Randomized Ensemble Learning with Heterogeneous Features for Large-Scale Data Analytics

Submitted by

Caihao Cui

Master of Engineering, 2014 Harbin University of Science and Technology

Bachelor of Engineering, 2011 Harbin University of Science and Technology

A thesis submitted in total fulfilment
of the requirements for the degree of
Doctor of Philosophy

Department of Computer Science and Information Technology
School of Engineering and Mathematical Sciences
College of Science, Health and Engineering

La Trobe University

Victoria, Australia

October 2018

Statement of Authorship

Caihao Cui

October 2018

Acknowledgements

First, I would like to thank my academic supervisor, Reader and Associate Professor Dianhui Wang, for his professional academic training and technical support. I benefited not only from his deep insights and knowledge but also from his professionalism as a respectable academic. I also appreciate his work ethic, personal integrity and career guidance in helping me to grow up in both the academic and industrial communities. These are the most valuable things that I learned over the past four years.

I also want to thank our research group members, including Dr Ming Li for checking the programs and for participating in interesting discussions on machine learning algorithms and sharing academic papers and also thanks to the two visiting scholars, Dr Xuemei Dong and Mr Bin Wang, and university statistics consultant, Dr Xia Li, for sharing their knowledge on probability and statistical learning theory in the group seminars. Additionally, I want to thank Dr Slaven Marusic and his team in Aurecon Digital Group where I learned a lot during the APR.Intern project between Nov-2017 to Mar-2018.

I would also like to thank my parents and friends for their support and hospitality. In particular, I want to thank my wife and best friend, Yue Chen, for her constant understanding and for putting up with me over the years in Melbourne.

Finally, I would like to acknowledge the support of the La Trobe University Full-Fee Research Scholarship (LTUFFRS) which covered the four years' tuition fees for my doctoral candidature, and the La Trobe University Postgraduate Research Scholarship (LTUPRS) for three-and-a-half years living allowance which enabled me to focus on this research.

Publication List

- **C. Cui**, D. Wang, High dimensional data regression using Lasso model and neural networks with random weights, *Information Sciences*, (372) 505-517, 2016.
- D. Wang, **C. Cui**, Stochastic configuration networks ensemble with heterogeneous features for large-scale data analytics, *Information Sciences*, (417) 55-71, 2017.
- **C. Cui**, D. Wang, Lasso stochastic configuration network ensemble for high dimensional data regression, *Knowledge-Based Systems*, 2018 (to be submitted).
- D. Wang, **C. Cui**, Evolving stochastic configuration networks for power demand forecasting of fused magnesium furnace, *IEEE Transactions on Industry Informatics*, 2018 (to be submitted).

Abstract

This thesis develops randomized ensemble learning with heterogeneous features for large-scale data analytics. Building models quickly on large-scale data with a high degree of accuracy is significant today because of the vast number of real-world applications from various domains, such as bioinformatics and engineering. After an extensive review of randomized methods for building neural networks and ensemble learning techniques, we propose three randomized ensemble learning frameworks, where the stochastic configuration networks (SCNs) are employed as the base learner models. The SCN-based ensemble learning systems (SCNE) are built on the dataset with heterogeneous features and then aggregated by different approaches including bagging, boosting and negative correlation learning. Two iterative algorithms are designed to solve the computing issue when the ensemble system is too large to be solved by the classic pseudo-inverse method. To solve the high dimensional data regression problem, the Lasso-SCNE framework is proposed with ensemble feature generation and hidden node pruning techniques to maintain the balance between model interpretation and prediction accuracy. To further improve the performance of the ensemble system and optimise the node controllable SCN model, we propose the evolving-SCNE framework, taking advantage of the genetic operations to upgrade the ensemble system generation by generation. The experiment results on industrial applications and simulations demonstrate some merits of the proposed SCNE frameworks and algorithms for dealing with large-scale data modelling problems.

Keywords: ensemble learning, randomized neural networks, stochastic configuration network ensemble, bagging, boosting, negative correlation learning, Lasso, genetic algorithms, heterogeneous features.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning has received a large amount of attention over the past years due to its significant role in data analytics [41]. People today greatly benefit from the new technologies relating to data storage and transmission and the move towards modern digitalisation in industry. Data collection and storage today is much easier and far more cost effective than ever before. Hence, data is being collected at a very rapid rate, even though it may not be clear as to how it will be used. Traditional machine learning methods cannot cope with this large volume of data and the data redundancy problem hence, advanced algorithms and innovative frameworks are expected to gradually address these issues to provide more insights for researchers and enhance the productivity of industry [31, 41, 87].

Various ensemble learning frameworks have been proposed and developed over the last two decades to address this challenge. Many innovative ideas and theoretical works, including bagging, boosting, AdaBoost and random forests [9, 10], have been devoted to improving the generalization ability of ensemble learning systems with parallel applications.

Recall that the basis of ensemble learning theory starts from building many learners (base models) with a rational sampling implementation and then combines the predictability of all the learners as the final ensemble output following two specific strategies. One is on the model side, improving the diversity of the learners in the ensemble using different types of learners. The other is on the data side, for example, the bagging method, training the learner on different sub-datasets randomly selected from the original dataset; and the boosting method, training the learner sequentially with weighted samples based on the previous ensemble output [12, 17, 34, 84, 85].

However, instead of building the same type of learners independently, the negative correlated learning (NCL) strategy builds the learners with influence from other learners to increase the diversity of the ensemble. This strategy may compromise the ability of an individual learner, but the ensemble output accuracy is improved significantly. Moreover, there is another approach for improving the diversity on the feature side, that is, changing the features of the training dataset of every learner. The random forest (RF) approach applies this idea but in a brute way, randomly assigning different features for every decision tree model.

With regard to neural network ensembles, the base models are trained by the well-known error backpropagation (BP) algorithm. Various BP-based learning methods have shown their effectiveness in iteratively training a network, but in practice, it is still a challenge to estimate an appropriate architect for a network model. Furthermore, it is even more time-consuming to apply the existing ensemble framework for large-scale data sets based on these BP-based neural network models.

Randomized learning techniques have been extensively explored in relation to neural network modelling since the 80s and 90s, in terms of different perspectives and/or in various contexts, including the expanded presentation for perceptron [94], radial basis function (RBF) networks [58], single hidden layer feed-forward neural networks (SLFNs) [88], random vector functional-link (RVFL) networks [68, 69, 70, 116], kernel approximation [75] and reservoir computing [60] etc. Based on the function approximation theory, both the randomized RBF networks and RVFL networks belongs to random basis approximators (RBAs) [92].

The basic idea behind these methods on training one structure fixed neural networks comprises two steps: first, randomly assigning the input weights and biases of the nodes of the hidden layers; then, computing the weights of the final layer from the linear equation system using the well-known least squares method and its variants with regularizations.

In [39], Igelnik and Pao mathematically proved that an RVFL network could be a universal approximator with probability one for continuous maps on compact sets, with random input weights and biases of hidden nodes chosen through a uniform distribution in proper scopes. In [98], Ivan et al. empirically demonstrated that the trained RVFL net could not approximate the target function with a high probability when an inappropriate scope is used for assigning the weights and biases of the hidden nodes. Thus, the key parameters to design RVFL networks are the number of hidden nodes and the scope of the random parameters. In [49], Li and Wang point out the common mistakes made by people when employing RVFL models in applications, for example, the commonly used scope range $[-1, +1]$ is not optimal, thereby it should be data-dependent and estimated by experiments. To address these issues, Wang and Li proposed the stochastic configuration networks (SCN) in [105]. The proposed SCN modelling method is different from the existing learning methods for single-layered feed-forward networks. Instead of a fixed structure, the SCN model is an incremental neural network constructed by adding selected nodes which are subjected to an inequality constraint. The selected nodes are from a candidate node pool whose weights and biases are assigned following an adaptive scheme of the scopes. In [104], Li and Wang proposed two new methods that extended the ability of SCN to uncertainty data based on the kernel density estimation (KDE) method and the maximum correntropy criterion (MCC), respectively. Then, they proposed a deep SCN in [103] which improved the stochastic configuration algorithm in a generalized form.

In [2], Alhamdoosh and Wang employed RVFL networks as base learners to develop a fast decorrelated neuro-ensemble (termed DNNE) aiming at applying a randomized model in ensemble learning. From the experiment results, DNNE performs well on smaller data sets. However, it is quite limited when dealing with large-scale data because of its high computational complexity, the scalability of numerical algorithms for the least squares solution, and hardware constraints. In addition to these disadvantages of the RVFL modes, the main shortcomings of DNNE come from the two aspects: (i) the system inputs are centralized or combined with different types of features; and (ii) the analyzed method of computing the output weights becomes infeasible for large-scale data sets, which is related to the nature of the base learner model. Moreover, in the real world, data can be gathered from different sources or data storage centers. Some features of these data may be extracted by certain feature selection algorithms in the data cleaning or pre-processing procedure even before they are passed to build the models.

The main part of this thesis is devoted to the proposal of stochastic configuration network ensemble (SCNE) framework as a randomized ensemble learning technique with heterogeneous features. The advanced SCN model is employed as the base learner in the different aggregation approaches. The Lasso-SCNE framework is proposed aim at solving the high-dimensional regression problem and reducing the number of the hidden nodes to improve the generalization ability of the neuro-ensemble. The further evolving SCNE framework is based on genetic algorithms to upgrade the SCNE for industrial data modeling task, meanwhile providing an approach to obtain an optimal SCN model and control the hidden node number.

The stochastic configuration networks ensemble (SCNE) framework is different from the existing neural network ensemble as it applies the SCN as the base learner in the ensemble system. Based on traditional ensemble learning, we first adapt the bagging and boosting strategies for SCNE, and then propose the extended SCNE with heterogeneous features and the negative correlation learning (NCL) strategy. Thus, SCNE is a more general form of the randomized ensemble framework, where a set of input features are fed into the SCN base models separately. Two iterative methods, named the Block-Jacobi method and Block-GaussSeidel method, are selected as feasible solutions to evaluate the output weights of the learners in SCNE. In addition, an analysis and discussion on the convergence of these iterative schemes is given through a demonstration on the correlations among the iterative solutions and the pseudo-inverse solution.

SCNE mostly focuses on the combination methods and diversity of the learners. But the randomness of the building process of the SCN models could lead to redundant nodes in their hidden layers. It may not compromise the performance of the ensemble, but it increases the model size and computing cost.

Recall that the basic philosophy of and motivation for applying the randomized algorithm on building neural networks (SCNs) is to accelerate the training process and obtain plausible models compared with the ones trained by the time-consuming and complex optimization methods. This fact also reflects the uncertainty of the randomized model, which may potentially lead to the unstable performance of SCNs. The SCNE framework decreases uncertainty by using the NCL strategy, but the hidden nodes of the pretrained SCN models in the ensemble have been added and fixed during the model combination. The redundant nodes are still kept inside the ensemble, making it overstaffed. It is hard to judge whether they are good or not for the generalization ability of the ensemble, but they certainly increase the computing cost and size of the ensemble.

Thus, we propose two new methods, Lasso-SCNE and evolving SCNE (eSCNE), from the perspective of SCN node ensemble instead of the original SCN model ensemble. They are both motived by the idea of the selective ensemble strategy, but they select the hidden nodes generated from a pool of pretrained SCN models instead of models.

The Lasso-SCNE applies the Lasso method with l_1-regularization to limit the size of the hidden nodes in the final ensemble by estimating all the hidden nodes at the same time. The cross-validation method is used here to estimate the regulating factor with respect to accuracy. Then, this method is generalized specifically into a framework for high dimensional data regression, where model interpretation and prediction accuracy are regularized. In this framework, the Lasso method can be used twice, one as a feature selector for the generation of ensemble features with selective regularizing factors estimated via a cross-over procedure; the other to get the Lasso-SCNE from the node pool from the pertained SCNs on the ensemble features. Experiments with comparisons of estimating the protein content of milk from its NMR spectrum are carried out by a data set with 31,570 dimensions (spectrum size) and 120 samples. The results demonstrate that our proposed solution for data regression problems with small samples and high dimensionality is promising, and the learning system performs robustly with respect to a key parameter setting in the ensemble feature generation.

The eSCNE uses the modified genetic algorithm (GA) with node size control and a novel design of the fitness function, in which the fraction rank values and condition numbers of the SCNs hidden layer output matrices are added. This technique also provides a new way to build SCNE. Another contribution is the node index matrix which is proposed in the method to represent the SCN models and nodes, making the GA optimization (selection, crossover, mutation, etc) processing visible. It also helps to add the limitation of the mutation range easily to avoid generating poor SCNs in the breeding generation. Empirical studies support our long-held conjecture that the algebraic properties of the output matrix of SCNs are strongly correlated to the SCNs final generalization abilities. A comprehensive comparison of experiments using a real industrial task on forecasting the power demand of four fused magnesium furnace (FMF) machines is undertaken. The experiment results demonstrate that the eSCNE method outperforms the original techniques in building SCN models and SCNE from several perspectives.

Structure of the Thesis

Chapter 1 introduces the background, developments and challenges of the randomized ensemble learning on modern data analytics and briefly describes the main contributions of this thesis including the SCNE, Lasso-SCNE and eSCNE frameworks.

Chapter 2 reviews the developments of ensemble learning, including bagging, boosting and negative correlation learning. It also provides the technical support for feature extraction methods of ensemble learning and revisits the randomized algorithm for neural networks. Moreover, it briefly discuss the existing randomized networks ensemble framework, DNNE, and introduce the genetic algorithm.

Chapter 3 proposes the framework of SCNE with heterogeneous features, including the modified bagging, boosting and NCL strategies. It also highlights the generalization error estimation in evaluating the ensemble model. For large-scale data analytics with heterogeneous features, SCNE employs the block Jacobi and Gauss-Seidel methods to iteratively evaluate the output weights of the SCNs. Additionally, a convergence analysis is given with a demonstration on the uniqueness of these iterative solutions. The results of two experiments on real datasets also show the superiority of SCNE over DNNE networks for prediction.

