tian, and Shuicheng Yan. Geometric ℓ p-norm feature pooling for image classification. In *2011 IEEE Conference on Computer Vision and Pattern Recognition, IEEE*, pages 2609–2704, 2011.
- [180] Kevin Jarrett, Koray Kavukcuoglu, M Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision, IEEE*, pages 2146–2153, 2009.
- [181] Andrew Saxe, Pang W Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Y Ng. On random weights and unsupervised feature learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1089–1096, 2011.
- [182] Le Zhang and P.N. Suganthan. A comprehensive evaluation of random vector functional link networks. *Information Sciences*, doi: <http://dx.doi.org/10.1016/j.ins.2015.09.025>, 2015.
- [183] Thomas Kailath, Ali H Sayed, and Babak Hassibi. *Linear estimation*, volume 1. Prentice Hall Upper Saddle River, NJ, 2000.
- [184] Simon S Haykin. *Adaptive Filter Theory*. Pearson Education India, 2008.

-
- [185] Louis L Scharf. *Statistical Signal Processing*, volume 98. Addison-Wesley Reading, MA, 1991.
 - [186] Edwin KP Chong and Stanislaw H Zak. *An introduction to optimization*, volume 76. John Wiley & Sons, 2013.
 - [187] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *Computer Vision–ECCV 2012*, Springer, pages 702–715. 2012.
 - [188] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536, 2004.
 - [189] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012.
 - [190] Thang Ba Dinh, Nam Vo, and Gérard Medioni. Context tracker: Exploring supporters and distracters in unconstrained environments. In *2011 IEEE Conference on Computer Vision and Pattern Recognition, IEEE*, pages 1177–1184, 2011.
 - [191] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.
 - [192] David S Bolme, J Ross Beveridge, Bruce Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *IEEE Conference on Computer Vision and Pattern Recognition, IEEE*, pages 2544–2550, 2010.
 - [193] Robert M Gray. *Toeplitz and Circulant Matrices: A Review*. Now Publishers Inc, 2006.
 - [194] Ryan Rifkin, Gene Yeo, and Tomaso Poggio. Regularized least-squares classification. *Nato Science Series Sub Series III Computer and Systems Sciences*, 190:131–154, 2003.
 - [195] Ting Liu, Gang Wang, and Qingxiong Yang. Real-time part-based visual tracking via adaptive correlation filters. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4902–4912, June 2015.
 - [196] Yang Li, Jianke Zhu, and Steven CH Hoi. Reliable patch trackers: Robust visual tracking by exploiting reliable patches. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE*, pages 353–361, 2015.
 - [197] Kaihua Zhang, Lei Zhang, Qingshan Liu, David Zhang, and Ming-Hsuan Yang. Fast visual tracking via dense spatio-temporal context learning. In *Computer Vision–ECCV 2014*, pages 127–141. Springer, 2014.
 - [198] Martin Danelljan, Gustav Häger, Fahad Khan, and Michael Felsberg. Accurate scale estimation for robust visual tracking. In *British Machine Vision Conference, Nottingham, September 1-5, 2014, BMVA Press*, 2014.
 - [199] Ming Tang and Jiayi Feng. Multi-kernel correlation filter for visual tracking. In *The IEEE International Conference on Computer Vision*, December 2015.

Bibliography

- [200] Chao Ma, Xiaokang Yang, Chongyang Zhang, and Ming-Hsuan Yang. Long-term correlation tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5388–5396, 2015.
- [201] Zhibin Hong, Zhe Chen, Chaohui Wang, Xue Mei, Danil Prokhorov, and Dacheng Tao. Multi-store tracker (muster): a cognitive psychology inspired approach to object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 749–758, 2015.
- [202] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *The Journal of Machine Learning Research*, 12:2211–2268, 2011.
- [203] Jingjing Yang, Yuanning Li, Yonghong Tian, Lingyu Duan, and Wen Gao. Group-sensitive multiple kernel learning for object categorization. In *IEEE 12th International Conference on Computer Vision, IEEE*, pages 436–443, 2009.
- [204] Huchuan Lu, Wenling Zhang, and Yen-Wei Chen. On feature combination and multiple kernel learning for object tracking. In *Computer Vision–ACCV 2010*, pages 511–522. Springer, 2011.
- [205] Fan Yang, Huchuan Lu, and Ming-Hsuan Yang. Robust visual tracking via multiple kernel boosting with affinity constraints. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(2):242–254, 2014.
- [206] David G Lowe. Object recognition from local scale-invariant features. In *IEEE 19th international conference on Computer vision, IEEE*, volume 2, pages 1150–1157, 1999.
- [207] Paul Viola, Michael J Jones, and Daniel Snow. Detecting pedestrians using patterns of motion and appearance. In *IEEE 9th International Conference on Computer Vision, IEEE*, pages 734–741, 2003.
- [208] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2037–2041, 2006.
- [209] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [210] Yong Liu and Xin Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399–1404, 1999.
- [211] Joost Van De Weijer, Cordelia Schmid, Jakob Verbeek, and Diane Larlus. Learning color names for real-world applications. *IEEE Transactions on Image Processing*, 18(7):1512–1523, 2009.
- [212] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, pages 1–42, 2014.
- [213] Ali S Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops, IEEE*, pages 512–519, 2014.
- [214] Zhi-Hua Zhou, Nitesh V Chawla, Yaochu Jin, and Greg J Williams. Big data opportunities and challenges: discussions from data analytics perspectives. *IEEE Computational Intelligence Magazine*, 9(4):62–74, 2014.

Details of the 121 UCI datasets

Table A.1 Details of the 121 UCI datasets used in this thesis.

Datasets	Number of Patterns	Number of Features	Number of Classes
abalone	4177	8	3
ac-inflam	120	6	2
acute-nephritis	120	6	2
adult	48842	14	2
annealing	798	38	6
arrhythmia	452	262	13
audiology-std	226	59	18
balance-scale	625	4	3
balloons	16	4	2
bank	45211	17	2
blood	748	4	2
breast-cancer	286	9	2
bc-wisc	699	9	2
bc-wisc-diag	569	30	2
bc-wisc-prog	198	33	2
breast-tissue	106	9	6
car	1728	6	4
ctg-10classes	2126	21	10
ctg-3classes	2126	21	3
chess-krvk	28056	6	18
chess-krvkp	3196	36	2
congress-voting	435	16	2
conn-bench-sonar	208	60	2
conn-bench-vowel	528	11	11
connect-4	67557	42	2
contrac	1473	9	3
credit-approval	690	15	2
cylinder-bands	512	35	2
dermatology	366	34	6
echocardiogram	131	10	2
ecoli	336	7	8
energy-y1	768	8	3
energy-y2	768	8	3
fertility	100	9	2
flags	194	28	8
glass	214	9	6
haberman-survival	306	3	2
hayes-roth	132	3	3
heart-cleveland	303	13	5
heart-hungarian	294	12	2
heart-switzerland	123	12	2
heart-va	200	12	5
hepatitis	155	19	2
hill-valley	606	100	2
horse-colic	300	25	2
ilpd-indian-liver	583	9	2
image-segmentation	210	19	7
ionosphere	351	33	2
iris	150	4	3
led-display	1000	7	10
lenses	24	4	3

Continued on next page

Appendix A. Details of the 121 UCI datasets

Table A.1 – continued from previous page

Datasets	Number of Patterns	Number of Features	Number of Classes
letter	20000	16	26
libras	360	90	15
low-res-spect	531	100	9
lung-cancer	32	56	3
lymphography	148	18	4
magic	19020	10	2
mammographic	961	5	2
miniboone	130064	50	2
molec-biol-promoter	106	57	2
molec-biol-splice	3190	60	3
monks-1	124	6	2
monks-2	169	6	2
monks-3	3190	6	2
mushroom	8124	21	2
musk-1	476	166	2
musk-2	6598	166	2
nursery	12960	8	5
oocMerl2F	1022	25	3
oocMerl4D	1022	41	2
oocTris2F	912	25	2
oocTris5B	912	32	3
optical	3823	62	10
ozone	2536	72	2
page-blocks	5473	10	5
parkinsons	195	22	2
pendigits	7494	16	10
pima	768	8	2
pb-MATERIAL	106	4	3
pb-REL-L	103	4	3
pb-SPAN	92	4	3
pb-T-OR-D	102	4	2
pb-TYPE	105	4	6
planning	182	12	2
plant-margin	1600	64	100
plant-shape	1600	64	100
plant-texture	1600	64	100
post-operative	90	8	3
primary-tumor	330	17	15
ringnorm	7400	20	2
seeds	210	7	3
semeion	1593	256	10
soybean	307	35	18
spambase	4601	57	2
spect	80	22	2
spectf	80	44	2
st-aus-credit	690	14	2
st-german-credit	1000	24	2
st-heart	270	13	2
st-image	2310	18	7
st-landsat	4435	36	6
st-shuttle	43500	9	7
st-vehicle	846	18	4
steel-plates	1941	27	7
synthetic-control	600	60	6
teaching	151	5	3
thyroid	3772	21	3
tic-tac-toe	958	9	2
titanic	2201	3	2
rains	10	28	2
twonorm	7400	20	2
vc-2classes	310	6	2
vc-3classes	310	6	3
wall-following	5456	24	4
waveform	5000	21	3
waveform-noise	5000	40	3
wine	179	13	3
w-qua-red	1599	11	6
w-qua-white	4898	11	7
yeast	1484	8	10
zoo	101	16	7

Some keys are: ac-inam=acute-inflammation, bc=breastcancer, congress-vot= congressional-voting, ctg=cardiotocography, conn-benchsonar/ vowel= connectionist-benchmark-sonar-mines-rocks/vowel-deterding, pb=pittsburg-bridges, st=statlog, aus=australian, vc=vertebral-column, w-qua=wine-quality.