Chapter 4 extends the SCNE to Lasso-SCNE based on the Lasso method to reduce the redundant nodes in the learners, aiming to downsize the SCNE and improve the performance. This method is generalized specifically into a framework for high dimensional data regression with the generation of ensemble features and SCNE node selection. The experiment results on the NMR datasets demonstrate that the proposed solution for data regression problems with small samples and high dimensionality is promising, and the learning system performs robustly.

Chapter 5 describes a new development of evolving SCNE (eSCNE) that uses the genetic algorithm (GA) to upgrade an SCNE with the hidden node size control of the learners from the pre-trained SCNE. The resulting SCNE performs better than the original one. It can be used as an approach to generate an optimal SCN model. A comprehensive comparison is carried out on a task of forecasting the power demand for FMF machines. The results demonstrate that the eSCNE outperforms the original SCN models and SCNE from several perspectives.

Chapter 6 concludes the main contributions of this thesis, along with possible further developments and applications.

Notation:

- $\mathbb{R}$: The set of real numbers;
- $\mathbb{R}^+$: The set of positive real numbers;
- $\mathbb{R}^d$: d-dimensional Euclidean space;
- $\{0,\ 1\}$: The set containing 0 and 1;
- $\{0,\ 1, \cdots, n\}$: The set of all integers between 0 and n;
- a_i : Element i of vector a;
- $A_{N\times p}$: Matrix A, N rows and p columns;
- $A_{i,j},\ A_{ij}$: Element i,j of matrix A;
- $A_{i,:}$: Row i of matrix A;
- $A_{:,j}$: Column j of matrix A;
- p_{data} : The data generating distribution;
- $\boldsymbol{x}^{(i)}$: The i-th example (input) from a dataset;
- $y^{(i)} or\ \boldsymbol{y}^{(i)}$: The target associated with $\boldsymbol{x}^{(i)}$ for supervised learning;
- $X_{N\times p}$: The $N \times p$ matrix with input example $\boldsymbol{x}^{(i)}$ in row $X_{i,:}$;

- A^T: Transpose of matrix A;
- A^{-1}: Inverse of matrix A;
- $A^{\dagger}$: Moore-Penrose pseudoinverse of matrix A;
- $\frac{dy}{dx}$: Derivative of y with respect to x;
- $\frac{\partial y}{\partial x}$: Partial derivative of y with respect to x;
- $P(a)$: A probability distribution over a discrete variable;
- $p(a)$: A probability distribution over a continuous variable, or over a variable whose type has not been specified;
- $E_{x \sim P}\left[f\left(x\right)\right]$ or $E\left\{f\left(x\right)\right\}$: Expectation of $f(x)$ with respect to $P(x)$;
- $Var(f(x))$: Variance of $f(x)$ under P(x);
- $Cov\left(f\left(x\right),\ g\left(x\right)\right)$: Covariance of $f\left(x\right)$ and $g(x)$ under $P(x)$;
- $f(\boldsymbol{x};\ \boldsymbol{\theta})$: A function of $\boldsymbol{x}$ parametrized by $\boldsymbol{\theta}$. (Sometimes we write $f(\boldsymbol{x})$ and omit the argument $\boldsymbol{\theta}$ to lighten notation);
- $log(x)$: Natural logarithm of x.

Chapter 2

Related Work and Technical Support

This chapter is devoted to the exposition of the related works on the history and the technical developments of the randomized ensemble for machine learning and data analytics. Section 2.1 reviews the basic concepts of ensemble methods and strategies with the theoretical foundations and algorithms. These techniques can be modified to build a neural network ensemble with randomized methods. Section 2.2 revisits the literature and developments of feature engineering. It summarizes several wildly-used feature construction, selection and extraction methods for ensemble learning and the representation learning with auto-encoders, restricted Boltszman machine and convolutional neural networks. Section 2.3 reviews the randomized algorithm for building neural networks. It begins with the most commonly used neural networks to random vector functional-link (RVFL) networks, and the newly developed stochastic configuration networks (SCN) and its variations. Section 2.4 discusses the existing randomized neural network ensemble models, decorrelated neuro-ensemble (DNNE), and points out its limitations and shortcomings for the requirements of modern data analytics. In Section 2.5, the central concepts and steps of genetic algorithms are introduced and how they can be used in optimizing the learners of an ensemble system is demonstrated. In summary, this chapter discusses and comments on the existing work, leading to the proposed randomized ensemble framework of this thesis.

2.1 Ensemble Learning

The major task of machine learning, pattern recognition, and data mining is to construct good models from datasets. The basic premise of machine learning from data is to use a set of observations (samples) to uncover an underlying process, but it is a broad premise, and difficult to fit into a single framework. Thus, different learning paradigms have arisen to deal with different situations and assumptions.

One of the most critical issues for learning systems is to construct a learner with high generalization performance. The most popular way is to find the best model for a target task. In general, it is difficult to estimate the best model with given finite samples. On the other hand, compared to that obtained by a single model, combining multiple regression learners has been shown to improve generalization performances [3, 40, 108, 109].

Ensemble learning is the process by which multiple models or learners are strategically generated and combined to solve a computational intelligence problem [22, 119]. Ensemble learning is primarily used to improve the performance of a model for classification, prediction, function approximation, etc., or reduce the likelihood of the unfortunate selection of a poor one. Other applications of ensemble learning include assigning confidence to the decision made by the model, selecting optimal (or near optimal) features, data fusion, incremental learning, non-stationary learning and error-correction.

An ensemble-based system is obtained by combining diverse models. Therefore, such systems are also known as multiple-model ensemble systems or just ensemble systems. There are several scenarios where using an ensemble-based system makes statistical sense, which is discussed below in detail. However, to fully and practically appreciate the importance of using ensemble systems, it is instructive to look at a psychological backdrop to this otherwise statistically sound argument: we use such an approach routinely in our daily lives by asking the opinions of several experts before making decisions or drawing conclusions.

In contrast to common learning approaches which try to construct only one learner from training data, ensemble methods try to construct a set of learners and combine them to solve the same problem. Figure 2.1 shows a common ensemble architecture. An ensemble contains some learners called base learners. Base learners are usually generated from training data by a base learning algorithm which can be a decision tree, neural network or other kinds of learning algorithms, as shown in Figure 2.2. Most ensemble methods use a single base learning algorithm to produce homogeneous base learners, i.e., learners of the same type, leading to **homogeneous ensembles**, but there are also some methods which use multiple learning algorithms to produce **heterogeneous learners**, i.e., learners of different types, leading to heterogeneous ensembles. In the latter case, there is no single base learning algorithm and thus, some people refer to the learners as individual learners or component learners or base learners.

There are three threads of early contributions that led to the current area of ensemble methods; that is, combining classifiers, ensembles of weak learners and a mixture of experts.

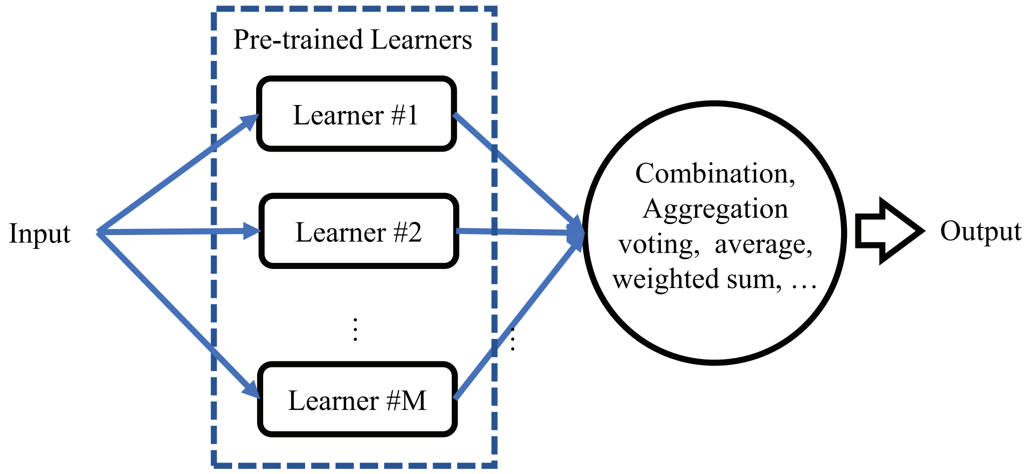

Figure 2.1: A common ensemble architecture

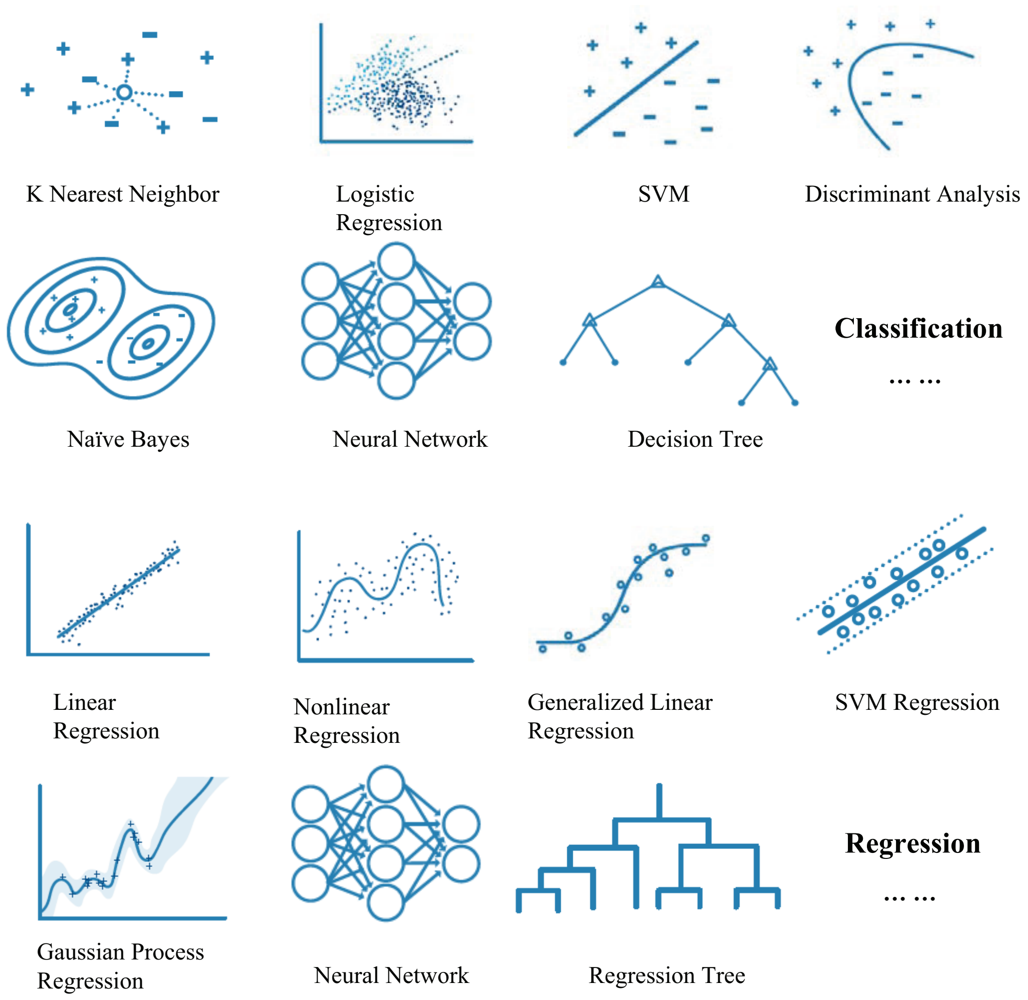

Figure 2.2: Common classification and regression models.

- Combining classifiers was mostly studied by the pattern recognition community. In this thread, researchers generally work on strong classifiers and try to design powerful combining rules to obtain stronger combined classifiers. As a consequence, this thread of work has accumulated a deep understanding of the design and use of different combining rules.

- Ensembles of weak learners were mostly studied by the machine learning community. In this thread, researchers often work on weak learners and try to design powerful algorithms to boost the performance from weak to strong. This thread of work has led to the emergence of popular ensemble methods such as bagging [9] and AdaBoost [27], etc., and a theoretical understanding of why and how weak learners can be boosted to become strong ones.

- A mixture of experts [4, 12, 61] was mostly studied by the neural networks community. In this thread, researchers generally consider a divide-and-conquer strategy, try to learn a mixture of parametric models jointly and use combining rules to obtain an overall solution.

Ensemble methods have been a significant learning paradigm since the 1990s, due to two pieces of pioneering work. One is [34], in which it was found that predictions made by the combination of a set of classifiers are often more accurate than predictions made by the best single classifier. The other is theoretical [86] in which it was proved that weak learners could be boosted to strong learners. Since strong learners are desirable yet difficult to achieve, whereas weak learners are easy to obtain in real practice, this result opens the promising direction of generating strong learners by ensemble methods.

According to how the base learners are generated, there are two paradigms of ensemble methods, that is, sequential ensemble methods where the base learners are generated sequentially, with AdaBoost [43, 86] as a representative, and parallel ensemble methods where the base learners are generated in parallel, with bagging [9] as a representative.

Remark: After generating a set of base learners, rather than trying to find the best single learner, ensemble systems resort to finding combinations of the leaner set to achieve a strong generalization ability, where the combination method plays a crucial role. In [22], Dietterich listed three fundamental reasons for the benefits:

- **Statistical issue**: It is often the case that the hypothesis space is too large to explore when there is limited training data and there may be several different hypotheses giving the same accuracy on the training data. If the learning algorithm chooses one of these hypotheses, there is a risk that a mistakenly chosen hypothesis will not predict the future data well.

- **Computational issue**: Many learning algorithms perform a local search that may get stuck in local optima. Even if there are enough training data, it may still be challenging to find the best hypothesis. By running a local search from many different starting points, the combination may provide a better approximation to the true unknown hypothesis.

- **Representational issue**: In many machine learning tasks, the true unknown hypothesis can not be represented by any hypothesis in the hypothesis space.

These three issues are among the most important reasons why traditional learning approaches fail. A learning algorithm that suffers from the statistical issue is generally said to have a high "variance", a learning algorithm that suffers from the computational issue can be described as having a high "computational variance", and a learning algorithm that suffers from the representational issue is generally said to have a high "bias" [46, 49, 61, 108, 117]. Therefore, through combination, the variance, as well as the bias of learning algorithms, may be reduced which has been confirmed by many empirical studies [40, 68, 74, 89].

Suppose there are M individual learners $\{f_1,\ f_2, \cdots f_M\}$, let $f_{ens}^{(M)}$ be the ensemble system. The most commonly used combination methods are as follows:

- **Averaging** is the most popular and fundamental combination method for numeric outputs where the task is to combine learners to attain the final prediction on the real-valued variable. It includes:

  - **Simple Averaging**: $f_{ens}^{(M)} = \frac{1}{M}\sum_{m=1}^{M} f_m$;
  - **Weighted Averaging**: $f_{ens}^{(M)} = \sum_{m=1}^{M} a_m f_m$, and $a_m \geq 0$, $\sum_{m=1}^{M} a_m = 1$.

- **Voting** is the most popular and fundamental combination method for nominal outputs (labels) where the task is to combine learners to predict the class label from a set of possible class labels. The output of the base learner could be a vector with a crisp label or class probability. It includes:

  1. **Majority Voting**: every classifier votes for one class label, and the final output class label is the one that receives more than half of the votes; if none of the class labels receives more than half of the votes, a rejection option will be given, and the combined classifier makes no prediction.
  2. **Plurality Voting**: In contrast to majority voting which requires the final winner to take at least half of the votes, plurality voting takes the class label which receives the largest number of votes as the final winner.

3. **Weighted Voting**: If the individual classifiers have unequal performance, intuitively, it is reasonable to give more power to the stronger classifiers in voting, realized by weighted voting.
4. **Soft Voting**: For individual classifiers which produce crisp class labels, majority voting, plurality voting, and weighted voting can be used, while for individual classifiers which produce class probability outputs, soft voting is generally the choice by simply averaging all the individual outputs or the weighted output.

Note the details on the ensemble system for classification are simplified here because this thesis mainly focuses on research into regression and prediction tasks.

Remark: When the base learners are trained, combining all the base learner directly as the ensemble system is not the optimal approach as there are others methods to improve the performance and diversity of the ensemble system, such as base learner selection, ensemble pruning, ensemble clustering and even GA approaches. Thus, we use word **aggregation** to represent all the post-processing methods after the training of learners, which consists of all the combination methods.

2.1.1 Bagging

The basic motivation of parallel ensemble methods is to exploit the independence between the base learners, since errors can be reduced dramatically by combining independent base learners, such as the bagging method (also refereed as bootstrap aggregating), which is one of the earliest, most intuitive and perhaps the most straightforward ensemble-based algorithms, with surprisingly good performance [9]. The diversity of learners in bagging is obtained by using bootstrapped replicas of the training data [10]. That is, different training data subsets are randomly drawn with replacement from the entire training dataset. Figure 2.3 shows that each training data subset is used to train a different learner of the same type. Individual learners are then combined by taking a simple majority vote of their decisions for classification or taking an average in regression as the ensemble output. Since the training datasets may overlap substantially, additional measures can be used to increase diversity, such as using a subset of the training data to train each classifier or using relatively weak learners (such as decision stumps in random forest [10]).

As Breiman [9] indicated, given m training examples, the probability that the i-th training example is selected $0, 1, 2, \cdots$ times is approximately Poisson distributed with $\lambda = 1$, and thus, the probability that the i-th example will occur at least once is $1 - (1/e) \approx 0.632$. In other words, for each base learner in bagging, there are about 36.8% original training examples which have not been used in its training process. The

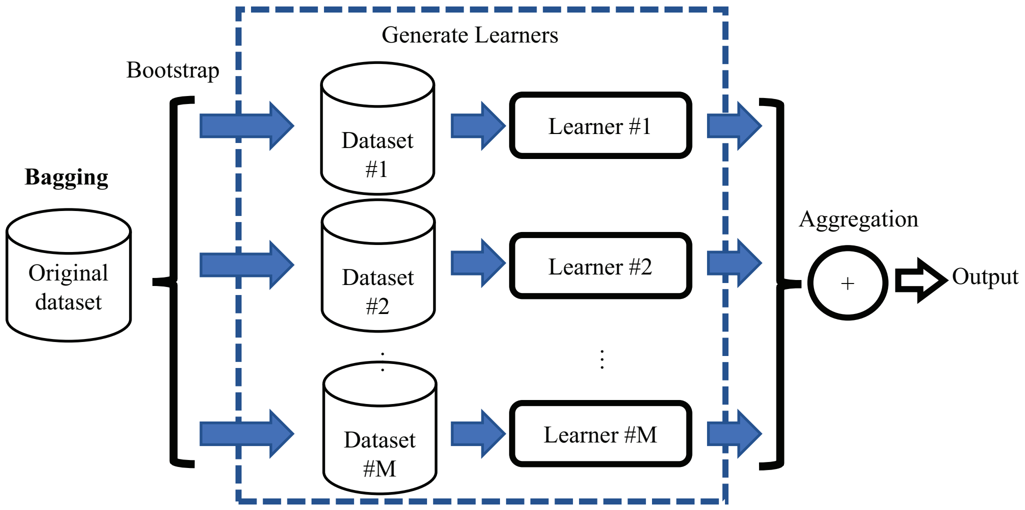

Figure 2.3: Diagram of bagging method.

goodness of the base learner can be estimated by using these out-of-bag examples, and after this, the generalization error of the bagged ensemble can be estimated [9, 109]. The simple version of the bagging algorithm is rewritten in Algorithm 2.1 as follows:

Algorithm 1 Bagging

Input: Dataset $D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(N)}, y^{(N)})\}$; Base learning algorithm L; Number of base learner M. Aggregation method A;

Output: $f_{ens}^{(M)}(x)$.

```
1: for $m = 1, 2, \cdots M$: do
2:     $f_m = L(D, D_{bs})$; $D_{bs}$ is the bootstrap distribution;
3: end for
4: return $f_{ens}^{(M)}(x) = A\{f_1, f_2, \cdots, f_M\}$.
```

2.1.2 Boosting

The basic motivation of sequential methods is to exploit the dependence between the base learners, since the overall performance can be boosted in a residual-decreasing way, such as the boosting method shown in Figure 2.4. Similar to bagging, boosting also creates an ensemble of learners by reweighting and resampling the data, which are then combined as the output [27]. However, in boosting, resampling is strategically geared to provide the most informative training data for each consecutive learner. Moreover, the term boosting also refers to a family of algorithms that can convert weak learners to strong learners. The best known of all the ensemble-based algorithms, AdaBoost (Adaptive Boosting) extends boosting to multi-class

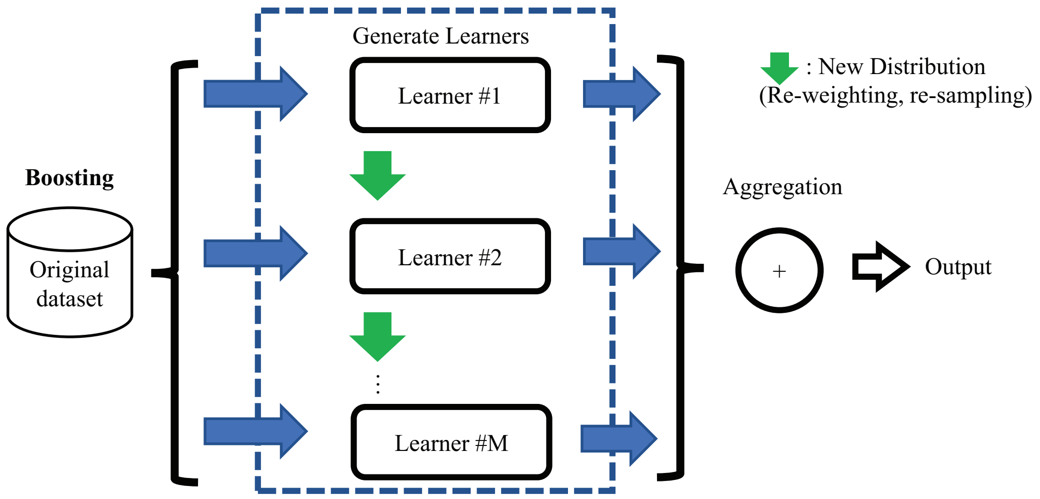

Figure 2.4: Diagram of boosting method.

and regression problems [26]. Furthermore, AdaBoost has many variations, such as AdaBoost.M1 for classification problems where each classifier can attain a weighted error of no more than AdaBoost.M2 for those weak classifiers that cannot achieve this error maximum (particularly for problems with a large number of classes, where achieving an error of less than becomes increasingly difficult), AdaBoost.R (for regression problems), among many others [22, 67, 119]. Algorithm 2.2 presents a general boosting procedure.

Algorithm 2 Boosting

```
Input: Dataset $D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(N)}, y^{(N)})\}$; Base learning
    algorithm $L$; Number of base learner $M$. Aggregation method $A$;
Output: $f_{ens}^{(M)}(x)$.
1: $D_1 = D$; Initialize distribution;
2: for $m = 1, 2, \cdots M$: do
3:     $f_m = L(D_m)$; Train a weak learner from distribution $D_m$;
4:     $e_m = P_{x,y\in\ D_m}(f_m(x) \neq y)$; Evaluate the error of $f_m$;
5:     $D_{m+1}$ =UpdateDistribution($D_m,\ e_m$);
6: end for
7: return $f_{ens}^{(M)}(x) = A\{f_1, f_2, \cdots, f_M\}$.
```

2.1.3 Negative Correlation Learning

Negative correlation learning (NCL) [79] encourages a different individual learner in an ensemble to learn different parts or aspects of training data so that the ensemble can learn the whole training data better. The goal of each learner training is to generate the best result for the whole ensemble. Unlike traditional bagging and boosting approaches, NCL was introduced to train base models simultaneously in a cooperative manner that decorrelates individual errors. Figure 2.5 shows a diagram of the NCL-based ensemble system. NCL method is based on the solid theoretical analysis of the generalization error between a single learner and an ensemble system for modelling tasks in machine learning.

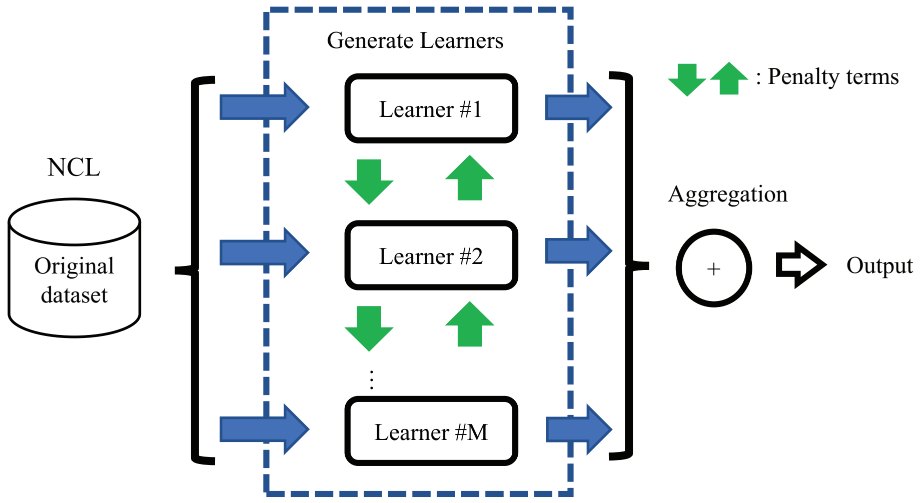

Figure 2.5: Negative correlation method in ensemble learning.

Generalization Error of A Single Learner: For data regression, the most common prediction task is to estimate a function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ from a real valued feature vector $x \in \mathbb{R}^d$ to a target domain $y \in \mathbb{R}$ on a training data set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(N)}, y^{(N)})\}$ of N sample pairs. The sample pair $(x^{(n)}, y^{(n)})$ is an independent and identically distributed (i.i.d.) sample from an unknown joint probability distribution $p(x, y)$. Note that the training set D is a realization of a random sequence that shares the same distribution $p(x, y)$.

To generate one learner for this task a model $f(x; \theta)$ is constructed, such as neural networks particularly for non-linear regressions. θ is a parameter vector and in case of neural networks, it corresponds to the weights and biases of the nodes. We assume that there is a functional relationship between training pairs$x^{(n)}$ and $y^{(n)}$: $y^{(n)} = g\left(x^{(n)}\right) + \varepsilon$, where ε is the additive noise with zero mean ($E\{\varepsilon\} = 0$) and finite variance ($Var\{\varepsilon\} = \sigma^2 < \infty$).

For convenience, let $z^N = \{z^{(1)}, z^{(2)}, \cdots, z^{(N)}\}$, where $z^{(n)} = (x^{(n)}, y^{(n)})$, denotes the training set. z^N is a realization of a random sequence $Z^N = \{Z^{(1)}, Z^{(2)}, \cdots, Z^{(N)}\}$, whose nth component consists of a random vector $Z^{(n)} = (X^{(n)}, Y^{(n)})$. In other words, $z^{(n)}$ is an independent and identically distributed (i.i.d.) sample from an unknown joint probability distribution $p(x, y)$. Then, the training process applies an optimization algorithm to estimate θ by minimizing the mean square error with the given z^N:

$$\theta^*\left(z^N\right) = argmin \frac{1}{N}\sum_{n=1}^{N}\left(y^{(n)} - f\left(x^{(n)};\theta\right)\right)^2 . \tag{2.1}$$

Since $\theta^*(z^N)$ depends on the given training data z^N, we write $\theta^*(z^N)$ to clarify the dependency of z^N. Therefore, the output value of f with an input x would be written as $f(x; \theta^*(z^N))$, for simplicity, denoted by $f(x; z^N)$. Note here that $f(x; z^N)$ is also a realization of random variable $f(x; Z^N)$.