Appendix B

Results of RVFL variants

Table B.1 Detailed results of Regularized Least Square based RVFL variants on *Sigmoid*, *Sine* and *Hardlim* Activation Function

Data	<i>Sigmoid</i>				<i>Sine</i>				<i>Hardlim</i>			
	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$
abalone	64.08	64.08	64.1	64.13	65.09	65.09	65.09	65.09	58.29	58.29	64.56	64.56
ac-inflam	100	100	100	100	100	100	100	100	100	100	100	100
acute-nephritis	100	100	100	100	100	100	100	100	93.33	93.33	100	100
adult	82.38	82.38	82.38	82.39	84.23	84.24	84.23	84.24	77.89	77.89	82.38	82.38
annealing	66	57	64	64	15	16	15	16	73	73	56	56
arrhythmia	70.13	70.13	69.47	69.47	70.8	70.8	71.68	70.35	64.82	64.82	69.91	69.69
audiology-std	64	64	64	64	64	64	64	64	56	56	64	64
balance-scale	90.38	90.38	90.38	90.38	91.83	91.83	91.83	91.83	84.46	84.46	87.34	87.34
balloons	100	75	62.5	75	62.5	56.25	62.5	62.5	93.75	93.75	62.5	62.5
bank	89	89	89	89	89.54	89.54	89.54	89.54	88.5	88.5	88.98	88.98
blood	77.14	77.14	77.14	77.14	79.41	79.28	79.55	79.41	79.95	79.95	79.28	79.28
breast-cancer	73.24	73.24	74.3	74.3	73.59	73.59	71.48	71.48	72.89	72.89	70.42	70.42
bc-wisc	97.14	97.14	97.14	97.14	97.29	97.29	97.14	97.14	97.71	97.71	97.14	97.14
bc-wisc-diag	96.48	96.48	96.48	96.48	94.48	96.48	96.48	96.48	94.37	94.37	96.13	96.13
bc-wisc-prog	76.53	76.53	81.12	81.63	79.08	78.57	79.08	78.57	78.57	78.57	81.12	81.12
breast-tissue	71.15	70.19	68.27	67.31	72.12	72.12	68.27	72.12	67.31	67.31	68.27	68.27
car	79.75	79.8	79.75	79.75	88.31	88.43	88.31	88.43	75.87	75.87	82	82
ctg-10classes	90.44	90.44	90.21	90.25	90.54	90.49	90.54	90.49	85.69	85.69	88.37	88.37
ctg-3classes	73.87	73.87	73.96	73.96	78.11	78.15	78.11	78.06	57.53	57.53	68.22	68.22
chess-krvk	26.92	26.89	26.69	26.69	32.5	32.46	32.51	32.46	19.81	19.81	26.89	26.89
chess-krvkp	95.34	95.34	95.4	95.4	95.4	95.4	95.43	95.4	73.65	73.65	94.37	94.37
congress-voting	61.24	61.24	62.16	62.16	61.7	61.24	62.16	62.16	60.09	60.09	61.47	60.09
conn-bench-sonar	75	75	75.48	75.48	77.88	78.37	77.88	78.37	80.29	80.29	76.92	76.92
conn-bench-vowel	77.92	78.14	78.14	78.14	92.21	92.21	92.21	92.21	66.23	66.23	71.65	71.65
connect-4	75.39	75.39	75.39	75.39	75.62	75.62	75.62	75.62	75.38	75.38	75.38	75.38
contrac	53.19	53.19	53.19	53.19	53.8	53.8	53.8	53.74	49.8	49.8	52.92	52.92
credit-approval	86.63	86.63	86.63	86.63	86.19	85.9	86.63	86.63	80.81	80.81	86.63	86.63
cylinder-bands	72.46	72.66	70.51	71.68	75.39	73.44	75.2	75.39	70.7	70.7	70.51	70.51
dermatology	97.25	97.25	97.25	98.08	98.63	98.08	98.63	97.53	91.48	91.48	97.8	97.8
echocardiogram	84.85	85.61	85.61	85.61	84.85	84.85	85.61	85.61	83.33	83.33	84.09	84.09
ecoli	86.9	86.9	86.9	86.9	87.5	87.5	87.5	87.5	80.06	80.06	86.31	86.31
energy-y1	87.63	87.89	87.89	87.76	88.8	88.93	88.8	88.93	84.24	84.24	84.24	84.24
energy-y2	84.64	84.64	83.46	83.46	87.24	87.24	87.24	87.24	84.51	84.51	84.51	84.51
fertility	88	88	88	88	88	88	88	88	88	88	88	88
flags	48.44	48.44	50	50	44.79	44.79	46.35	46.35	39.06	39.06	45.83	45.83
glass	65.09	65.09	65.09	65.09	66.04	66.51	63.68	64.15	67.92	67.92	65.57	65.57
haberman-survival	73.68	73.68	73.68	73.68	73.68	73.68	73.68	73.68	69.74	69.74	74.34	74.34
hayes-roth	75	75	75	75	75	75	75	75	75	75	75	75
heart-cleveland	61.51	61.18	61.18	61.51	60.53	60.2	60.2	60.86	59.21	59.21	60.2	60.2
heart-hungarian	85.27	85.27	84.93	84.93	85.27	84.93	85.27	85.27	79.79	79.79	82.53	82.53
heart-switzerland	49.19	49.19	50	50	49.19	48.39	47.58	41.94	47.58	47.58	46.77	46.77
heart-va	41.5	41	41.5	41	33.5	32.5	32.5	32.5	34.5	34.5	34.5	34.5
hepatitis	78.85	78.85	84.62	84.62	81.41	78.21	79.49	83.97	83.33	83.33	81.41	81.41
hill-valley	49.34	49.34	49.34	49.34	49.34	49.34	49.5	49.34	49.34	49.34	49.34	49.34
horse-colic	63.24	63.24	77.94	77.94	60.29	90.29	60.29	60.29	63.24	63.24	73.53	73.53
ilpd-indian-liver	72.95	72.95	72.95	72.77	73.12	73.29	73.12	73.29	72.43	72.43	72.26	72.26

Continued on next page

Appendix B. Results of RVFL variants

Table B.1 – continued from previous page

Data	Sigmoid				Sine				Hardlim			
	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$
image-segmentation	28	28	27.86	27.86	25.62	25.38	25.29	25.29	14.67	14.67	18.9	18.9
ionosphere	87.22	89.49	90.06	90.06	87.5	87.22	87.78	87.5	86.65	86.65	88.92	88.92
iris	97.3	97.3	97.3	97.3	97.97	97.97	97.97	97.97	93.24	93.24	97.3	97.3
led-display	75.7	75.7	75.7	75.1	73.9	73.9	73	73	69	69	72	72
lenses	83.33	83.33	70.83	70.83	79.17	79.17	83.33	83.33	87.5	87.5	87.5	87.5
letter	58.77	58.82	58.87	58.9	73.49	73.59	73.5	73.61	21.86	21.86	58.07	58.07
libras	76.67	77.22	76.94	76.94	79.44	79.44	79.44	79.44	71.94	73.61	75.28	75.28
low-res-spect	89.29	89.29	89.66	89.66	89.85	90.04	90.04	89.85	87.59	87.59	88.16	88.16
lung-cancer	53.13	50	53.13	53.13	46.88	46.88	50	50	43.75	43.75	62.5	40.23
lymphography	87.84	87.84	89.19	89.19	83.11	83.11	82.43	82.43	77.7	77.7	88.51	88.51
magic	78.62	78.62	78.62	78.63	83.84	83.89	83.84	83.89	73.28	73.28	79.46	79.46
mammographic	82.19	82.19	82.29	82.29	83.02	82.71	83.02	83.02	75.04	75.04	82.08	82.08
miniboone	88.41	88.41	88.72	88.72	89.86	89.86	89.87	89.87	81.24	81.24	87.45	87.45
molec-biol-promoter	78.85	78.85	81.73	81.73	80.77	80.77	92.31	78.85	75	75	87.5	87.5
molec-biol-splice	81.02	81.05	80.55	80.58	81.27	81.24	80.68	80.65	59.5	59.5	80.11	80.11
monks-1	62.96	62.73	62.04	62.96	56.71	56.48	60.88	59.72	55.09	55.09	55.32	55.32
monks-2	67.13	67.13	67.13	67.13	66.9	66.9	66.9	66.9	66.2	66.2	66.44	66.44
monks-3	53.47	53.47	57.18	57.18	63.19	63.19	63.19	62.96	53.01	53.01	55.56	55.56
mushroom	95.52	95.52	95.52	95.53	99.66	99.66	99.66	99.67	84.33	84.33	95.05	95.05
musk-1	80.67	80.67	79.83	79.83	80.67	80.46	81.06	81.09	76.89	76.89	80.04	80.04
musk-2	94.44	94.51	94.78	94.77	94.74	94.69	95.69	95.95	87.17	87.17	94.44	94.44
nursery	89.74	89.74	89.74	89.75	91.61	91.62	91.61	91.62	57.21	57.21	89.17	89.17
oocMer12F	82.35	82.35	83.33	83.33	85.88	85.88	85.88	85.88	71.47	71.47	80.59	80.59
oocMer14D	91.86	91.86	91.96	91.96	93.04	93.43	93.04	93.04	89.61	89.61	91.27	91.27
oocTris2F	80.7	80.7	80.37	80.37	85.53	85.42	85.42	85.42	68.31	68.31	79.28	79.28
oocTris5B	92.87	92.76	93.09	93.09	93.75	93.53	93.75	93.86	84.98	84.98	92.21	92.21
optical	94.32	94.32	94.55	94.55	95.27	95.33	95.27	95.16	63.49	63.49	92.32	92.32
ozone	97.16	97.16	97.16	97.16	97.16	97.16	97.16	97.16	97.16	97.16	97.16	97.16
page-blocks	93.38	93.38	93.38	93.38	95.01	95.01	95.01	95.01	92.74	92.74	94.66	94.66
parkinsons	88.78	89.29	89.29	89.29	87.76	88.27	87.76	87.24	86.73	86.73	86.22	86.22
pendigits	84.16	84.16	84.19	84.19	92.11	92.11	92.11	92.11	62.61	62.61	83.65	83.65
pima	76.3	76.3	76.3	76.3	76.56	76.56	76.56	76.56	73.7	73.7	77.08	77.08
pb-MATERIAL	92.31	92.31	92.31	92.31	92.31	92.31	92.31	92.31	87.5	87.5	92.31	92.31
pb-REL-L	69.23	69.23	69.23	69.23	70.19	70.19	70.19	70.19	73.08	73.08	73.08	73.08
pb-SPAN	69.57	69.57	71.74	71.74	69.57	69.57	73.91	73.91	72.83	72.83	72.83	72.83
pb-T-OR-D	89	89	89	89	89	89	89	89	87	87	86	88
pb-TYPE	67.31	67.31	67.31	67.31	67.31	67.31	66.35	67.31	62.5	62.5	61.4	61.4
planning	71.11	71.11	71.11	71.11	71.11	71.11	71.11	71.11	70	70	67.22	67.22
plant-margin	75.06	75.19	75.63	75.69	76.12	76.19	75.56	75.62	41	41	61.69	61.69
plant-shape	40.94	40.94	41.69	41.75	48.62	48.75	48.8	49	35.75	35.75	46.25	46.25
plant-texture	77.25	77.13	77.63	77.69	78.75	78.94	78.94	79.37	50.44	50.44	67.06	67.06
post-operative	72.73	72.73	72.73	72.73	72.73	72.73	72.73	72.73	72.73	72.73	72.73	72.73
primary-tumor	53.66	53.66	54.57	54.57	53.35	53.05	53.35	53.35	49.39	49.39	52.74	52.74
ringnorm	89.5	89.92	89.64	90.08	92.45	94.84	92.45	94.84	79.07	79.07	87.12	87.12
seeds	95.67	95.67	95.67	95.67	95.19	95.19	95.67	95.67	90.87	90.87	94.71	94.71
semeion	86.37	86.37	88.19	87.12	86.68	86.75	87.75	88.06	72.49	72.49	87.44	87.44
soybean	47.34	47.87	48.94	48.94	45.74	45.21	48.67	48.67	41.76	41.76	49.2	49.2
spambase	90.57	90.54	90.57	90.54	90.52	90.72	90.59	90.61	75.07	75.07	90.41	90.41
spect	58.06	57.53	54.84	54.84	67.2	54.84	69.35	54.84	60.22	60.22	63.44	63.44
spectf	91.98	91.98	91.98	91.98	91.44	66.84	91.98	91.44	91.44	91.44	91.98	91.98
st-aus-credit	67.44	67.44	67.3	67.44	66.42	64.97	66.28	65.12	67.3	67.3	67.3	67.3
st-german-credit	78.2	78.1	77.7	77.7	77.5	77.4	77.5	77.7	73	73	78	78
st-heart	88.06	88.06	88.43	88.43	87.69	88.06	88.43	88.43	82.46	82.09	86.57	86.57
st-image	88.91	88.99	88.95	88.95	92.81	92.29	92.81	92.29	76.21	76.21	89.34	89.34
st-landsat	82.35	82.4	82.5	82.5	84.5	84.55	84.5	84.6	73.5	73.5	82.05	82.05
st-shuttle	89.1	89.1	89.1	89.1	92.26	92.37	92.28	92.34	88.06	88.06	96.87	96.87
st-vehicle	77.84	77.84	78.08	78.91	83.41	80.45	83.41	83.41	60.55	60.55	76.54	76.54
steel-plates	70.46	70.46	70.67	70.67	73.76	73.71	73.81	73.76	57.94	57.94	66.24	66.24
synthetic-control	95.33	95.33	95.5	95.5	95.17	95.17	95.17	95.17	91	91	91	91
teaching	51.97	53.95	51.97	51.97	56.58	55.26	49.34	51.97	51.32	51.32	51.97	51.97
thyroid	93.79	93.79	93.79	93.79	93.82	93.82	93.82	93.82	93.38	93.38	93.76	93.76
tic-tac-toe	97.59	97.59	97.59	97.59	97.59	97.59	97.59	97.59	73.95	73.95	97.59	97.59
titanic	78.18	78.14	78.18	78.18	78.41	78.41	78.59	78.59	78.95	78.95	78.95	78.95
rains	87.5	87.5	87.5	87.5	100	100	87.5	87.5	87.5	87.5	87.5	87.5
twonorm	97.88	97.88	97.92	97.92	97.96	97.96	97.92	97.92	81.64	81.64	97.91	97.91
vc-2classes	87.01	87.01	87.01	87.01	86.69	86.69	86.69	86.69	84.09	84.09	83.12	83.12
vc-3classes	81.17	81.17	81.49	81.49	81.49	81.49	81.82	81.82	82.14	82.14	81.17	81.17
wall-following	70.44	70.51	70.58	70.61	80.68	80.85	80.74	80.92	51.36	51.36	65.8	65.8
waveform	86.74	86.76	86.58	86.58	86.56	86.74	86.54	86.58	74.64	74.64	85.9	85.9
waveform-noise	85.68	85.66	86.02	86.02	85.88	85.88	85.88	85.86	60.3	60.3	86	86
wine	98.86	99.43	99.43	98.86	98.86	99.43	100	98.3	94.89	94.89	99.43	99.43
w-qua-red	59.81	59.81	59.69	59.75	59.06	59.06	59.06	59.06	55.5	55.5	58.94	58.94
w-qua-white	70.44	70.51	70.58	70.64	54.51	54.53	54.51	54.53	49.8	49.8	54.55	54.55
yeast	54	54.02	54.04	54.06	60.71	60.71	60.78	60.71	52.22	52.22	57.41	57.41
zoo	97.88	97.88	97.92	97.92	97.96	97.88	97.92	97.92	81.64	81.64	97.91	97.91

Table B.2 Detailed results of Regularized Least Square based RVFL variants on *Tribas*, *Radbas* and *Sign* Activation Function