Define $Z_0 = (X_0,\ Y_0) \in \mathbb{R}^{d+1}$ as a new random sample pair, taken from the sample distribution $Z^{(n)}$, but is independent of $Z^{(n)}$ for all n. Then the generalization error (G_E) of learner f can be defined as the following mean squared error (MSE) averaged over all possible realizations of Z^N and Z_0:

$$G_E\left(f\right) = E_{Z^N}\left\{E_{Z_0}\left\{\left[Y_0 - f\left(X_0; Z^N\right)\right]^2\right\}\right\}. \tag{2.2}$$

Here $E_{Z^N}\left\{\cdot\right\}$ and $E_{Z_0}\left\{\cdot\right\}$ represent the expectation with respect to the distribution of Z^N and Z_0, respectively. From the above definition, it is clear that $G_E\left(f\right)$ depends on neither training set z^N nor an unknown sample z_0, and only depends on the sample size N and the model of learner f. With simple algebra, Eq. (2.2) can also be expressed by the following familiar "bias/variance" decomposition [30].

$$G_E\left(f\right) = E_{X_0}\left\{\ Var\left\{f|X_0\right\} + Bias\left\{f\left|X_0\right)\right\}^2\ \right\} + \sigma^2, \tag{2.3}$$

where $Var\left\{f|X_0 = x_0\right\}$ and $Bias f|X_0 = x_0$ are conditional variance and conditional bias given $X_0 = x_0$:

$$Var\left\{f|X_0\right\} = E_{Z^N}\left\{\left(f\left(X_0; Z^N\right) - E_{Z^N}\left\{f\left(X_0; Z^N\right)\right\}\right)^2\right\}, \tag{2.4}$$

$$Bias\left\{f|X_0\right\} = E_{Z^N}\left\{f(X_0; Z^N)\right\} - g(X_0). \tag{2.5}$$

Note that $Var\left\{f|X_0\right\}$ and $Bias\left\{f|X_0\right\}$ are random variable dependent on random variables X_0. They can be defined using a conditional expectation like $E_{Z^N|X_0}\left\{f\left(X_0, Z^N\right)|X_0\right\}$. However, since Z^N is independent of X_0, it becomes $E_{Z^N}\left\{f(X_0; Z^N)\right\}$.

Generalization Error of Ensemble System: Let $\{f_1,\ f_2,\ \cdots, f_M\}$ denote M learners of the ensemble system, where the m-th learner is separately trained on $z_{(m)}^N, m = 1, 2, \cdots, M$. For simplicity, we assume that the sample size of each training set is uniformly N. Note that $z_{(m)}^N$ is a realization of a random sequence $Z_{(m)}^N$ with the same distribution $p(x, y)$. However, they cannot always be assumed to be mutually independent. The independent case will be discussed later as a special case.

The output of the ensemble for some input x is defined as the simple average of outputs of M learners for x after they have been separately trained. Specifically,

$$f_{ens}^{(M)}(x) = \frac{1}{M}\sum_{m=1}^{M} f_m\left(x; z_m^{(N)}\right). \tag{2.6}$$

Then, we have the following theorem about the generalization error of the ensemble system.

Theorem 2.1 (Generalization error in general case): Let $G_E\left(f_{ens}^{(M)}\right)$ denote the generalization error of the ensemble system given by Eq. (2.6). We have

$$G_E\left(f_{ens}^{(M)}\right) = E_{X_0}\left\{ \frac{1}{M}\overline{Var}\left\{f|X_0\right\} + \left(1-\frac{1}{M}\right)\overline{Cov}\left(X_o\right) + \overline{Bias}\left\{f\,|X_0)\right\}^2 \right\} + \sigma^2, \tag{2.7}$$

where $\overline{Var}\left(X_0\right)$, $\overline{Cov}\left(X_0\right)$, and $\overline{Bias}\left(X_0\right)$ are average conditional variance, conditional covariance and conditional bias, averaged over M learners, respectively, That is

$$\overline{Var}\left(X_0\right) = \frac{1}{M}\sum_{m=1}^{M} Var\left\{f_m|X_0\right\}, \tag{2.8}$$

$$\overline{Cov}\left(X_0\right) = \frac{1}{M(M-1)}\sum_{m}\sum_{m'\neq m} Cov\left\{f_m, f_{m'}|X_0\right\}, \tag{2.9}$$

$$\overline{Bias}\left(X_0\right) = \frac{1}{M}\sum_{m=1}^{M} Bias\left\{f_m|X_0\right\}. \tag{2.10}$$

Here, f_m denotes $f_m(X_0; Z_{(m)}^{M})$. Using the following average generalization error averaged over M learners,

$$\overline{G_E} = \frac{1}{M}\sum_{m=1}^{M}\left(E_{X_0}\left\{\ Var\left\{f|X_0\right\} + Bias\left\{f\,|X_0)\right\}^2\ \right\} + \sigma^2\right). \tag{2.11}$$

We can also present the relationship between $G_E\left(f_{ens}^{(M)}\right)$ and $\overline{G_E}$ as follows:

$$G_E\left(f_{ens}^{(M)}\right) = \frac{1}{M}\bar{G}_E + \left(1-\frac{1}{M}\right)\sigma^2$$
$$+ E_{X_0}\left\{ \left(1-\frac{1}{M}\right)\overline{Cov}\left(X_0\right) + \frac{1}{M^2}\sum_{m}\sum_{m'\neq m} Bias\left\{f_m|X_0\right\} Bias\left\{f_{m'}|X_0\right\} \right\}. \tag{2.12}$$

Corollary 1 (Same Learner Case): If the models of the learners are the same, such as neural networks with the same network structure, then we have

$$G_E\left(f_{ens}^{(M)}\right) = E_{X_0}\left\{ \frac{1}{M}Var\left\{f|X_0\right\} + \left(1-\frac{1}{M}\right)\overline{Cov}\left(X_o\right) + Bias\left\{f\,|X_0)\right\}^2 \right\} + \sigma^2. \tag{2.13}$$

An equivalent expression is

$$G_E\left(f_{ens}^{(M)}\right) = \frac{1}{M} G_E(f) + \left(1 - \frac{1}{M}\right) E_{X_0}\left\{Bias\left\{f\,|X_0\right)\right\}^2 + \sigma^2 \right\}. \tag{2.14}$$

Proof: Since f_m and $f_{m'}$ are mutually independent, $Cov\left\{f_m, f_{m'}|X_0\right\} = 0$. Hence, $\overline{Cov} \equiv 0$ in Eqs. (2.13, 2.14), respectively. These results can easily be extended to general ensemble systems defined by

$$f_{ens}^{(M)}(x) = \sum_{m=1}^{M} a_m\, f_m\left(x; z_m^{(N)}\right), \ \ where\ \ a_m > 0,\ \sum_{m=1}^{M} a_m = 1. \tag{2.15}$$

In this case, we have

$$G_E\left(f_{ens}^{(M)}\right) = E_{X_0}\left\{\sum_{m=1}^{M}\sum_{l=1}^{M} a_m^* a_l^* R_{ml}\right\},\ where\ a_i^* = \sum_j R_{ij}^{-1} / \sum_k \sum_j R_{kj}^{-1}. \tag{2.16}$$

a_i^* denotes the optimal weight that minimizes the generalization error of the general ensemble system in Eq. (2.16). R_{ij}^{-1} indicates the ij-th the component of the inverse matrix R. The ij-th the component matrix R is given by

$$R_{ij} = \begin{cases} Var\left\{f_i|X_0\right\} + Bias\{f_i|X_0\}^2, & if\ i = j, \\ Cov\left(f_i,\ f_j|X_0\right) + Bias\left\{f_i|X_0\right\} Bias\left\{f_j|X_0\right\}, & otherwise. \end{cases} \tag{2.17}$$

Note if M learners are from the same model type, then a_i^* reduces to $1/M$.

Remarks: At this point, the following important observations can be made:

- If f_m and $f_{m'}$ are negatively correlated, then the correlation contributes to a decrease in the generalization error. Conversely, if f_m and $f_{m'}$, are positively correlated, then the correlation increases the generalization error.

- Under the uncorrelated assumption, $Var\{f_{ens}^{(M)}|X_0\}$ reduces to $\overline{Var}\left(X_0\right)/M$ due to the averaging, and this reduction decreases the generalization errors. On the other hand, since $Bias\{f_{ens}^{(M)}|X_0\}$ is a simple average of M biases $\left(Bias\left\{f_m|X_0\right\}, m = 1, 2, \cdots, M\right)$, the bias term basically does not contribute to a decrease in the generalization error. Therefore, combining learners that have a higher ratio of variance to bias is more useful for decreasing generalization error than combining estimators that have a lower ratio of variance to bias.

- Even if we increase the number of ensembles M, the generalization error cannot be less than $E_{X_0}\{Bias\left\{f\,|X_0\right)\}^2\} + \sigma^2$. That is, the amount gives the lower bound of $G_E(f_{ens}^{(M)})$.

Negative Correlation Learning: NCL is a typical training scheme to build ensemble systems. The fundamental idea behind this learning algorithm lies in reducing the covariance ($\overline{Cov}\left(X_o\right)$ in Eq. (2.7)) among the base learners while ensuring the variance and bias terms of the ensemble are not increased. NCL amends the cost function with a penalty term that weakens the relationship with other individuals and controls the trade-off among the bias, variance and covariance in ensemble learning. Mathematically, the cost function of every leaner is modified with an extra decorrelation penalty term p_m as follows:

$$\theta_m^*\left(z_{(m)}^N\right) = argmin\frac{1}{N}\sum_{n=1}^{N}\left(y^{(n)} - f\left(x^{(n)};\theta\right)\right)^2 + \lambda p_m\left(x^{(n)}\right), \tag{2.18}$$

where λ is a regularization factor. The penalty term p_m can be designed in different ways depending on whether the learners of the ensemble are trained sequentially (boosting) or parallelly (bagging).

$$p_m\left(x^{(n)}\right) = \left(f_m\left(x^{(n)}\right) - y^{(n)}\right)\sum_{m'=1}^{m-1}\left(f_{m'}\left(x^{(n)}\right) - y^{(n)}\right), \tag{2.19}$$

$$p_m\left(x^{(n)}\right) = \left(f_m\left(x^{(n)}\right) - f_{ens}^{(M)}\left(x^{(n)}\right)\right)\sum_{m'\neq m}\left(f_{m'}\left(x^{(n)}\right) - f_{ens}^{(M)}\left(x^{(n)}\right)\right). \tag{2.20}$$

For instance, in sequential mode, it could decorrelate the current learner with all previous learners as in Eq. (2.19); while in parallel mode, it reduces the correlation mutually among all learners as in Eq. (2.20) using the actual ensemble output $f_{ens}^{(M)}(x^{(n)})$ instead of the target function $y^{(n)}$. Thus, we could simply change the cost function of the base learning algorithm L in Algorithms 1 and 2 to obtain the NCL version of bagging and boosting algorithms.

In [12], Brown thoroughly analysed the characteristics of the NCL technique and showed that it works for multilayer perceptron (MLP) and radial basis function (RBF) networks. In [17], a regularized version of the negative correlation learning techniques was proposed aiming at reducing the overfitting risk for noisy data.

2.2 Feature Extraction for Ensemble Learning

In additions to the learner model and algorithm, the most critical issue in building a learning system is feature extraction, which is an essential part of processing the raw data before building learners or ensembles.

In general, feature extraction belongs to feature engineering which refers to the process of using the domain knowledge of the data to create features that make machine learning algorithms work. The features in the data are important to the predictive models. The quality and quantity of the features will have a great influence on whether the model is good or not. The main reasons why feature extraction techniques are used are:

- to make the model easier to interpret, removing variables that are redundant and not adding any information;
- to reduce the size of the problem to enable the algorithms to work faster, making it possible to handle high-dimensional data;
- to reduce overfitting issues.

In the literature, there are several types of methods to complete the feature engineering task. First, the filter methods select the features by ranking them on how useful they are for the model and to compute the usefulness score, statistical tests and correlation results are used (e.g., Chi-square, ANOVA, Pearsons correlation) [1, 52]. Secondly, wrapper methods generate different subsets of features, and each subset is then used to build a model and train the learning algorithm. The best subset is selected by testing the algorithm. To select the features for the subsets different criteria are used (e.g. forward and backward selection) [73]. Finally, the embedded methods are a combination of the two previous methods. In this thesis, we only focus on the primary stream methods for feature engineering which are close to our proposed ensemble system discussed in later chapters.

2.2.1 General and Randomized Methods for Feature Engineering

Feature construction, extraction and selection techniques have gained wide interest from practitioners from statistics, pattern recognition and data mining to machine learning [1, 53], because they are helpful in making modelling tasks easier and to obtain better generalization results.

Feature construction and extraction can be viewed as data representation techniques. In some cases, the representation contains more features than necessary so feature selection can be employed to simplify the sample format. In other cases, the representations may be insufficient to describe the problem for some learning algorithms so feature construction can be engaged to enrich the feature. If some constructed features are not useful at all, feature selection can then remove these useless features. In short, feature engineering techniques aim at preserving the topological structure of the data and enhancing the discriminatory power.

From the various feature engineering methods, we list the following methods which are useful and can be applied to build the ensemble system.