Data	<i>Tribas</i>				<i>Radbas</i>				<i>Sign</i>			
	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$
abalone	65.47	65.47	65.47	65.47	65.68	65.68	65.71	65.71	58.36	58.36	64.56	64.56
ac-inflam	100	100	100	100	100	100	100	100	100	100	100	100
acute-nephritis	100	100	100	100	100	100	100	100	93.33	93.33	100	100
adult	83.1	83.1	83.1	83.1	84.61	84.61	84.61	84.61	77.89	77.89	82.38	82.38
annealing	24	24	24	24	18	18	20	17	73	73	56	56
arrhythmia	66.81	66.81	71.46	71.46	67.04	67.26	72.12	72.12	65.49	65.93	71.24	69.25
audiology-std	60	60	60	60	64	64	64	64	56	56	64	64
balance-scale	89.9	89.9	89.9	89.9	91.35	91.35	91.35	91.35	84.29	84.29	87.34	87.34
balloons	81.25	81.25	81.25	81.25	87.5	87.5	87.5	87.5	93.75	93.75	62.5	62.5
bank	89.23	89.23	89.23	89.23	89.85	89.85	89.8	89.8	88.47	88.47	88.98	88.98
blood	79.68	79.68	79.68	79.68	79.81	79.81	79.41	79.41	79.95	79.95	79.28	79.28
breast-cancer	71.83	71.83	71.83	71.83	73.94	73.94	73.59	73.59	72.89	72.89	70.07	70.07
bc-wisc	97.29	97.29	97.29	97.29	97.14	97.14	97.14	97.14	97.29	97.29	97.14	97.14
bc-wisc-diag	96.65	96.65	96.65	96.65	96.13	96.13	96.13	95.95	94.37	94.37	95.95	95.95
bc-wisc-prog	80.61	80.61	80.61	79.59	75	75	75	80.61	76.53	76.53	81.12	81.12
breast-tissue	69.23	69.23	69.23	69.23	61.54	61.54	61.54	61.54	66.35	66.35	68.27	68.27
car	87.79	87.79	87.79	87.79	89.76	89.76	89.81	89.81	75.87	75.87	82	82
ctg-10classes	88.98	88.98	88.98	88.98	90.63	90.63	90.77	90.77	85.97	85.97	88.37	88.37
ctg-3classes	72.41	72.41	72.41	72.41	79.1	79.1	79.28	79.28	57.16	57.16	68.22	68.22
chess-krvk	27.74	27.74	27.74	27.74	34.11	34.1	34.11	33.97	19.81	19.81	26.89	26.89
chess-krvkv	94.96	94.96	94.96	94.96	95.4	95.4	95.49	95.49	73.65	73.64	94.37	94.37
congress-voting	59.86	61.93	62.39	62.16	63.07	63.07	62.61	62.61	61.24	61.24	61.47	61.47
conn-bench-sonar	81.73	81.73	82.69	82.69	81.25	81.73	81.25	81.25	78.85	78.85	78.85	78.85
conn-bench-vowel	89.61	89.61	89.61	89.61	95.02	95.02	95.24	95.24	66.23	66.23	71.65	71.65
connect-4	75.45	75.45	75.45	75.45	75.56	75.56	75.57	75.58	75.38	75.38	75.38	75.38
contrac	55.5	55.5	55.5	55.5	52.65	52.38	52.58	52.65	49.8	49.8	52.92	52.92
credit-approval	86.77	86.92	87.06	87.06	86.48	87.79	86.63	86.63	80.81	80.81	86.63	86.63
cylinder-bands	75.78	75.78	75.59	75.59	74.8	74.8	74.8	74.8	70.7	70.7	74.61	74.61
dermatology	97.25	97.25	97.25	97.25	97.53	97.53	97.53	97.53	91.48	91.48	97.8	97.8
echocardiogram	84.09	84.09	84.09	84.09	86.36	86.36	87.12	87.12	84.09	84.09	83.33	83.33
ecoli	86.01	86.01	86.31	86.31	88.39	87.5	87.8	87.8	80.06	80.06	86.31	86.31
energy-y1	89.45	89.45	89.45	89.45	88.8	88.8	88.8	88.8	84.24	84.24	84.24	84.24
energy-y2	88.8	88.8	88.8	88.8	89.06	89.06	89.58	89.58	84.64	84.64	84.51	84.51
fertility	89	89	89	89	87	87	87	87	89	89	88	88
flags	89	89	89	89	48.44	48.44	51.04	52.08	21	38.54	45.83	45.83
glass	64.62	64.62	68.4	68.4	66.51	66.51	66.51	66.51	68.4	68.4	61.79	61.79
haberman-survival	75	75	75	75	73.68	73.68	73.68	73.68	69.74	69.74	74.34	74.34
hayes-roth	78.57	78.57	78.57	78.57	75	75	75	75	71.43	71.43	71.43	71.43
heart-cleveland	61.18	60.86	61.18	60.86	60.86	60.86	60.86	60.86	59.21	59.21	60.2	60.2
heart-hungarian	87.33	87.33	87.33	87.33	83.22	83.56	83.56	83.22	80.14	80.14	81.85	81.85
heart-switzerland	41.94	41.94	41.94	41.94	37.1	37.1	37.1	37.1	47.58	47.58	47.58	47.58
heart-va	32	32	32.5	32.5	35	35	35	35	34.5	34.5	34.5	34.5
hepatitis	83.97	86.54	83.33	83.33	83.33	83.97	83.33	83.33	82.69	82.69	78.85	79.49
hill-valley	49.34	49.34	49.5	49.34	49.34	49.37	49.34	49.34	49.34	49.34	49.34	49.34
horse-colic	67.65	67.65	69.12	69.12	64.71	69.12	69.12	69.12	60.29	60.29	73.53	73.53
ilpd-indian-liver	73.46	73.46	73.46	73.46	72.95	72.95	72.95	72.95	72.43	72.43	72.43	72.43
image-segmentation	28.57	28.57	28.57	28.57	26.86	26.86	27.43	27.43	14.67	14.67	18.9	18.9
ionosphere	89.2	85.23	89.2	89.2	89.49	89.49	89.77	89.77	87.78	87.78	87.78	87.5
iris	97.97	97.97	97.97	97.97	98.65	98.65	98.65	98.65	93.24	93.24	97.3	97.3
led-display	72.6	72.6	72.4	72.3	74.1	73.9	73.8	73.8	69.3	69.3	72	72
lenses	83.3	83.3	83.3	83.3	75	75	75	75	87.5	87.5	87.5	87.5
letter	63.06	63.06	63.06	63.06	76.77	76.77	76.78	76.78	21.83	21.83	58.07	58.07
libras	82.2	82.2	82.2	82.2	80.28	80.28	80.28	80.28	73.61	73.61	73.33	73.33
low-res-spect	90.04	90.04	90.04	90.04	90.41	90.41	90.79	90.79	87.59	87.59	87.97	87.97
lung-cancer	37.5	37.5	46.83	40.63	43.75	43.75	43.75	46.88	40.63	40.63	37.5	40.63
lymphography	87.84	87.84	88.51	88.51	87.84	87.84	86.49	86.49	77.7	77.7	88.51	88.51
magic	80.93	80.93	80.93	80.93	84.75	84.75	84.75	84.75	73.28	73.28	79.46	79.46
mammographic	81.25	81.25	81.25	81.25	82.92	82.92	83.13	83.13	75.94	75.94	82.08	82.08
miniboone	89.86	89.86	89.88	89.88	89.93	89.93	89.94	89.94	81.24	81.24	87.45	87.45
molec-biol-promoter	78.85	78.85	79.81	83.65	81.73	81.73	80.77	70.88	74.04	74.04	82.69	65.38
molec-biol-splice	81.87	81.87	81.84	81.84	81.37	81.37	81.37	81.37	59.35	59.35	80.11	80.11
monks-1	50.69	50.69	50.69	50.69	50.46	50.46	50.46	50	55.09	55.09	55.09	55.09
monks-2	66.67	66.67	66.67	66.67	66.67	66.67	66.67	66.67	66.2	66.2	66.2	66.2
monks-3	64.35	64.12	64.12	64.12	48.38	48.38	48.38	48.38	53.01	53.01	55.32	55.32
mushroom	96.44	96.44	96.44	96.44	99.94	99.94	99.94	99.94	84.33	84.33	95.05	95.05
musk-1	85.71	85.71	82.35	82.35	83.82	83.82	84.02	84.02	73.32	73.32	80.88	80.88
musk-2	94.5	94.56	94.94	94.98	94.56	94.6	95.62	95.85	87.17	87.17	94.44	94.44
nursery	89.42	89.42	89.42	89.42	92.15	92.15	92.15	92.15	57.44	57.44	89.17	89.17