Lasso: Least absolute shrinkage and selection operator was first formulated by Tibshirani in 1996 [95]. It is a powerful method that performs two main tasks: regularization and feature selection. The Lasso method puts a constraint on the sum of the absolute values of the model parameters; the sum has to be less than a fixed value (upper bound). To do so, the method applies a shrinking (regularization) process where it penalizes the coefficients of the regression variables shrinking some of them to zero. During the feature selection process the variables that still have a non-zero coefficient after the shrinking process are selected to be part of the model. Given the input data X, an $N \times p$ matrix, each row represents a sample with p features, and the target matrix or vector is Y (or y). The Lasso estimate is defined as follows:

$$\beta^{Lasso} = arg\min_{\beta} \frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} - \beta_0 - \sum_{j=1}^{p} x^{(i,j)} \beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j \,|. \tag{2.21}$$

The goal of this process is to minimize the prediction error. In practice, the tuning parameter λ controls the strength of the penalty. When λ is sufficiently large then coefficients are forced to be exactly equal to zero, which reduce dimensionality. The larger the parameter λ, the more coefficients are reduced to zero. On the other hand, if $\lambda = 0$, we have an OLS (Ordinary Least Square) regression.

Moreover, Lasso helps to increase the model interpretability by eliminating irrelevant variables that are not associated with the response variable, which also reduces overfitting [42, 95, 96, 106]. There are extended versions of Lasso such as Elastic Nets [120] and Group Lasso [28, 62]. In [8], Boyd proposed a distributed optimization method, alternating direction method of multipliers (ADMM), to solve the Lasso problem. This Lasso method is used in our proposed ensemble system as discussed in later chapters.

mRMR: Minimal-redundancy-maximal-relevance (mRMR) is a feature selection method according to the maximal statistical dependency criterion based on mutual information, which selects a compact set of superior features at low cost with more sophisticated feature selectors (e.g., wrappers) [73]. A theoretical analysis of the mRMR condition shows that it is equivalent to the maximal dependency condition for first-order feature selection.

Furthermore, the mRMR incremental selection scheme contributes to avoiding the problematic multivariate density estimation in maximizing dependency. Assume the task is to find a feature set S with m features $\{x_i\}$, which jointly have the largest dependency on the target class c, the mRMR is to optimize D (Max-Relevance / Max-Dependency) and R (Min-Redundancy) simultaneously:

$$\max \Phi\left(D, R\right), \ \Phi = D - R, \tag{2.22}$$

where $D = 1/\left|S\right| \sum_{x_i \in S} I(x_i; c)$ and $R = 1/\left|S\right|^2 \sum_{x_i, x_j \in S} I\left(x_i, x_j\right)$. By combining mRMR and other more sophisticated feature selectors, experiment results confirmed that mRMR leads to promising improvement on feature selection and classification accuracy.

MIC: Maximal information coefficient is a measure of dependence for two-variable relationships [77]. It captures a wide range of associations both functional and not, and for functional relationships provides a score that roughly equals the coefficient of determination (R^2) of the data relative to the regression function. Intuitively, MIC is based on the idea that if a relationship exists between two variables, then a grid can be drawn on the scatterplot of the two variables that partitions the data to encapsulate that relationship. MIC belongs to a broader class of **maximal information-based nonparametric exploration (MINE)** statistics for identifying and classifying relationships. The MIC uses binning as a means to apply mutual information on continuous random variables. Binning has been used as a way of applying mutual information to continuous distributions; what MIC contributes in addition is a methodology for selecting the number of bins and picking a maximum over many possible grids. The rationale is that the bins for both variables should be chosen in such a way that the mutual information between the variables is maximal $H(X_b) = H(Y_b) = H(X_b, Y_b)$. Thus, the following two properties hold. First, the bins would have roughly the same size, because the entropies $H(X_b)$ and $H(Y_b)$ are maximized by equal-sized binning. And second, each bin of X will roughly correspond to a bin in Y.

RDC: Randomized dependence coefficient is a measure of nonlinear dependence between random variables of an arbitrary dimension based on the Hirschfeld-Gebelein-Renyi Maximum Correlation Coefficient [57]. RDC is defined in terms of correlation of random non-linear copula projections; it is invariant with respect to marginal distribution transformations, has low computational cost and is easy to implement. Given the random samples $X \in \mathbb{R}^{p\times n}$ and $Y \in \mathbb{R}^{q\times n}$ the parameters $k \in \mathbb{N}^+$and $s \in \mathbb{R}^+$, the RDC between X and Y is defined as: $rdc(X, Y; k, s) := sup_{\alpha,\beta}\rho(\alpha^T\Phi^{k,s}_{P(X)}, \beta^T\Phi^{k,s}_{P(Y)})$, which is the largest canonical correlation between $\alpha^T\Phi^{k,s}_{P(X)}$ and $\beta^T\Phi^{k,s}_{P(Y)}$, where $\Phi^{k,s}_X \in R^{2k\times n}$, $\Phi^{k,s}_{X_{i,j}} = \phi(w_j^T x_i + b_j)$, $w_j \sim N(0, sI)$ and $b_j \sim U(-\pi, \pi)$.

RIC: Randomized information coefficient is a mutual information based measure with low variance, to quantify the dependency between two sets of numerical variables [78]. It is a randomized method to ranking the relationships between features and targets. It formally establishes the importance of achieving low variance when comparing relationships using the mutual information estimated with grids. Additionally, it experimentally demonstrates the effectiveness of RIC for (i) detecting noisy dependencies and (ii) ranking dependencies for the applications of genetic network inference and feature selection for regression. Similar to RDC, the RIC bewteen X and Y is defined as the expected Normalized Mutual Information (NMI) across all possible discretization grids that encapsulate the joint probability distribution for (X, Y): $ric(X, Y) \triangleq \int_G NMI(X, Y|G)P(G)dG$, where a grid G for the sets of variable X and Y is the Cartesian product of two partitions G_X and G_Y: i.e., $G = G_X \times G_Y$.

The theoretical analysis of RIC justifies the benefits of having a low-variance estimator of mutual information based on grids for the task of ranking relationships, where systematic biases cancel each other out. By reducing the estimation variance of mutual information with grids, RIC is extremely competitive for ranking different relationships. Based on the experiment results, RIC is very competitive over other 16 state-of-the-art measures. Other prominent features of RIC include its simplicity and efficiency, making it a promising new method for dependency assessment.

2.2.2 Neural Networks for Representation Learning

Neural networks, such as autoencoders (AE), restricted Boltzmann machines (RMBs) and convolutional neural networks (ConvNets), can be used for feature extraction instead of making perdition or classification. A basic autoencoder (AE) is a type of unsupervised three-layer neural network that is trained to reconstruct its input to its output. A restricted Boltzmann machine (RBM) is a generative stochastic artificial neural network that can learn a probability distribution over its set of inputs. Convolutional neural networks are a specialized kind of neural network for processing data that has a known, grid-like topology, which is the corner-stone model in the modern deep learning community. In short, the hidden layers outputs of these kinds of neural networks can represent a new format of the raw input and using these new representations could build better models than using the raw input directly. A general model usually consists of two parts, an encoder and predictor as shown in Figure 2.5A.

Autoencoder: A basic autoencoder (AE) is a type of unsupervised three-layer neural network that is trained to reconstruct its input to its output. Internally, it has a hidden layer h that describes a code used to represent the input. It can be viewed as comprising two parts, an encoder network $h = f(x)$, and a decoder network $r = g(h)$. The encoder usually transforms the input data from a high dimensional space to code

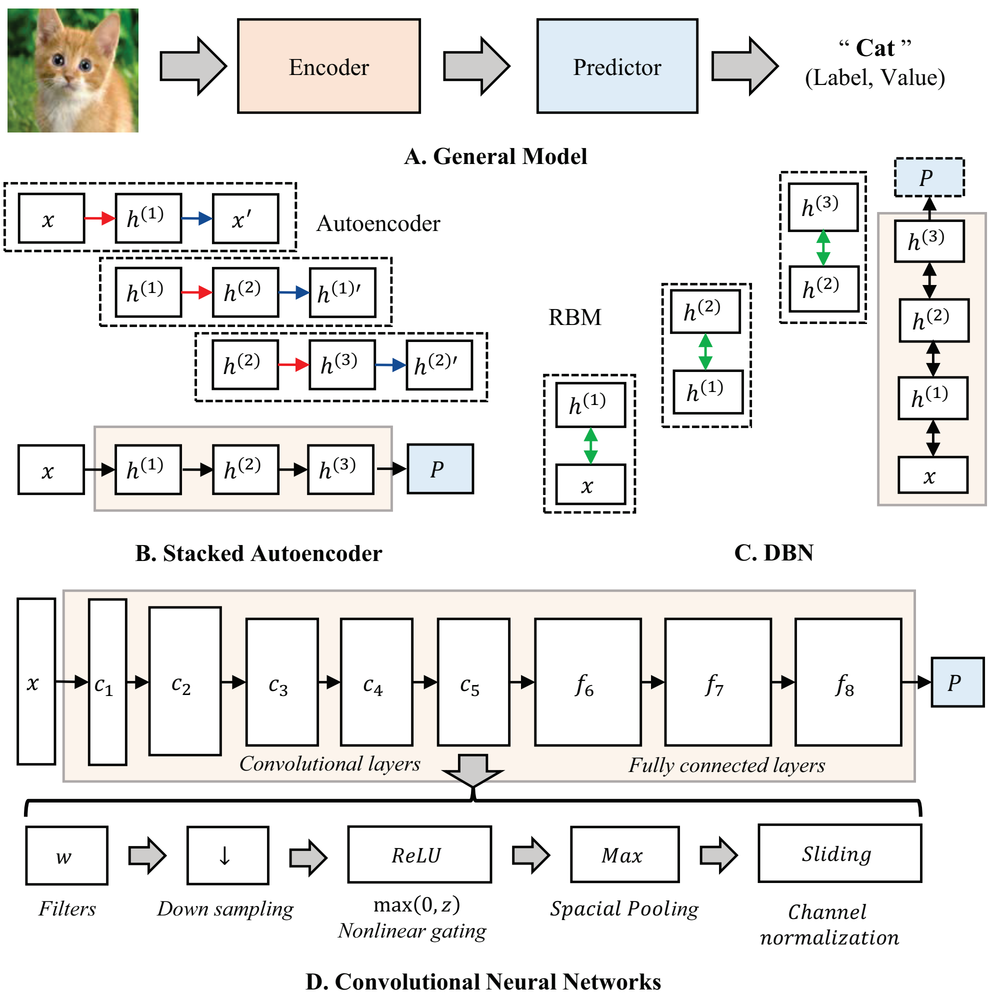

Figure 2.6: Neural networks in deep learning.

in a low dimensional space, and the decoder remodels the code to its initial input. We expect the encoding and decoding processes will result in h containing useful properties, which lead to variant autoencoders, such as, regularized autoencoders and sparse autoencoders [46, 87]. The learning processes of all these autoencoders can be described simply as minimizing the loss function with a regularization term:

$$J_{AE} = \sum_{x \in D} L\left(x, g\left(f\left(x\right)\right)\right) + \Omega(h, \theta, x). \tag{2.23}$$

For regularized autoencoders, the regularization term becomes $\Omega(\theta) = \lambda(\| W_1 \|_F^2 + \| W_2 \|_F^2)$; for sparse autoencoders, $\Omega(h) = \lambda\ KL(\rho \parallel \hat{\rho})$, where KL denotes Kullback-Leibler divergence, ρ is a sparsity parameter (close to zero), and $\hat{\rho}$ is the average activation vector of the hidden layer. Rather than adding a penalty Ω, denoising autoencoders minimize $J_{AE} = L\left(x, g\left(f\left(\widetilde{x}\right)\right)\right)$, where $\widetilde{x}$ is a copy of x that has been corrupted by some form of noise [100]. Another strategy for regularizing autoencoder called contractive autoencoder [99] is to use a penalty as $\Omega(h,\, x) = \lambda(\parallel \bigtriangledown_x h \parallel_F^2)$ by penalizing the derivatives of the hidden layer output. It has theoretical connections to denoising autoencoders, manifold learning and probabilistic modelling. In particular, AE can be stacked together into a multilayer structure, such as the stacked autoencoders shown in Figure 2.6B.

RBM: A restricted Boltzmann machine (RBM) is a generative stochastic artificial neural network that can learn a probability distribution over its set of inputs by the contrastive divergence (CD-k) training algorithm. RBMs have found applications in dimensionality reduction [36], classification [45], collaborative filtering [81], feature learning [20] and topic modelling [35]. They can be trained in either supervised or unsupervised ways, depending on the task. RBM can also be used in deep learning networks. In particular, deep belief networks (DBN Figure 2.6C) can be formed by "stacking" RBMs and optionally fine-tuning the resulting deep network with gradient descent and back-propagation methods [47] [71].

ConvNets: convolutional neural networks (ConvNets) are a specialized kind of neural network for processing data that have a known, grid-like topology. They are the corner-stone model in modern deep learning and have been extremely successful in practical application since the 90s [31, 47]. Figure 2.6D demonstrate an example of the ConvNets with details of the components and operations in each layer. The name "convolutional neural network" indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. Moreover, they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks developed for learning regular neural networks still apply.

The main difference is that the ConvNets architectures make the explicit assumption that the inputs are 2D/3D-images, which allows us to encode specific properties, making the forward function more efficient to implement and vastly reducing the number of parameters in the network. ConvNets take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. It automatically reduces the images into a limited-length vector for the learning task; those vectors are the representation of the raw inputs in the feature space generated from the ConvNets, which make the task of modelling the last layer easier.

In sum, as we know feature construction, extraction and selection are essential, another research topic is about data causality, referred to as "**causal inference in machine learning**", which can make better generalizations from sparse data when learning about word meanings, unobserved properties, causal relationships, and many other aspects of the world applications [72].

2.3 Randomized Algorithms for Neural Networks

Neural networks built by stochastic gradient decrease (SGD) algorithms are popular and successful in applications, but they suffer from local minima, over-fitting, the sensitivity of learning parameter options and a slow learning rate. Randomized learning techniques have been extensively explored in relation to neural network modelling since the 80s and 90s, in terms of different perspectives and/or in various contexts, including the expanded presentation for perceptron [94], radial basis function (RBF) networks [58], single hidden layer feed-forward neural networks (SLFNs) [14], random vector functional-link (RVFL) networks [68, 70, 116] kernel approximation [75] and reservoir computing [60] etc. Based on the function approximation theory, both the randomized RBF networks and RVFL networks belong to random basis approximators (RBAs) [92]. The stochastic configuration networks (SCN) [105] add a supervision mechanism in the training process, putting on the random basis, thus a significantly boosting the performance of randomized neural networks.