Continued on next page

Appendix B. Results of RVFL variants

Table B.2 – continued from previous page

Data	Tribas				Radbas				Sign			
	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$
oocMerl2F	84.31	84.31	84.41	84.41	85.59	85.49	85.69	85.69	71.47	71.47	80.59	80.59
oocMerl4D	91.47	91.47	91.47	91.47	93.14	93.14	93.14	93.14	89.71	89.71	91.27	91.27
oocTris2F	83.88	84.21	83.77	83.88	84.32	84.32	84.87	84.87	68.86	68.86	79.28	79.28
oocTris5B	94.19	94.19	94.08	94.08	94.74	94.74	93.97	93.97	84.98	84.98	92.21	92.21
optical	93.27	93.27	93.38	93.38	94.94	94.88	95.16	95.16	63.49	63.49	92.43	92.43
ozone	97.16	97.16	97.16	97.08	97.16	97.16	97.16	97.16	97.16	97.16	97.16	97.16
page-blocks	95.6	95.6	95.6	95.6	95.12	95.12	95.12	95.12	92.74	92.74	94.66	94.66
parkinsons	87.24	87.24	87.76	87.76	87.24	87.24	87.24	87.24	86.73	86.73	86.73	86.73
pendigits	88.05	88.05	88.05	88.05	93.94	93.94	93.94	93.94	62.61	62.61	83.65	83.65
pima	76.95	76.95	76.95	76.95	76.04	76.04	76.43	76.17	74.09	74.09	77.08	77.08
pb-MATERIAL	91.35	91.35	91.35	91.35	92.31	92.31	92.31	92.31	89.42	89.42	92.31	92.31
pb-REL-L	67.31	67.31	66.35	66.35	70.19	70.19	70.19	70.19	73.08	73.08	73.08	73.08
pb-SPAN	70.65	70.65	77.17	70.65	69.57	69.57	71.74	69.57	69.57	69.57	69.57	69.57
pb-T-OR-D	89	89	89	89	89	89	89	88	86	86	86	88
pb-TYPE	70.19	70.19	70.19	70.19	67.31	67.31	67.31	67.31	58.65	58.65	61.54	61.54
planning	70	70	67.22	67.22	70	70	67.22	67.22	70	70	70	70
plant-margin	72.62	72.62	72.69	72.69	76.12	76.62	77	77	39.5	39.5	61.56	61.56
plant-shape	55.13	55.13	55.44	55.44	52.19	52	52.87	52.94	35	35	46.19	46.19
plant-texture	74.44	74.5	74.56	74.56	78.19	78.13	78.5	78.31	50	50	67.13	67.13
post-operative	72.73	72.73	72.73	72.73	72.73	72.73	72.73	72.73	72.73	72.73	72.73	72.73
primary-tumor	54.88	54.88	54.88	54.88	56.71	56.71	53.35	53.96	49.09	49.09	52.74	52.74
ringnorm	92.49	92.49	92.49	92.49	94.91	94.91	94.92	94.91	79.07	79.07	87.12	87.12
seeds	94.71	94.71	95.67	95.67	95.19	95.19	95.67	95.67	88.46	88.46	94.71	94.71
semeion	87.06	87.06	87.69	88.76	86.68	86.68	88.13	88.25	70.73	70.73	87.44	87.44
soybean	55.85	55.32	53.99	53.99	55.32	55.05	52.66	52.93	41.76	41.76	49.2	49.2
spambase	91.39	91.39	91.39	91.39	90.78	90.76	91.07	91.07	75.09	75.09	90.33	90.33
spect	67.28	54.84	69.35	54.84	67.2	54.84	69.35	54.84	55.38	55.38	54.84	54.84
spectf	83.96	43.32	83.96	83.96	90.91	91.44	90.91	91.98	91.44	91.44	91.98	91.98
st-aus-credit	66.72	66.72	66.72	66.72	67.73	67.73	67.44	67.44	67.3	67.3	67.3	67.3
st-german-credit	77.6	77.6	77.6	77.6	76.1	76	77.3	77.1	73	73	73	73
st-heart	86.94	86.94	86.57	86.57	87.31	87.31	86.94	86.94	82.46	82.46	86.57	86.57
st-image	89.99	89.99	90.03	90.03	93.15	93.2	93.15	93.15	76.21	76.21	89.34	89.34
st-landsat	83.1	83.1	83.1	83.1	84.8	84.8	84.85	84.85	73.5	73.5	82.05	82.05
st-shuttle	94.43	96.43	96.43	96.43	94.23	94.23	94.23	94.23	88.06	88.06	96.87	96.87
st-vehicle	77.37	77.37	77.49	77.49	82.7	82.58	82.58	82.58	60.55	60.55	76.54	76.54
steel-plates	70.72	70.72	70.72	70.72	73.87	73.92	74.07	74.12	57.94	57.94	66.24	66.24
synthetic-control	95.5	96.83	96.83	96.83	96.33	96.33	96.33	96.33	90.5	90.5	91	91
teaching	40.79	57.24	55.26	54.61	53.29	53.95	51.32	53.95	51.32	51.32	51.32	51.32
thyroid	93.76	93.76	93.76	93.76	93.79	93.79	93.79	93.79	93.38	93.38	93.76	93.76
tic-tac-toe	97.38	97.49	97.59	97.59	97.59	97.59	97.59	97.59	73.95	73.95	97.59	97.59
titanic	78.68	78.68	78.68	78.68	78.59	78.59	78.59	78.59	78.95	78.95	78.95	78.95
rains	87.5	62.5	87.5	87.5	37.5	37.5	87.5	87.5	87.5	87.5	87.5	87.5
twonorm	97.85	97.85	97.86	97.86	97.91	97.91	97.81	97.81	81.64	81.64	97.91	97.91
vc-2classes	85.06	85.06	85.06	85.06	87.01	87.01	87.01	87.01	84.09	84.09	83.12	83.12
vc-3classes	82.47	82.47	82.47	82.47	82.14	82.14	85.06	82.79	82.14	82.14	81.17	81.17
wall-following	70.88	70.88	70.88	70.89	82.33	82.33	82.5	82.5	51.36	51.36	65.8	65.8
waveform	85.56	85.56	85.56	85.56	86.22	86.28	86.24	86.24	74.74	74.74	85.9	85.9
waveform-noise	86.38	86.38	86.54	86.48	86.36	86.36	86.36	86.36	60.3	60.3	85.98	85.98
wine	98.3	98.3	98.43	98.86	99.43	99.43	99.43	99.43	94.89	94.89	99.43	99.43
w-qu-a-red	59.25	59.25	59.31	59.32	59.69	59.69	59.62	59.62	55.37	55.37	58.94	58.94
w-qu-a-white	53.8	53.8	53.82	53.82	54.04	54.06	54	53.98	49.8	49.8	54.45	54.45
yeast	60.71	60.71	60.71	60.71	60.65	60.65	60.65	60.65	52.29	52.29	57.41	57.41
zoo	97.85	97.85	97.86	97.86	97	97	97	97	81.64	81.64	97.97	97.97

Table B.3 Detailed results of Moore-Penrose pseudoinverse based RVFL variants on *Sigmoid*, *Sine* and *Hardlim* Activation Function

Data	Sigmoid				Sine				Hardlim			
	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$
abalone	65.88	65.78	65.85	66.04	66.02	66.5	66.26	66.04	57.47	57.47	64.68	64.68
ac-inflam	100	100	100	100	100	100	100	100	100	100	100	100
acute-nephritis	100	100	100	100	100	100	100	100	91.67	91.67	100	100
adult	84.96	84.9	84.91	84.88	84.83	84.9	84.88	84.88	77.87	77.87	82.38	82.38
annealing	14	17	14	14	16	15	15	15	65	65	16	16
arrhythmia	66.81	68.36	38.72	38.72	65.71	65.04	32.74	32.52	59.29	0	37.17	37.61
audiology-std	60	60	64	36	64	56	64	36	12	12	64	64
balance-scale	91.03	91.03	91.03	91.03	90.38	90.22	90.22	90.22	84.13	84.13	88.46	88.46
balloons	68.75	75	56.25	56.25	75	75	62.5	56.25	81.25	81.25	93.75	93.75
bank	89.78	89.76	89.82	89.89	89.78	89.82	89.82	89.82	88.61	88.61	89	89
blood	77.01	77.27	77.54	78.88	78.21	77.27	77.81	78.21	79.81	79.81	79.28	79.28