In the following sections, we focus on the basic neural networks and the most-used randomized neural networks, RVFL and SCN, with discussions about the parameter estimation and algorithm implementations.

2.3.1 Neural Networks

Neural networks have received extensive attention due to their high potential for various applications, such as financial forecasting, pattern recognition, computer version, natural language processing and machine translation [46, 87]. A simple feedforward neural networks model with only one hidden layer contains L hidden neurons can be descripted as:

$$f\left(x\right)=\sum_{l=1}^{L}\beta_l\phi_l\left(w_l^Tx+b_l\right)\ , \tag{2.24}$$

where w_l and b_l is the input weight vector and bias of the l-th hidden neuron, and ϕ_l is the activation function and the β_l is the output weight vector.

The learning task finds the proper weights and biases of all the networks that minimize the cost function on the training data set D. Various training algorithms have been developed based on the well-known gradient descent optimization algorithms for training the networks. Readers are suggested to refer to [80] for details, where Ruder reviewed the different variants of gradient descent, summarized the challenges, and introduce the most common optimization algorithm, for example, batch gradient decrease (BGD), stochastic gradient decrease (SGD), mini-batch gradient descent, adaptive gradient algorithm (Adagrad and Adadelta), root mean square propagation (RMSProp) and adaptive moment estimation (Adam, AdamMax, Nadam) [31, 31, 47, 87, 100]. It can be concluded that the mainstream methods for training neural networks are dominated by the gradient-descent optimization methods.

From the single layer feedforward neural networks (SLFNs), to MLP and then the hot Deep Networks, more and more applications have been developed, and some reveal remarks results. As we mentioned above, the AE, RBM and ConvNets are neural networks but with more complex structure and objectives. They could be used as models or feature extractors in machine learning.

Since 2012, deep neural networks (DeepNets) have achieved the competing ability to beat human on several games, for example, the AlphaGo from Google-Brain has won the world champion in the ancient Chinese chess game ”GO”, which is considered the most challenging game in the world and requires very high intelligence for being a master [18, 66, 90, 91]. Moreover, the tech-giants, such as Google, Baidu, Tesla and Apple, have been spending enormous resources in developing AI applications, like the autopilot vehicles, which will change everyone life in the future.

The success of applying neural networks is inspiring. However there are still some difficulties in building neural network models. It becomes a common sense by many researchers that it is hard to settle the structure of neural networks including the layers, hidden node number and the connection between them. The gradient based algorithms suffer from the local minimum issue and the gradient vanishing problems. Another obstacle is that the training cost including computing time and the hardware is massive and it is increasing. Even though there are better approaches to ease these difficulties; they have not been fully solved.

2.3.2 Random Vector Functional Link Networks

Since the late 90s, a different type of neural network, named the random vector functional-link (RVFL) network, has attracted the attentions of academics due to its difference from the SLFNs [39, 68, 69, 70]. Instead of the classic layer-by-layer design, the original architecture of RVFL networks has direct links from its input layer to the output layer, aiming to enhance the data representations of the hidden layer as shown in Figure 2.7. The weights and biases of the hidden layer nodes are randomly generated and fixed. Then, the training task of RVFL networks is transformed into a linear regression task. The output weights of the hidden nodes can be optimized by various algorithms from the least square regression analysis. Igelnik

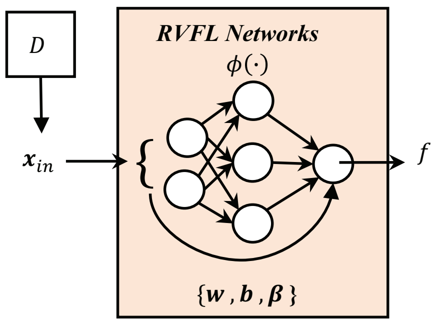

Figure 2.7: Random vector functional link networks.

et al. [39] verified the universal approximation capability of RVFL networks using the Monte-Carlo method on the limit-integral representation of the target function. Their theoretical justifications also apply to networks whose hidden neurons are from the products of univariate functions or radial basis functions. To compare the RVFL networks and the stochastic configuration networks (SCN), we briefly present the mathematical formulation of their result as follows:
Theorem 2.2 (Igelnik and Pao [39]): For any compact set $D \subset \mathbb{R}^d$, given $f \in C(D)$ (i.e., the set of all continuous functions defined over D), and any activation g that satisfies $\int_R \|g(t)^2\| dt < \infty$ or $\int_R \left|g\prime(t)^2\right| dt < \infty$, there exist a sufficiently large L, a set of $\beta_1,\ \beta_2, \cdots, \beta_L$ and a probabilistic space X_L, such that $f^{(L)} = \sum_{l=1}^{L} \beta_l g_l(x; w_l, b_l)$ can approximate f with arbitrary accuracy in the sense of probability one, if w_l and b_l are randomly assigned over X_L and follow a certain distribution. This result can be expressed as

$$\lim_{L\to\infty} E\left(f_D \left|f(x) - f^{(L)}(x)\right| dx\right) = 0, \tag{2.25}$$

where E is the expectation operator with respect to the probabilistic space X_L.
RVFL networks Algorithm: RVFL networks with a single output and L hidden nodes can be expressed by

$$f(x) = \sum_{l=1}^{L} \beta_l \phi_l\left(w_l^T x + b_l\right), \tag{2.26}$$

where $\phi_l\left(w_l^T x + b_l\right) = 1/1 + e^{-(w_l^T x + b_l)}$. The random weights w_l and biases b_l take uniformly distributed random values in the interval $[-\alpha, +\alpha]$ which are estimated empirically. Assuming there is a training data set D with N samples, the loss function for estimating the output weights β can be formulated with a regularization term μ:

$$\beta^* = argmin_\beta \sum_{i=1}^{N} \left(y_i - \sum_{l=1}^{L} \beta_l \phi_l \left(w_l^T x_i + b_l \right) \right)^2 + \mu \parallel \beta \parallel_2^2. \tag{2.27}$$

The matrix form of Eq. (2.27) can be written as

$$\beta^* = arg \min_\beta \parallel Y - \Phi\beta \parallel_2^2 + \mu \parallel \beta \parallel_2^2, \tag{2.28}$$

where Φ is the output matrix of the hidden layer of RVFL networks,

$$\Phi = \begin{pmatrix} \phi_1(x_1) & \cdots & \phi_L(x_1) \\ \vdots & \ddots & \vdots \\ \phi_1(x_N) & \cdots & \phi_L(x_N) \end{pmatrix}_{N \times L}. \tag{2.29}$$

If $\mu = 0$, the least squares method is used, $\beta^* = \Phi^\dagger Y$, where $\Phi^\dagger$ is the pseudo-inverse of Φ. When $\mu > 0$, the optimal β^* can be formulated as a ridge regression estimate:

$$\beta^* = \left(\Phi^T \Phi + \mu I \right)^{-1} \Phi^T Y, \tag{2.30}$$

where I is the identity matrix of suitable dimensionality. In the case of $N < L$, Eq. (2.30) can be simplified using the fact that, for any $\mu > 0$, there is:

$$\left(\Phi^T \Phi + \mu I \right)^{-1} \Phi^T = \Phi^T \left(\Phi \Phi^T + \mu I \right)^{-1}, \tag{2.31}$$

thus, an alternative formula is obtained for the optimal output weights β^*, where the $N \times N$ matrix $\Phi\Phi^T$ is easier to invert: $\beta^* = \Phi^T \left(\Phi\Phi^T + \mu I \right)^{-1} Y$. Furthermore, the l_1-norm regularization can be used to achieve RVFL networks with sparse output layers, which is usually called the Lasso problem [24, 95], hence we can simply change the function as follows:

$$\beta^* = arg \min_\beta \parallel Y - \Phi\beta \parallel_2^2 + \mu \|\beta\|_2^2, \tag{2.32}$$

Yang et al. [112] also provided a comprehensive review of some representative l_1-minimization methods, which can also be applied to RVFL networks modelling.

Li et al. [48] made corrections to the proof of **Theorem 2.1** on the scope setting for w and b, and then Husmeier upgraded the universal approximation property of RVFL networks with a symmetric interval setting for the random parameters in [38]. This property only holds when the target functions satisfy the Lipschitz condition. However, the above theoretical works did not cover algorithm design or the implementation issues of RVFL networks. Ivan et al. [98] investigated the importance of randomness on the models performance empirically and demonstrated that the trained RVFL networks fail in approximating the target function with a very high probability when it used an inappropriate scope setting for the hidden parameters. This phenomenon was further investigated by Gorban in [32] with mathematical justifications proving that in the absence of additional conditions, one RVFL network needs an exponentially growing number of nodes to approximate a non-linear map, and the resulting RVFL model is extremely sensitivity to parameters. This means that the excellent approximation performance of RVFL networks is not guaranteed for every random assignment of the hidden parameters.

Additionally, RVFL networks have been widely explored in different contexts, frameworks and applications including optimal control [69], time series prediction [16, 76] handwritten digits recognition [71], semi-supervised [82], distributed learning [85], ensemble learning [2], multi-source data modeling [50, 51], and deep neural networks [15, 118]. However, the practical issues and common pitfalls of RVFL network-based data modelling were not deeply analyzed until Li and Wangs work appeared in [49].

Li and Wang focus on the impact of the parameter scope of the RVFL networks and empirically show that a widely used setting (e.g. $[-\alpha, +\alpha]$) is misleading. They also prove that RVFL networks may have no learning or generalization capabilities if an inappropriate scope is chosen for randomly assigning the input weights and biases. The theoretical result in [38] verifies the universal approximation property of a RVFL-based learner with this kind of symmetric interval setting for the random parameters. However, this specific scope should be based on the training samples. Otherwise the universal approximation capability of the RVFL-based learner is not guaranteed for a given fixed scope. Moreover, there exist many cases when the probability that RVFL networks have a full-column-rank hidden output matrix approaches zero. The RVFL networks may perform poorly in both training and testing, due to an inappropriate setting of the scope. They also provide a theoretical verification for the infeasibility of a class of incremental RVFL (IRVFL) networks for universal approximation.

With the conclusion that a fixed scope may not cover the reasonable range that is appropriate for randomly assigning the hidden parameters (they named this ”supportive range” for random parameters), they propose the stochastic configuration networks (SCN) [105], a novel framework for constructive neural networks via randomized learning techniques, to solve the issues that RVFL networks could not.

2.3.3 Stochastic Configuration Networks

In 2017, Wang and Li proposed the SCN, a framework for building neural networks incrementally by randomized methods with supervisory mechanisms, and proposed three algorithms SC-I, SC-II and SC-III for implementations. This work provides the theory and applicable methods for solving real data modelling tasks with randomized neural networks. The supervisory mechanism of the SCN incrementally adds the hidden nodes whose weights and biases are constrained and selected automatically during the training process. This constructive approach guarantees the SCN is a universal approximator for a given prediction task. Figure 2.8 demonstrates the structure of SCN. For completeness, we revisit the main theoretical result in **Theorem 2.3** below.

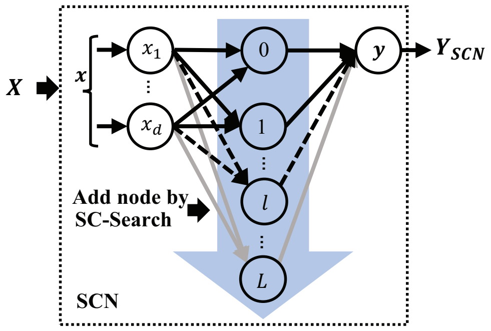

Figure 2.8: Stochastic configuration networks.

Given a target function $f : \mathbb{R}^d \to \mathbb{R}^m$. Suppose that an SCN model has already been built with $L-1$ hidden nodes, i.e., $f_{L-1} = \sum_{l=1}^{L-2} \beta_l \phi_l$, where $\beta_l = \left[\beta_{l,1}, \beta_{l,2}, \cdots, \beta_{l,m}\right]^T$, and $\phi_l(w_l^T x + b_l)$ is an activation function of the lth hidden node with random input weights w_l and $biasb_l$. Denoting the residual error by $e_{L-1}^* = f - f_{L-1} = \left[e_{L-1,1}^*, e_{L-1,2}^*, \cdots, e_{L-1,m}^*\right]$, where $\left[\beta_1^*, \beta_2^*, \cdots, \beta_{L-1}^*\right] = argmin_\beta \parallel f - \sum_{l=1}^{L-1} \beta_l \phi_l \parallel$.

Theorem 2.3 (Wang and Li [105]) Suppose that span (Γ) is dense in L_2 space and for any $\phi \in \Gamma$, where $0 < |\phi| < b_\phi$ for some $b_\phi \in R^+$. Given $0 < r < 1$ and a non-negative real number sequence $\{\mu_L\}$ with $\lim_{L\to+\infty} \mu_L = 0$ subjected to $\mu_L \leq 1 - r$. For $L = 1, 2, \cdots$, denoted by

$$\delta_L^* = \sum_{q=1}^{m} \delta_{L,q}^*, \delta_{L,q}^* = (1 - r - \mu_L)\, eL - 1, q * 2, q = 1, 2, \cdots, m. \tag{2.33}$$

If the random basis function ϕ_L is generated to satisfy the following inequalities:

$$\langle e_{L-1,q}^*, \phi_L \rangle^2 \geq b_\phi^2 \sigma_{L,q}^*, \ q = 1, 2, \cdots, m, \tag{2.34}$$

and the output weights are evaluated by

$$[\beta_1^*, \beta_2^*, \cdots, \beta_L^*] = arg\ min_\beta = \parallel f - \sum_{l=1}^{L} \beta_l \phi_l \parallel. \tag{2.35}$$

Then, we have $\lim_{L\to+\inf} \parallel f - f_L^* \parallel = 0$, where $f_L^* = \sum_{l=1}^{L} \beta_l^* \phi_l, \beta_l^* = \left[\beta_{l,1}^*, \beta_{l,2}^*, \cdots, \beta_{l,m}^*\right]^T$. The experiments demonstrate that the proposed SC algorithms are more effective and efficient in data modeling, compared with the existing methods such as modified quickprop (MQ) [44] and IRVFL [49].