Continued on next page

Table B.3 – continued from previous page

Data	Sigmoid				Sine				Hardlim			
	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$
breast-cancer	70.07	72.89	69.37	72.54	69.37	69.37	75	72.54	71.13	71.13	71.13	71.13
bc-wisc	96.29	96.71	96.43	96.43	96.57	96.71	96.57	96.43	96.71	96.71	97.14	97.14
bc-wisc-diag	95.95	95.95	95.77	96.13	96.3	96.13	95.42	95.95	94.72	94.72	95.42	95.42
bc-wisc-prog	77.55	79.59	79.59	81.12	77.55	77.55	79.08	81.12	77.55	77.55	79.59	79.59
breast-tissue	62.5	65.38	66.35	64.42	62.5	64.42	68.27	64.42	56.73	56.73	68.27	68.27
car	95.6	95.72	96.12	96.06	96.35	96.18	96.24	95.72	75.93	75.93	82.18	82.18
ctg-10classes	79	79.1	77.97	79	79.05	78.77	78.95	78.86	57.02	57.02	68.83	68.83
ctg-3classes	91.01	91.2	91.1	90.91	90.54	90.96	90.96	90.16	85.45	85.45	88.56	88.56
chess-krvk	41.96	41.97	42.09	42.06	45.13	45.25	45.27	45.42	19.81	19.81	26.74	26.74
chess-krvkv	94.96	94.99	95.4	95.34	94.87	94.96	95.53	95.4	73.56	73.56	94.46	94.46
congress-voting	61.7	61.01	58.49	58.72	62.16	61.01	58.94	59.63	61.47	61.47	59.17	59.17
conn-bench-sonar	69.71	68.27	70.19	70.67	72.12	67.31	70.19	70.67	73.56	73.56	49.04	70.19
conn-bench-vowel	93.94	94.59	93.29	93.07	93.07	95.24	97.62	93.94	66.67	66.67	71.65	71.65
connect-4	75.85	75.82	76.4	76.42	75.82	75.8	76.36	76.42	75.38	75.38	75.44	75.44
contrac	51.9	51.7	53.53	53.74	52.11	51.49	53.8	53.67	49.93	49.93	52.92	52.92
credit-approval	86.19	86.34	86.92	87.06	86.19	86.19	88.95	87.06	80.09	80.09	86.92	86.92
cylinder-bands	73.63	73.63	73.63	77.73	74.22	72.66	78.32	74.02	69.53	69.53	70.12	70.12
dermatology	97.53	97.25	96.43	96.7	97.53	97.53	96.7	96.7	89.29	89.29	97.8	97.8
echocardiogram	79.55	78.79	85.61	87.12	80.3	78.03	85.61	87.12	77.27	77.27	84.09	84.09
ecoli	86.61	86.9	85.71	85.71	88.39	86.9	85.71	85.71	78.87	78.87	84.82	84.82
energy-y1	95.05	95.05	94.92	94.79	95.05	95.05	94.79	94.4	83.85	83.85	86.59	86.59
energy-y2	90.36	90.36	90.23	91.8	91.02	90.1	90.23	91.28	84.77	84.77	83.2	83.2
fertility	88	88	88	88	88	88	88	88	88	88	88	88
flags	46.35	47.4	45.83	46.35	46.35	47.4	44.79	45.83	35.42	35.42	44.79	44.79
glass	67.92	67.45	64.62	58.96	68.4	68.4	66.98	66.04	60.38	60.38	64.15	64.15
haberman-survival	73.68	72.7	74.01	73.36	74.67	73.03	73.68	73.36	69.74	69.74	74.34	74.34
hayes-roth	85.71	85.71	78.57	82.14	75	85.71	78.57	78.57	71.43	71.43	67.86	71.43
heart-cleveland	58.88	57.57	61.51	61.18	61.18	56.58	59.21	61.18	60.2	60.2	59.87	59.87
heart-hungarian	83.22	81.85	84.59	83.9	83.22	81.16	85.62	83.9	77.4	77.4	84.59	84.59
heart-switzerland	46.77	42.74	41.13	45.16	45.97	37.9	45.16	44.35	43.55	43.55	25.81	25.81
heart-vv	33	32.5	36	35.5	34	33	33	36	29.5	29.5	31.5	31.5
hepatitis	76.92	76.92	69.23	68.59	78.21	78.21	78.85	73.72	79.49	79.49	76.92	76.92
hill-valley	78.38	78.55	56.6	50.5	47.69	47.69	47.19	47.85	49.34	49.34	72.77	72.77
horse-colic	44.12	44.12	66.18	66.18	39.71	42.65	33.82	66.18	54.41	54.41	66.18	66.18
ilpd-indian-liver	71.92	72.09	72.43	73.8	71.92	71.92	72.77	73.8	72.09	72.09	72.26	72.26
image-segmentation	25.33	26.62	23.43	0.48	24.29	24.71	25.19	0.57	21.14	21.14	19.43	19.43
ionosphere	87.78	89.2	86.93	87.78	89.2	90.34	89.2	88.07	86.36	86.36	88.64	88.64
iris	98.65	98.65	97.97	93.24	98.65	98.65	98.65	97.97	94.59	94.59	97.3	97.3
led-display	73.9	73.4	73.9	70.4	74.2	73.7	74.3	70.3	69.3	69.3	72.1	72.1
lenses	41.67	41.67	50	50	45.83	45.83	58.33	58.33	87.5	87.5	87.5	87.5
letter	82.59	82.67	83.16	83.25	82.54	82.64	83.2	83.16	21.84	21.84	58.07	58.07
libras	75	72.78	47.78	47.78	76.11	72.78	60.28	52.22	61.11	61.11	56.67	56.67
low-res-spect	86.65	86.28	86.84	86.47	85.34	87.03	86.28	86.47	80.83	80.83	83.08	83.08
lung-cancer	53.13	53.13	46.88	46.88	46.88	46.88	46.88	46.88	59.38	59.38	59.38	59.38
lymphography	84.46	70.27	81.76	82.43	83.78	69.59	84.46	83.11	72.97	72.97	87.84	87.84
magic	85.69	85.76	85.81	85.78	85.76	85.76	85.72	85.74	73.28	73.28	79.46	79.46
mammographic	81.98	81.46	83.02	79.69	81.88	81.67	81.46	81.46	76.15	76.15	82.19	82.19
miniboone	91.41	91.41	91.7	91.7	91.38	91.39	91.67	91.7	81.24	81.24	90.13	90.13
molec-biol-promoter	65.38	76.92	72.12	71.15	67.31	67.31	74.04	66.35	71.15	71.15	72.12	61.54
molec-biol-splice	80.52	80.87	81.56	80.93	80.3	80.65	81.24	80.93	58.91	58.91	80.33	80.33
monks-1	52.08	50	57.18	51.39	50.46	50.23	52.08	51.39	55.09	55.09	61.81	61.81
monks-2	58.1	67.13	66.67	66.67	56.71	67.13	66.67	66.67	67.36	67.36	66.44	66.44
monks-3	48.84	51.39	37.27	37.73	50.69	49.31	37.5	36.81	53.01	53.01	55.32	55.32
mushroom	100	99.91	100	100	99.93	99.93	100	100	84.28	84.28	95.06	95.06
musk-1	76.26	81.09	77.1	68.49	78.99	79.83	78.15	78.15	73.32	73.32	75.42	75
musk-2	94.1	94.82	96.33	96.39	94.75	94.21	95.94	96.38	87.17	87.17	94.42	94.42
nursery	95.76	95.72	95.93	95.88	95.83	95.83	95.86	95.86	57.48	57.48	89.17	89.17
oocMerl2F	84.71	84.31	83.14	80.2	84.22	83.82	82.94	83.24	70.88	70.88	80.78	80.78
oocMerl4D	92.25	91.96	91.57	91.76	93.24	91.96	91.76	91.76	89.41	89.41	91.08	91.08
oocIris2F	83.99	84.43	84.65	84.87	84.1	83.99	84.54	84.54	69.08	69.08	78.73	78.73
oocIris5B	93.64	93.86	94.52	92.54	93.53	93.31	94.19	92.54	83.99	83.99	91.89	91.89
optical	95.33	95.38	95.77	95.94	95.21	95.27	95.77	95.88	63.49	63.49	92.49	92.49
ozone	97.16	97.16	97.08	97.08	97.16	97.16	97.08	97.08	97.16	97.16	97.08	97.08
page-blocks	95.85	95.8	95.76	95.58	95.92	95.89	95.72	95.83	92.73	92.73	94.63	94.63
parkinsons	87.76	89.29	87.24	86.22	86.22	89.29	85.71	85.71	80.1	80.1	87.24	87.24
pendigits	95.74	96.54	96.88	96.86	96.6	96.57	96.86	96.86	62.52	62.52	83.56	83.56
pima	75.78	75.26	77.08	76.56	76.56	75.65	76.82	76.56	74.48	74.48	76.69	76.69
pb-MATERIAL	91.35	87.5	91.35	91.35	90.38	87.5	89.42	91.35	68.27	68.27	92.31	92.31
pb-REL-L	61.54	64.42	66.35	70.19	63.46	62.5	71.15	70.19	71.15	71.15	71.15	71.15
pb-SPAN	61.96	63.04	69.57	67.39	54.35	56.52	68.48	67.39	55.43	55.43	68.48	68.48
pb-T-OR-D	88	89	88	85	88	88	86	84	88	88	87	87
pb-TYPE	62.5	51.92	61.54	62.5	59.62	58.65	63.46	62.5	53.85	53.85	66.35	66.35
planning	71.11	71.11	65.56	66.11	71.11	71.11	66.67	66.11	71.11	71.11	68.33	68.33
plant-margin	76.69	76.5	74.75	75.44	76.19	76.75	77	75.5	38.06	38.06	61.69	61.69
plant-shape	50.81	51	53.13	53.25	50.06	50.38	52.88	53.88	34.56	34.56	45.31	45.31
plant-texture	78.69	78.69	79.88	80.25	78.69	78.81	80	79.94	48.94	48.94	67.25	67.25

Continued on next page

Appendix B. Results of RVFL variants

Table B.3 – continued from previous page

Data	Sigmoid				Sine				Hardlim			
	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$
post-operative	72.73	72.73	71.59	71.59	68.18	72.73	71.59	71.59	70.45	72.73	70.45	70.45
primary-tumor	56.71	56.71	50.61	50.3	56.4	56.1	50.61	50.61	42.07	42.07	51.52	51.52
ringnorm	95.03	94.84	95.03	95.22	93.82	94.84	94.43	95.26	79.07	79.07	87.11	87.11
	95.19	94.71	95.67	95.19	95.19	95.19	96.15	95.67	87.98	87.98	95.67	95.67
semeion	84.92	84.86	85.18	85.18	84.99	85.24	85.24	85.11	69.1	69.1	82.35	82.35
soybean	47.34	47.34	47.87	46.81	46.01	46.81	47.07	46.54	32.98	32.98	48.94	48.94
spambase	90.72	90.65	90.74	90.7	90.26	90.63	90.74	90.76	75.24	75.24	90.04	90.04
spect	63.44	63.98	62.37	62.37	40.32	44.62	62.37	61.83	58.6	58.6	58.06	58.06
spectf	63.1	66.84	54.55	54.55	73.26	82.35	74.87	86.63	91.98	91.98	78.07	78.07
st-aus-credit	68.02	68.02	66.28	64.24	68.02	65.12	66.28	64.39	68.02	68.02	64.39	64.39
st-german-credit	78.2	78.1	75.3	77.3	78.1	77.9	75.6	77.4	72.6	72.6	77.9	77.9
st-heart	84.7	84.7	86.94	86.94	84.33	83.58	86.19	87.31	82.09	82.09	85.07	85.07
st-image	94.41	94.41	94.06	94.06	94.28	94.15	94.06	94.02	73.35	73.35	89.38	89.38
st-landsat	85.25	84.65	84.95	85.15	84.6	85.35	85	85.1	73.5	73.5	81.95	81.95
st-shuttle	98.88	98.88	98.91	98.91	99.14	99.1	99.13	99.13	88.07	88.07	96.81	96.81
st-vehicle	80.92	81.28	78.08	78.2	81.64	81.87	77.96	78.2	60.07	60.07	78.79	78.79
steel-plates	73.71	73.51	75	73.45	74.02	73.76	75.05	74.95	55.46	55.46	65.93	65.93
synthetic-control	92.67	93.33	93	95	93.67	93.67	95	95.17	89.5	89.5	91.17	91.17
teaching	50.66	53.29	50	53.95	50.66	50.66	50	53.95	47.37	47.37	49.34	49.34
thyroid	93.82	93.79	93.23	89.99	93.76	93.79	93.79	90.05	93.38	93.38	93.79	93.79
tic-tac-toe	97.7	97.7	97.7	97.7	97.7	97.7	97.7	97.7	74.16	74.16	97.59	97.59
titanic	78.95	78.95	78.95	78.95	78.95	78.95	78.95	78.95	78.95	78.95	78.82	78.82
rains	87.5	87.5	87.5	87.5	87.5	87.5	87.5	87.5	87.5	75	87.5	87.5
twonorm	97.86	97.86	97.82	97.88	97.82	97.85	97.84	97.88	81.62	81.62	97.86	97.86
vc-2classes	82.79	83.12	84.42	86.69	82.79	83.44	86.04	86.69	83.12	83.12	83.12	83.12
vc-3classes	82.14	81.82	82.47	82.47	82.14	82.47	82.79	82.47	82.14	82.14	81.49	81.49
wall-following	82.95	83.06	84.07	84.2	82.24	82.97	83.6	84.09	51.37	51.37	65.76	65.76
waveform	86.36	86.28	86.14	86.12	86.16	86.18	86.18	86.12	74.6	74.6	85.44	85.44
waveform-noise	86.26	86	85.92	86.04	83.98	84.04	86.06	86.04	60.28	60.28	86.16	86.16
wine	99.43	99.43	100	100	99.43	99.43	100	100	90.34	90.34	100	100
w-qua-red	58.63	58.69	59.63	59.44	58.81	58.69	59.19	59.75	55.5	55.5	58.88	58.88
w-qua-white	54.92	54.72	55.11	54.98	54.84	54.76	54.55	55.11	49.71	49.71	54.47	54.47
yeast	59.7	58.96	60.24	59.3	60.51	59.7	60.65	59.3	51.48	51.48	58.09	58.09
zoo	94	94	95	95	95	94	94	94	92	92	93	93

Table B.4 Detailed results of Moore-Penrose pseudoinverse based RVFL variants on *Tribas*, *Radbas* and *Sign* Activation Function

Data	Tribas				Radbas				Sign			
	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$
abalone	65.54	65.54	65.54	65.54	65.85	65.95	66.31	66.26	57.47	57.47	64.68	64.68
ac-inflam	100	100	100	100	100	100	100	100	100	100	100	100
acute-nephritis	100	100	100	100	100	100	100	100	91.67	91.67	100	100
adult	83.1	83.1	83.1	83.1	84.87	84.92	84.78	84.74	77.87	77.87	82.38	82.38
annealing	14	14	14	14	24	46	15	15	65	65	16	16
arrhythmia	59.07	58.63	29.65	30.75	61.06	60.84	43.81	43.58	63.27	63.94	36.95	36.95
audiology-std	56	64	64	64	52	52	56	56	12	12	64	64
balance-scale	89.42	89.42	89.42	89.42	91.03	91.35	90.87	91.03	84.13	84.13	88.46	88.46
balloons	87.5	87.5	62.5	62.5	75	75	81.25	81.25	81.25	81.25	81.25	81.25
bank	89	89	89	89	89.76	89.78	89.71	89.76	88.61	88.61	89	89
blood	79.55	79.55	79.55	79.55	77.27	77.14	78.07	79.41	79.95	79.95	79.28	79.28
breast-cancer	76.76	74.65	76.76	76.76	70.42	69.01	72.89	71.48	71.13	71.13	71.13	71.13
bc-wisc	96.86	96.86	96.86	96.86	97.14	97.14	97.14	97.43	96.71	96.71	97.14	97.14
bc-wisc-diag	95.07	95.42	93.31	93.31	96.3	96.3	92.78	95.6	94.72	94.72	95.42	95.42
bc-wisc-prog	76.53	77.55	80.1	78.06	77.04	77.55	80.1	78.57	77.55	77.55	79.59	79.59
breast-tissue	60.58	57.69	65.38	65.38	68.27	63.46	62.5	66.35	56.73	56.73	68.27	68.27
car	88.37	88.37	88.37	88.37	95.2	95.2	95.43	95.37	75.93	75.93	82.18	82.18
ctg-10classes	71.19	71.19	71.19	71.19	79.05	79.57	79.43	79.61	57.02	57.02	68.83	68.83
ctg-3classes	89.27	89.27	89.27	89.27	91.2	91.48	91.01	90.87	85.45	85.45	88.56	88.56
chess-krvk	28.1	28.1	28.1	28.1	47.57	47.55	47.56	47.54	19.81	19.81	26.74	26.74
chess-krvkp	93.87	93.87	94.52	94.52	94.43	94.46	95.24	95.24	73.56	73.56	94.46	94.46
congress-voting	59.17	61.01	59.86	59.63	60.09	60.78	61.01	61.01	61.47	61.47	59.17	59.17
conn-bench-sonar	77.4	77.4	77.4	78.37	70.19	70.19	70.19	71.63	73.56	73.56	69.71	70.19
conn-bench-vowel	89.61	89.61	89.61	89.61	96.54	96.54	83.12	95.89	66.67	66.67	71.65	71.65
connect-4	75.5	75.5	75.5	75.5	75.9	75.91	76.45	76.46	75.38	75.38	75.44	75.44
contrac	54.42	54.42	54.42	54.42	54.62	53.6	53.6	53.74	49.93	49.93	52.92	52.92
credit-approval	86.77	86.77	87.06	86.92	85.9	85.61	86.92	87.5	80.09	80.09	86.92	86.92
cylinder-bands	74.41	74.8	75.2	75	76.17	76.76	75	75	69.53	69.53	69.53	69.53
dermatology	98.35	98.35	97.8	97.8	97.8	97.53	97.8	97.8	89.29	89.29	97.8	97.8
echocardiogram	84.09	80.3	84.85	84.85	78.03	84.09	87.12	86.36	77.27	77.27	84.09	84.09