Furthermore, they proposed the robust SCN model for real applications, where data samples can be noisy and contain some outliers, based on the techniques of kernel density estimation (KDE) and maximum correntropy criterion (MCC) [104]. The main idea to build robust stochastic configuration networks with KDE (RSC-KDE) can be formulated as follows by solving a weighted least squares (WLS) problem, that is,

$$\min_{\beta,\theta} \sum_{i=1}^{N} \theta_i \parallel y_i - \sum_{l=1} L\beta_l \phi_l(w_l, x_i, b_l) \parallel^2, \tag{2.36}$$

where the $\theta_i \geq 0$ $(i = 1, 2, \cdots, N)$ is the i-the penalty weight, representing the contribution of the corresponding sample to the cost function Eq. (2.36). The robust stochastic configuration networks with the MCC (RSC-MCC) can be built according to the following optimization problem:

$$\max_{\beta} \sum_{i=1}^{N} \Psi\left(y_i - \sum_{l=1}^{L} \beta_l \phi_l\left(w_l,\ x_i, b_l\right)\right), \tag{2.37}$$

where $\Psi\left(u\right) = exp(- \parallel u \parallel^2 / 2\sigma^2)$ is a Gaussian kernel function with a fixed scale parameter σ.

Similar to the development of RSC-KDE, one key feature in building an RSCN model with MCC is to use a weighted version of the residual error in the construction of an SCN. Then, they extend the SCN to a multilayer structure, named DeepSCN, a randomized approach for incrementally building deep neural networks [103]. This work established the fundamental result on the universal approximation property of this type of network, and it also proposed an improved supervisory mechanism to constrain the random assignment to the weights and biases, and direct the layer structure of the networks. In their experiments, compared with the original SCN, the performance of DeepSCN was significantly improved by automatically stacking layers over the previous networks.

2.4 Randomized Neural Network Ensemble

There are various methods in existing research for neural network ensemble learning [13, 34, 55, 56, 79, 110], but they use the error back-propagation (BP) algorithm to build the neural networks. Unfortunately, the BP algorithm suffers from sensitivity to the setting of the learning rate, local minima and slow convergence. Therefore, it is challenging to apply the existing ensemble methods for large-scale data sets. To overcome this problem Alhamdoosh and Wang employed random vector functional-link (RVFL) networks to develop a fast decorrelated neuro-ensemble (termed DNNE) [2].

DNNE first builds a group of RVFL networks and then employs the least squares method with a negative correlation learning scheme to analytically calculate the output weights of these base networks. A minimum norm least squares solution is derived and formulated in a matrix form for computational exercises. Their results over the testing datasets show improved performance over bagging, boosting, simple ensembles and random forests.

DNNE can perform well on smaller data sets. However, it is quite limited for when dealing with large scale data because of its high computational complexity, the scalability of numerical algorithms for the least squares solution, and hardware constraints (mainly relation to PC memory). Recall that physical data may come from different types of sensors, localized information sources or potential features extracted from multiple runs of some specific feature selection algorithms. Thus, for large-scale data analytics, it is useful and significant to develop a generalized neuro-ensemble framework with heterogeneous features.

Furthermore, the RVFL networks in [2] were built with a default scope setting of the random weights and biases. From the theoretical statements on the universal approximation property of RVFL networks in [48] and the empirical results in [49], the default scope setting (i.e., $[-1, +1]$) for the random weights and biases cannot ensure the modelling performance at all. The limitations of DNNE can be summered as follows:

- The system inputs are centralized or combined with different types of features;
- The analysed method of computing the output weights becomes infeasible for large-scale data sets, which is related to the nature of the base learner model (i.e., the number of nodes at the hidden layer must be sufficiently large to achieve sound performance).

To relax these constraints and focus on the rapid building of neuro-ensembles with heterogeneous features, we generalize the classical NCL-based ensemble framework into a more general form in Chapter 3, where a set of input features are fed into the SCN base models separately. This work also provides a feasible solution by using two iterative methods for evaluating the output weights of the SCN ensemble (SCNE) with theoretical and experimental results.

2.5 Genetic Algorithm (GA)

A genetic algorithm (GA) is a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). It is commonly used to generate high-quality solutions to optimization and search problems by relying on bio-inspired operators such as mutation, crossover and selection [54]. In a genetic algorithm, a population of candidate solutions (called individuals) to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties (its chromosomes) which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible.

The evolution usually starts from a population of randomly generated individuals, and is an iterative process, with the population in each iteration called a generation. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved. If we assume the population is an ensemble and the individuals are learners, this evolutionary feature can help to improve the performance of the learners of an ensemble system by specifying the design of fitness, for example, choosing the best neural networks from an existing ensemble with the highest classification accuracy [17, 54, 110].

The better individuals are stochastically selected from the current population, and each individual's genome is modified (crossover and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced or a satisfactory fitness level has been reached for the population. In short, a typical genetic algorithm requires:

1. a genetic representation of the solution domain;

2. a fitness function to evaluate the solution domain.

Once the genetic representation and the fitness function are defined, a GA proceeds to initialize a population of solutions and then to improve it through repetitive application of the crossover, mutation and selection operators. In addition to the leading operators above, other heuristics techniques may be employed to make the calculation faster or more robust. Various methods penalize the crossover between candidate solutions that are too similar and intervene with the mutation process to avoid getting redundant "genes". All these techniques aim to encourage population diversity and help prevent premature convergence to a less optimal solution [63, 110], which are the same purpose of building a neural network ensemble.

Motivated by the GA and randomized neural network ensemble, in Chapter 5, we propose the GA-based evolving stochastic configuration network ensemble (eSCNE) framework with different modifications and algorithms, aiming to generate an optimal SCN model from an ensemble of pretrained SCN models.

2.6 Summary

We conclude this chapter with further remarks on the randomized neural network ensemble learning framework. According to the ensemble generalization error, combining learners that have a higher ratio of variance to bias is more useful for decreasing generalization error than combining estimators that have a lower ratio of variance to bias. Feature extraction is necessary for building learners of ensemble system. For the learner model of traditional neural networks trained by SGD based methods, the universal approximation property ensures that the learner can approximate a given target function with any accuracy, however for the cost effectiveness, advanced randomized algorithms could generate feasible neural networks in a short time. However, issues caused by the potential impacts of randomness still exist in randomized neural networks including RVFLs and SCNs. Ensemble approaches provide a way to solve these problems, thus how to effectively build an ensemble system of randomized learners with a universal approximation capability, how to select the features for these learners and how to generate the optimal learner from the ensemble, should be investigated in depth.

In the following chapters, we discuss the research on the frameworks for building the stochastic configuration network ensemble (SCNE) with different aggregation strategies (bagging, boosting and NCL). These frameworks consist of feature engineering approaches for ensemble system, such as heterogeneous feature generation with Lasso, and GA-based approaches for ensemble optimization. The theoretical results, such as algorithm development, verification of the experiment results on benchmark datasets and its implementations in biology and industrial case studies are provided in the following chapters.

Chapter 6

Conclusion and Further Work

This thesis presented the complete framework of the randomized neural network ensemble with heterogeneous features for data analytics including theoretical foundations and algorithm designs. Our proposed stochastic configuration networks ensemble (SCNE) framework takes advantage of the merits of advanced randomized learning techniques for neural networks and stochastic configuration networks (SCN) and employs them as the base learners in the ensemble system with different aggregation approaches.

Facing different data modeling challenges, such as heterogeneous features and high-dimensional samples, the SCNE provides specific solutions to obtain feasible results within a limited time and with reduced computing power. In Chapter 3, we showed that the generalized SCNE system trained on large dataset achieved good performance on challenging real-world data analytics. We also proposed the bagging SCNE, boosting SCNE and NCL SCNE for building the ensemble system. In particular, we designed the iterative algorithms, block Jacobi method and block Gauss-Seidel method, to replace the pseudo-inverse method in order to lower the computation cost, shorten the computing time and to make the most use of the hardware. To solve the high-dimensional data regression problem, we introduced the Lasso-SCNE framework in Chapter 4. In the case study using the NMR dataset for protein predictions, we used the Lasso as a feature selector to generate the feature ensemble and then pruned the SCNE again using the Lasso method. The results proved the advantages of Lasso-SCNE compared with seven other models. In Chapter 5, we developed an evolving SCNE framework based on genetic algorithms for industrial data modeling. We used the novel fitness function and for the first time, used the node index matrix in the GA operations. The results revealed the correlations between the SCN models generalization ability and the algebraic properties with its hidden node output matrix. Moreover, this eSCNE provides a new method by which to obtain an optimal SCN model with node number control.

While there are many possible directions for improving the SCNE system, it is important to consider which ones will lead to the biggest gain in the quality of predictions. Due to the time and space trade-off in computation and the uncertain nature of randomized algorithms, the SCN has unstable performance and a changing structure of networks on modeling tasks. Following the classic ensemble methods, the bagging SCNE, boosting SCNE, NCL-SCNE are proposed with a more general theorem of ensemble generalization error expression for modeling tasks on heterogeneous datasets. Thus, SCNE could lower this uncertainty by aggregating SCNs into an ensemble system. Meanwhile, there are redundant hidden nodes inside the trained SCN which increase the computing cost and the ensemble size. SCNE combined the Lasso and GA techniques (Lasso-SCNE and eSCNE) to prune these nodes and further obtain an optimal SCN from the node pool of the ensemble system, making the ensemble system slim and cost-effective. All these SCN-based ensemble systems are universal approximators with significant improvements compared to a single SCN. Based on the experiment results, it can be seen that the SCNE is more suitable for industrial applications than a single SCN model with respect to system safety and robustness.

It should be mentioned that in this thesis, we mainly focus on applying the SCN as the base learner for the ensemble system. Thus, the ensemble system is a homogeneous ensemble. It is clear that neural network design choices, such as using several layers instead of one and convolutions instead of locally connected units, lead to the biggest gains in the final model. These variants of the different neural network models can be easily assembled following the same principle as the SCNE building process. These models include the generalized SLFNs (GSLFNs) with direct links between the input layer and the output layer, recurrent neural networks (RNNs) with internal memory to process arbitrary sequences of inputs, as well as deep neural networks (DNNs) with a cascade of multiple layers for learning representation, like ConvNets and DeepSCN.

As previously mentioned, one way of improving the system is to look beyond the homogeneous ensemble. We could take advantages of different types of models and stack them together into a heterogeneous ensemble system to achieve the requirements of the modeling task and improve performance. Furthermore, this stacking ensemble system can be augmented gradually which makes system maintenance easier and cost effective. However, it will raise new challenge as to how to aggregate these models quickly and effectively. We believe that following a research path similar to SCNE, more useful solutions will be derived which will be worth exploring.

Before ending this thesis, we would like to restate the philosophy of the randomized ensemble learning methods. There is a large volume of research on randomized methods for building neural networks, but due to the nature of randomness, there is always a trade-off between training time and performance. Compared with the optimized model, the randomized model usually takes less time in training but ends up with less optimal performance. Furthermore, this process is usually not as stable as expected, but the ensemble learning techniques could solve this problem and make improvements using different aggregation approaches. The ensemble system also makes more sense when we consider the randomized model as a sample of a random variable from the view of statistics.

Finally, we believe that our SCNE framework can be used to facilitate industrial modeling tasks. In the case of FMF machine power demand prediction, we modified the base learner with a linear node to improve the prediction results and make the model interpretable. There is deeper research on the SCN in relation to fast computing and node pruning for solving different modeling tasks. New methods and applications are anticipated. Again, we see the possibility of using the SCNE as a powerful modeling framework for modern data analytics in the future.

Bibliography

[1] S. Abe. *Feature selection and extraction*, pages 331–341. Springer, 2010.

[2] M. Alhamdoosh and D. Wang. Fast decorrelated neural network ensembles with random weights. *Information Sciences*, 264:104–117, 2014.

[3] F. Anifowose, J. Labadin, and A. Abdulraheem. Improving the prediction of petroleum reservoir characterization with a stacked generalization ensemble model of support vector machines. *Applied Soft Computing*, 26:483–496, 2015.

[4] R. Avnimelech and N. Intrator. Boosted mixture of experts: an ensemble learning scheme. *Neural Computation*, 11(2):483–497, 1999.

[5] Z. J. Bai, M. Fahey, and G. Golub. Some large-scale matrix computation problems. *Journal of Computational and Applied Mathematics*, 74(1-2):71–89, 1996.

[6] E. Bair, T. Hastie, D. Paul, and R. Tibshirani. Prediction by supervised principal components. *Journal of the American Statistical Association*, 101(473):119–137, 2006.

[7] G. Bontempi and S. Ben Taieb. Statistical foundations of machine learning. *Universit Libre de Bruxelles*, 2011.

[8] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine learning*, 3(1):1–122, 2011.

[9] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[10] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[11] D. S. Broomhead and D. Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Report, Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.

[12] G. Brown, J. L. Wyatt, and P. Tino. Managing diversity in regression ensembles. *Journal of Machine Learning Research*, 6:1621–1650, 2005.

[13] G. Brown and J. M. Wyatt. Negative correlation learning and the ambiguity family of ensemble methods. *Multiple Classifier Systems, Proceeding*, 2709:266–275, 2003.

[14] F. L. Cao, D. H. Wang, H. Y. Zhu, and Y. G. Wang. An iterative learning algorithm for feedforward neural networks with random weights. *Information Sciences*, 328:546–557, 2016.