Continued on next page

Table B.4 – continued from previous page

Data	Tribas				Radbas				Sign			
	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$
ecoli	84.82	84.82	84.82	84.82	87.5	86.61	86.31	87.5	78.87	78.87	84.82	84.82
energy-y1	91.28	91.28	91.28	91.28	94.79	94.79	93.49	95.05	83.85	83.85	86.59	86.59
energy-y2	87.76	87.76	87.76	87.76	89.71	89.84	90.36	91.93	84.77	84.77	83.2	83.2
fertility	88	88	88	88	88	88	88	88	88	88	88	88
flags	54.17	50.52	45.83	48.44	45.31	45.83	47.4	46.35	35.42	35.42	44.79	44.79
glass	60.85	60.85	60.85	60.85	63.68	62.74	67.45	67.45	60.38	60.38	64.15	64.15
haberman-survival	75.99	75.99	75.99	75.99	73.36	73.03	74.01	73.36	69.74	69.74	74.34	74.34
hayes-roth	82.14	82.14	78.57	78.57	82.14	78.57	85.71	78.57	71.43	71.43	67.86	71.43
heart-cleveland	59.54	59.21	60.86	60.53	63.16	61.84	61.51	61.51	60.2	60.2	59.87	59.87
heart-hungarian	83.56	83.56	83.56	83.56	83.22	82.19	84.25	83.56	77.4	77.4	84.59	84.59
heart-switzerland	39.52	35.48	40.32	40.32	50.81	46.77	52.42	39.52	42.74	43.55	25.81	25.81
heart-va	32.5	33.5	33.5	33.5	33	36	37.5	36.5	29.5	29.5	31.5	31.5
hepatitis	78.21	79.49	80.77	82.05	76.28	77.56	80.77	76.28	79.49	79.49	76.92	76.92
hill-valley	71.12	69.64	72.44	72.94	68.32	70.13	34.82	34.82	49.34	49.34	72.77	72.77
horse-colic	42.65	52.94	67.65	69.12	41.18	52.94	47.06	66.18	54.41	54.41	66.18	66.18
ilpd-indian-liver	72.77	72.77	72.26	72.26	72.77	71.75	73.8	72.6	72.09	72.09	72.26	72.26
image-segmentation	19.43	22.24	0.19	0.19	14.29	14.29	26.57	3.29	12.33	21.14	19.43	19.43
ionosphere	87.5	88.64	87.5	87.5	89.49	89.77	87.22	88.35	86.36	86.36	88.64	88.64
iris	97.97	97.97	97.97	97.97	97.97	97.3	97.3	97.3	94.59	94.59	97.3	97.3
led-display	73.1	73.1	73.1	73.1	74	73.9	73.7	73.6	69.3	69.3	72.1	72.1
lenses	70.83	33.33	33.33	33.33	75	62.5	79.17	62.5	87.5	87.5	87.5	87.5
letter	62.89	62.89	62.89	62.89	82.76	82.74	83.49	83.56	21.84	21.84	58.07	58.07
libras	73.61	74.72	47.22	45	73.89	76.39	31.39	29.17	61.11	61.11	56.67	56.67
low-res-spect	87.03	86.65	85.53	87.22	87.97	87.97	83.27	88.16	80.83	80.83	82.71	83.08
lung-cancer	37.5	40.63	50	46.88	40.63	37.5	43.75	46.88	59.38	59.38	59.38	59.38
lymphography	81.08	79.73	70.95	70.95	84.46	82.43	82.43	82.43	72.97	72.97	87.84	87.84
magic	80.93	80.93	80.93	80.93	85.67	85.66	85.67	85.68	73.28	73.28	79.46	79.46
mammographic	82.29	82.29	82.29	82.29	81.46	81.56	81.46	81.35	76.15	76.15	82.19	82.19
miniboone	90.25	90.25	90.25	90.25	91.27	91.32	91.6	91.6	81.24	81.24	90.13	90.13
molec-biol-promoter	67.31	67.31	78.85	70.19	78.85	77.88	63.46	63.46	71.15	71.15	71.15	61.54
molec-biol-splice	81.37	81.27	80.93	80.93	81.56	81.34	81.4	81.18	58.91	58.91	80.33	80.33
monks-1	51.62	51.62	51.62	51.62	54.17	55.56	59.26	50.93	55.09	55.09	62.5	62.5
monks-2	67.13	67.13	65.51	65.51	66.67	66.44	67.36	68.29	67.36	67.36	66.44	66.44
monks-3	50.46	50.46	50.46	50.46	44.21	37.96	43.98	46.06	53.01	53.01	55.32	55.32
mushroom	96.48	96.48	96.48	96.48	100	100	100	100	84.28	84.28	95.06	95.06
musk-1	78.99	78.99	71.22	71.01	81.3	80.67	64.29	63.66	73.32	73.32	74.79	75
musk-2	94.44	94.57	94.89	94.89	94.6	94.33	95.98	96.06	87.17	87.17	94.42	94.42
nursery	89.11	89.11	89.17	89.17	95.51	95.52	95.52	95.81	57.48	57.48	89.17	89.17
oocMer12F	82.84	82.55	84.02	84.02	84.22	86.27	85.49	85.49	70.88	70.88	80.78	80.78
oocMer14D	91.18	91.37	91.57	91.57	92.94	91.86	91.76	91.67	89.41	89.41	91.08	91.08
oocTris2F	82.13	82.13	82.13	82.13	84.21	84.1	85.09	85.2	69.08	69.08	78.73	78.73
oocTris5B	92.21	92.43	92.43	92.43	93.42	93.97	92.43	92.32	83.99	83.99	91.89	91.89
optical	92.77	92.71	91.65	91.6	95.21	94.99	95.6	95.77	63.49	63.49	92.49	92.49
ozone	97.16	97.16	97.16	97.16	97.16	97.16	97.08	97.08	97.16	97.16	97.08	97.08
page-blocks	95.27	95.27	95.27	95.27	95.65	95.56	95.72	95.67	92.73	92.73	94.63	94.63
parkinsons	84.18	86.22	87.76	87.76	84.69	85.2	88.27	85.71	80.1	80.1	87.24	87.24
pendigits	88.08	88.08	88.08	88.08	96.54	97.11	96.6	96.66	62.52	62.52	83.56	83.56
pima	78.13	78.13	78.13	78.13	77.21	76.17	76.04	76.56	72.27	72.27	76.69	76.69
pb-MATERIAL	89.42	89.42	91.35	90.38	92.31	90.38	91.35	90.38	68.27	68.27	92.31	92.31
pb-REL-L	67.31	67.31	69.23	68.27	67.31	68.27	65.38	69.23	71.15	71.15	71.15	71.15
pb-SPAN	66.3	66.3	69.57	69.57	65.22	61.96	64.13	67.39	54.35	54.35	68.48	68.48
pb-T-OR-D	88	87	86	88	88	86	84	84	86	88	85	87
pb-TYPE	62.5	62.5	63.46	62.5	64.42	63.46	64.42	59.62	53.85	53.85	66.35	66.35
planning	71.11	71.11	66.67	66.67	71.11	71.11	67.22	65.56	71.11	71.11	68.33	68.33
plant-margin	68.63	68.44	68.56	68.56	76.75	76.69	77.38	77.25	38.06	38.06	61.69	61.69
plant-shape	54.38	54.38	53.88	53.88	54.31	53.94	54.63	54.81	34.56	34.56	45.31	45.31
plant-texture	74.19	74.19	73.19	73.25	78	78.25	79.75	79.88	48.94	48.94	67.25	67.25
post-operative	71.59	71.59	70.45	68.18	71.59	72.73	70.45	70.45	70.45	72.73	70.45	70.45
primary-tumor	50.61	50.3	55.49	55.49	51.22	52.74	55.49	54.88	42.07	42.07	51.52	51.52
ringnorm	92.42	92.42	92.42	92.42	94.96	94.97	95.27	95.24	79.07	79.07	87.11	87.11
seeds	94.23	94.23	96.15	96.15	94.71	95.19	95.19	95.67	87.98	87.98	95.67	95.67
semeion	84.3	84.55	83.29	84.86	84.67	86.18	83.86	83.79	69.1	69.1	82.35	82.35
soybean	53.46	52.66	44.95	44.95	55.85	57.18	48.14	44.15	32.98	32.98	48.94	48.94
spambase	90.37	90.41	91.07	91.04	90.76	90.74	90.7	90.74	75.24	75.24	90.04	90.04
spect	63.44	59.68	52.69	52.15	59.14	59.14	45.7	40.86	58.6	58.6	58.06	58.06
spectf	90.37	55.61	80.75	81.28	87.17	85.56	65.24	68.98	90.91	90.91	78.07	78.07
st-aus-credit	68.02	68.02	67.01	67.01	68.02	68.02	66.28	67.01	68.02	68.02	64.39	64.39
st-german-credit	77.2	77.3	76.9	77	76.8	74	77.3	77.1	72.6	72.6	77.9	77.9
st-heart	83.58	83.58	86.94	86.19	85.07	85.82	88.06	86.94	82.09	82.09	85.07	85.07
st-image	89.38	89.38	89.38	89.38	94.71	94.67	94.63	94.63	73.35	73.35	89.38	89.38
st-landsat	83.1	83.1	83.1	83.1	84.95	84.9	85.25	85.15	73.5	73.5	81.95	81.95
st-shuttle	96.47	96.47	96.47	96.47	99.29	99.3	99.36	99.36	88.07	88.07	96.81	96.81
st-vehicle	76.54	76.54	76.54	76.54	79.98	79.74	80.92	81.75	60.07	60.07	78.79	78.79
steel-plates	70.77	70.77	70.77	70.77	74.12	74.18	74.18	73.2	55.46	55.46	65.93	65.93
synthetic-control	93	93.5	92	92	92.67	92.67	95.83	95.83	89.5	89.5	91.17	91.17
teaching	52.63	52.63	54.61	54.61	56.58	53.95	55.26	51.97	47.37	47.37	49.34	49.34

Continued on next page

Appendix B. Results of RVFL variants

Table B.4 – continued from previous page

Data	<i>Tribas</i>				<i>Kadbas</i>				<i>Sign</i>			
	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$	$-b, -d$	$+b, -d$	$-b, +d$	$+b, +d$
thyroid	93.82	93.79	93.82	93.79	93.79	93.73	93.82	93.82	93.38	93.38	93.7	93.7
tic-tac-toe	97.59	97.59	97.59	97.59	97.8	97.8	97.59	97.91	74.16	74.16	97.59	97.59
titanic	78.95	78.95	78.95	78.95	78.95	78.95	78.95	78.95	78.95	78.95	78.82	78.82
rains	87.5	62.5	100	87.5	37.5	37.5	87.5	87.5	87.5	75	87.5	87.5
twonorm	97.8	97.8	97.8	97.8	97.92	97.72	97.85	97.85	81.62	81.62	97.86	97.86
vc-2classes	81.82	81.82	84.09	84.09	86.36	85.71	85.06	85.06	83.12	83.12	83.12	83.12
vc-3classes	81.49	81.49	81.82	81.82	84.74	83.44	84.09	84.09	82.14	82.14	81.49	81.49
wall-following	70.8	70.8	70.8	70.8	82.84	83.14	82.42	83.19	51.37	51.37	65.76	65.76
waveform	85.1	85.1	85.44	85.44	86.18	86.1	85.86	85.94	74.6	74.6	85.44	85.44
waveform-noise	84.94	84.88	86.2	86.2	84.92	84.92	86.04	86.06	60.28	60.28	86.16	86.16
wine	98.3	98.3	100	99.43	100	98.86	100	99.43	90.34	90.34	100	100
w-qu-a-red	60	60	60	60	59.38	59.63	58.88	59.13	55.5	55.5	58.88	58.88
w-qu-a-white	53.55	53.55	53.55	53.55	54.58	54.51	55.27	54.29	49.71	49.71	54.47	54.47
yeast	57.82	57.82	57.82	57.82	60.31	59.77	60.11	59.77	51.48	51.48	58.09	58.09
zoo	98	98	98	98	94	94	97	97	92	92	92	92

"-" and "+" represent without and with, respectively. "b" and "d" stand for bias and direct link, respectively.