[15] H. Cecotti. Deep random vector functional link network for handwritten character recognition. In *International Joint Conference on Neural Networks (IJCNN)*, pages 3628–3633. IEEE.

[16] C. P. Chen and J. Z. Wan. A rapid learning and dynamic stepwise updating algorithm for flat neural networks and the application to time-series prediction. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(1):62–72, 1999.

[17] H. H. Chen and X. Yao. Regularized negative correlation learning for neural network ensembles. *IEEE Transactions on Neural Networks*, 20(12):1962–1979, 2009.

[18] J. X. Chen. The evolution of computing: Alphago. *Computing in Science and Engineering*, 18(4):4–7, 2016.

[19] R. Clarke, H. W. Ressom, A. Wang, J. Xuan, M. C. Liu, E. A. Gehan, and Y. Wang. The properties of high-dimensional data spaces: implications for exploring gene and protein expression data. *Nature Reviews Cancer*, 8(1):37, 2008.

[20] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth International Conference on Artificial Intelligence and Statistics*, pages 215–223.

[21] C. Cui and D. Wang. High dimensional data regression using lasso model and neural networks with random weights. *Information Sciences*, 372:505–517, 2016.

[22] T. G. Dietterich. Ensemble methods in machine learning. *Multiple Classifier Systems*, 1857:1–15, 2000.

[23] D. L. Donoho. For most large underdetermined systems of linear equations the minimal. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(6):797–829, 2006.

[24] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–451, 2004.

[25] J. Q. Fan, R. Samworth, and Y. C. Wu. Ultrahigh dimensional feature selection: Beyond the linear model. *Journal of Machine Learning Research*, 10:2013–2038, 2009.

[26] Y. Freund. An adaptive version of the boost by majority algorithm. *Machine Learning*, 43(3):293–318, 2001.

[27] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[28] J. Friedman, T. Hastie, and R. Tibshirani. A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*, 2010.

[29] M. Garcia-Torres, F. Gomez-Vela, B. Melian-Batista, and J. M. Moreno-Vega. High-dimensional feature selection via feature grouping: A variable neighborhood search approach. *Information Sciences*, 326:102–118, 2016.

[30] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

[31] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[32] A. N. Gorban, I. Y. Tyukin, D. V. Prokhorov, and K. I. Sofeikov. Approximation with random bases: Pro et contra. *Information Sciences*, 364:129–145, 2016.

[33] J. Han, J. Pei, and M. Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[34] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[35] G. E. Hinton and R. R. Salakhutdinov. Replicated softmax: an undirected topic model. In *Advances in Neural Information Processing Systems*, pages 1607–1614.

[36] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[37] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[38] D. Husmeier. *Neural networks for conditional probability estimation: Forecasting beyond point predictions.* Springer Science and Business Media, 2012.

[39] B. Igelnik and Y.-H. Pao. Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *IEEE Transactions on Neural Networks*, 6(6):1320–1329, 1995.

[40] B. Igelnik, Y. H. Pao, S. R. LeClair, and C. Y. Shen. The ensemble approach to neural-network learning and generalization. *IEEE Transactions on Neural Networks*, 10(1):19–30, 1999.

[41] M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[42] I. Kamkar, S. K. Gupta, D. Phung, and S. Venkatesh. Stable feature selection for clinical prediction: Exploiting icd tree structure using tree-lasso. *Journal of Biomedical Informatics*, 53:277–290, 2015.

[43] M. J. Kearns. *The computational complexity of machine learning.* MIT press, 1990.

[44] T. Y. Kwok and D. Y. Yeung. Objective functions for training new hidden units in constructive neural networks. *IEEE Transactions on Neural Networks*, 8(5):1131–1148, 1997.

[45] H. Larochelle and Y. Bengio. Classification using discriminative restricted boltzmann machines. In *Proceedings of the 25th International Conference on Machine learning*, pages 536–543. ACM.

[46] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[47] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual International Conference on Machine Learning*, pages 609–616. ACM.

[48] J. Y. Li, W. S. Chow, B. Igelnik, and Y. H. Pao. Comments on" stochastic choice of basis functions in adaptive function approximation and the functional-link net"[with reply]. *IEEE Transactions on Neural Networks*, 8(2):452–454, 1997.

[49] M. Li and D. Wang. Insights into randomized algorithms for neural networks: Practical issues and common pitfalls. *Information Sciences*, 382:170–178, 2017.

[50] W. Li, K. Chen, and D. Wang. Industrial image classification using a randomized neural-net ensemble and feedback mechanism. *Neurocomputing*, 173:708–714, 2016.

[51] W. T. Li, D. H. Wang, and T. Y. Chai. Multisource data ensemble modeling for clinker free lime content estimate in rotary kiln sintering processes. *IEEE Transactions on Systems Man Cybernetics-Systems*, 45(2):303–314, 2015.

[52] Y. Li, B. L. Lu, and T. F. Zhang. Combining feature selection with extraction: Unsupervised feature selection based on principal component analysis. *International Journal on Artificial Intelligence Tools*, 18(6):883–904, 2009.

[53] H. Liu and H. Motoda. *Feature extraction, construction and selection: A data mining perspective*, volume 453. Springer Science and Business Media, 1998.

[54] Y. Liu. Generate different neural networks by negative correlation learning. *Advances in Natural Computation, Pt 1, Proceedings*, 3610:149–156, 2005.

[55] Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399–1404, 1999.

[56] Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, 2000.

[57] D. Lopez-Paz, P. Hennig, and B. Schlkopf. The randomized dependence coefficient. In *Advances in Neural Information Processing Systems*, pages 1–9.

[58] D. Lowe and D. Broomhead. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2(3):321–355, 1988.

[59] Z. Q. J. Lu. The elements of statistical learning: Data mining, inference, and prediction, 2nd edition. *Journal of the Royal Statistical Society Series a-Statistics in Society*, 173:693–694, 2010.

[60] M. Lukosevicius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[61] S. Masoudnia and R. Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2):275–293, 2014.

[62] L. Meier, S. A. van de Geer, and P. Buhlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society Series B-Statistical Methodology*, 70:53–71, 2008.

[63] M. Mitchell. *An introduction to genetic algorithms.* MIT press, 1998.

[64] H. Mller and J.-C. Freytag. *Problems, methods, and challenges in comprehensive data cleansing.* Professoren des Inst. Fr Informatik, 2005.

[65] E. Nowakowska, J. Koronacki, and S. Lipovetsky. Dimensionality reduction for data of unknown cluster structure. *Information Sciences*, 330:74–87, 2016.

[66] A. Okun and A. Jackson. Conversations with alphago. *Nature*, 550(7676):337–337, 2017.

[67] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

[68] Y. H. Pao, G. H. Park, and D. J. Sobajic. Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing*, 6(2):163–180, 1994.

[69] Y. H. Pao and S. M. Phillips. The functional link net and learning optimal control. *Neurocomputing*, 9(2):149–164, 1995.

[70] Y. H. Pao and Y. Takefuji. Functional-link net computing: theory, system architecture, and functionalities. *Computer*, 25(5):76–79, 1992.

[71] G. H. Park and Y. H. Pao. Unconstrained word-based approach for off-line script recognition using density-based random-vector functional-link net. *Neurocomputing*, 31(1-4):45–65, 2000.

[72] J. Pearl. Causal inference in statistics: An overview. *Statistics Surveys*, 3:96–146, 2009.

[73] H. C. Peng, F. H. Long, and C. Ding. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005.

[74] T. Poggio, R. Rifkin, S. Mukherjee, and P. Niyogi. General conditions for predictivity in learning theory. *Nature*, 428(6981):419–422, 2004.

[75] A. Rahimi and B. Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in Neural Information Processing Systems*, pages 1313–1320, 2009.

[76] Y. Ren, P. N. Suganthan, N. Srikanth, and G. Amaratunga. Random vector functional link network for short-term electricity load demand forecasting. *Information Sciences*, 367:1078–1093, 2016.

[77] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti. Detecting novel associations in large data sets. *Science*, 334(6062):1518–1524, 2011.

[78] S. Romano, N. X. Vinh, K. Verspoor, and J. Bailey. The randomized information coefficient: assessing dependencies in noisy data. *Machine Learning*, 107(3):509–549, 2018.

[79] B. E. Rosen. Ensemble learning using decorrelated neural networks. *Connection Science*, 8(3-4):373–384, 1996.

[80] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[81] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*, pages 791–798. ACM, 2007.

[82] S. Scardapane, D. Comminiello, M. Scarpiniti, and A. Uncini. A semi-supervised random vector functional-link network based on the transductive framework. *Information Sciences*, 364:156–166, 2016.

[83] S. Scardapane and D. Wang. Randomness in neural networks: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(2), 2017.

[84] S. Scardapane, D. Wang, and M. Panella. A decentralized training algorithm for echo state networks in distributed big data applications. *Neural Networks*, 78:65–74, 2016.

[85] S. Scardapane, D. Wang, M. Panella, and A. Uncini. Distributed learning for random vector functional-link networks. *Information Sciences*, 301:271–284, 2015.

[86] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[87] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[88] W. F. Schmidt, M. A. Kraaijveld, and R. P. Duin. Feedforward neural networks with random weights. In *Proceedings 11th International Conference on Pattern Recognition, Vol. II. Conference B: Pattern Recognition Methodology and Systems*, pages 1–4. IEEE, 1992.

[89] M. P. Sesmero, A. I. Ledezma, and A. Sanchis. Generating ensembles of heterogeneous classifiers using stacked generalization. *Wiley Interdisciplinary Reviews-Data Mining and Knowledge Discovery*, 5(1):21–34, 2015.

[90] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[91] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. T. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[92] M. Stinchcombe and H. White. Approximating and learning unknown mappings using multilayer feedforward networks with bounded weights. In *International Joint Conference on Neural Networks*, pages 7–16. IEEE, 1990.

[93] Y. Sun, S. Todorovic, and S. Goodison. Local-learning-based feature selection for high-dimensional data analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1610–1626, 2010.

[94] R. S. Sutton and S. D. Whitehead. Online learning with random representations. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 314–321, 1993.

[95] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B-Methodological*, 58(1):267–288, 1996.

[96] R. Tibshirani. Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society Series B-Statistical Methodology*, 73:273–282, 2011.

[97] R. J. Tibshirani. Univariate shrinkage in the cox model for high dimensional data. *Statistical Applications in Genetics and Molecular Biology*, 8(1), 2009.

[98] I. Y. Tyukin and D. V. Prokhorov. Feasibility of random basis function approximators for modeling and control. In *Control Applications,(CCA) Intelligent Control,(ISIC), 2009 IEEE*, pages 1391–1396. IEEE, 2009.

[99] P. Vincent, H. Larochelle, Y. Bengio, and P. A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103. ACM, 2008.

[100] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[101] D. Wang. Editorial: Randomized algorithms for training neural networks. *Information Sciences*, 364:126–128, 2016.

[102] D. Wang and C. Cui. Stochastic configuration networks ensemble with heterogeneous features for large-scale data analytics. *Information Sciences*, 417:55–71, 2017.

[103] D. Wang and M. Li. Deep stochastic configuration networks with universal approximation property. *arXiv: 1702.05639*, 2017.

[104] D. Wang and M. Li. Robust stochastic configuration networks with kernel density estimation for uncertain data regression. *Information Sciences*, 412:210–222, 2017.

[105] D. Wang and M. Li. Stochastic configuration networks: Fundamentals and algorithms. *IEEE Transactions on Cybernetics*, 47(10):3466–3479, 2017.

[106] H. S. Wang, G. D. Li, and C. L. Tsai. Regression coefficient and autoregressive order shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B-Statistical Methodology*, 69:63–78, 2007.

[107] S. Wold, A. Ruhe, H. Wold, and W. J. Dunn. The collinearity problem in linear-regression - the partial least-squares (pls) approach to generalized inverses. *Siam Journal on Scientific and Statistical Computing*, 5(3):735–743, 1984.

[108] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[109] D. H. Wolpert and W. G. Macready. An efficient method to estimate bagging's generalization error. *Machine Learning*, 35(1):41–55, 1999.

[110] Z. H. Wu, J. X. Zhou, Y. J. Chen, and S. F. Cheng. Genetic algorithm based selective neural network ensemble. In *In proceedings of the 17th International Joint Conference on Artificial Intelligence, Seattle, WA*, volume 2, pages 797–802, 2001.

[111] R. F. Xu and S. J. Lee. Dimensionality reduction by feature clustering for regression problems. *Information Sciences*, 299:42–57, 2015.

[112] A. Y. Yang, S. S. Sastry, A. Ganesh, and Y. Ma. Fast l_1-minimization algorithms and an application in robust face recognition: A review. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 1849–1852. IEEE, 2010.

[113] J. Yang and T. Chai. Data-driven demand forecasting method for fused magnesium furnaces. In *Intelligent Control and Automation (WCICA), 2016 12th World Congress on*, pages 2015–2022. IEEE, 2016.

[114] J. Yang, S. Lu, L. Wang, and Q. Xu. Intelligent cloud based monitoring system of electricity demand of fused magnesium furnace process. *International Journal of Simulation and Process Modelling*, 12(5):389–400, 2017.

[115] D. M. Young. *Iterative solution of large linear systems.* Elsevier, 2014.

[116] L. Zhang and P. N. Suganthan. A comprehensive evaluation of random vector functional link networks. *Information sciences*, 367:1094–1105, 2016.

[117] L. Zhang and P. N. Suganthan. A survey of randomized algorithms for training neural networks. *Information Sciences*, 364:146–155, 2016.

[118] L. Zhang and P. N. Suganthan. Visual tracking with convolutional random vector functional link network. *IEEE Transactions on Cybernetics*, 47(10):3243–3253, 2017.

[119] Z. Zhou. *Ensemble methods: foundations and algorithms.* Chapman and Hall/CRC, 2012.

[120] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.