Study on the Predictions of Gene Function and Protein Structure Using Multi-SVM and Hybrid EDA

CHEN, Benhui

January, 2011
Waseda University Doctoral Dissertation

Study on the Predictions of Gene Function and Protein Structure Using Multi-SVM and Hybrid EDA

CHEN, Benhui

Graduate School of Information, Production and Systems

Waseda University

January, 2011
Abstract

How to efficiently extract the useful information from high throughput biological data is one of the main challenges in bioinformatics. Machine learning techniques such as Support Vector Machines (SVMs), Neural Networks (NNs), Estimation of Distribution Algorithms (EDAs) have been effective in analyzing biological data because of their capabilities in handling uncertain data noise and in generalization. Gene function prediction and protein structure prediction are two important domains where machine learning techniques are applied in bioinformatics. From the view of machine learning, the gene function predictions can be seen as multi-label classification or hierarchical multi-label classification (HMC) tasks; the protein structure predictions can be seen as optimization tasks that search for the protein folding structure with minimum energy based on its amino acid sequence.

For complicated tasks in bioinformatics, applying machine learning methods simply usually cannot obtain expectable results. The different characteristics and challenges of the tasks will require novel modifications of existing machine learning methods. In this thesis, some improved methods based on multi-SVM and hybrid EDA are developed for solving complicated tasks in the predictions of gene function and protein HP model. According to the characteristics of applications, the prior knowledge of tasks and the information of biological data are extracted by some delicate techniques. The obtained knowledge and information are used to enhance machine learning methods. SVMs based methods are developed for solving different applications of the gene function prediction. EDAs based methods are proposed for solving protein HP model problems. The Dill’s HP-lattice model is a simplified model of protein structure prediction. It is a useful tool for investigating general properties of protein folding.

In gene function classification applications, there are some datasets with characteristics of high noise, large number of input features and relatively small number of training examples. For those datasets, conventional nonlinear classification methods are unavailable due to the over-fitting problem. A new multi-SVM system called composite kernel based SVM (ck-SVM) is proposed to realize a flexibility controllable classifier which can solve nonlinear tasks and avoid over-fitting in certain degree. The proposed ck-SVM method realizes a nonlinear separating boundary by estimating a series of piecewise local linear boundaries. A local linear multi-SVM system is estimated by a composite kernel incorporating prior knowledge of training data. Instead of building multiple local SVM models separately, the prior knowledge of local subsets is used to construct a composite kernel, and the local linear multi-SVM model is realized exactly in the same way as a single SVM model by using the composite kernel.
In multi-label gene function classification, a training example is associated with a set of labels and the task is to predict the label set for each unseen instance. The main challenge of this problem is that classes are usually overlapped and correlated. An improved method based on label ranking and delicate decision boundary SVM is proposed for solving the multi-label gene classification. The label ranking strategy requires a real-valued score for each class (label) to order a label rank, then classifies a new instance into the classes that ranking score above a threshold. In order to obtain a proper label rank, an improved probabilistic SVM with delicate decision boundary is used as the scoring method. It can improve the probabilistic label rank by introducing the information of overlapped training samples into learning procedure. Instead of estimating a static threshold for all testing instances, an instance-independent thresholding strategy is proposed to decide the classification results. It can estimate an appropriate threshold for each testing instance according to the characteristics of instance and label rank.

Hierarchical multi-label classification (HMC) is a special variant of multi-label classification where the classes are organized in a hierarchy. FunCat prediction problem is a difficult task of HMC. In FunCat datasets, there are hundreds of functional classes structured by a predefined hierarchy, and the example distributions of classes are usually high degree skewed (imbalanced) with negative more than positive. An improved method based on over-sampling and hierarchy constraint is proposed for solving the FunCat prediction. Two measures are implemented to improve the HMC performance by introducing the hierarchy constraint into learning procedures. Firstly, for imbalanced functional classes, a hierarchical SMOTE is proposed as over-sampling preprocessing to improve the SVM learning performance. Secondly, an improved True Path Rule consistency approach is introduced to ensemble the results of binary probabilistic SVM classifications. It can correct the classification results and guarantee the hierarchy constraint of classes.

The protein HP model prediction is an NP-complete optimization problem. The task is to predict the minimum energy lattice structure of a protein from its simplified two-letter (H and P) amino acid sequence. Evolutionary Algorithms (EAs) based methods can be used to solve this NP-complete problem. But for long protein sequences, conventional methods can only find the suboptimum solutions. A hybrid EDA is proposed for solving the protein HP model. In order to select better individuals for probabilistic model of EDA, a composite fitness function containing the information of folding structure core (H-Core) is introduced to replace the traditional fitness function of HP model. Then, local search with guided operators is utilized to refine found solutions for improving efficiency of EDA. In addition, an improved backtracking-based repairing method is proposed to repair invalid individuals sampled by the probabilistic model of EDA. It can significantly reduce the number of backtracking searching operation and the computational cost for long sequence protein.

For optimization problems with irregular and complex multimodal landscapes, EDAs always suffer from the drawback of premature convergence similar to other evolutionary algorithms. An adaptive niching EDA with balance searching based on clustering analysis is proposed for solving complex optimization problems. The Affinity Propagation (AP) clustering is used to adaptively partition the population into niches during a run of EDA firstly. Then, a niche capacity selection mechanism based on the Boltzmann scheme is introduced to realize a balance searching between exploration and exploitation. Two different selection strategies, novelty-proportionate selection based
on the searching history information and fitness-proportionate selection based on the best fitness in a niche, are assembled by a Boltzmann weight. Tuned by the Boltzmann weight, at the beginning phase of searching, the dominating novelty-proportionate selection can guide the EDA searching to cover the search space as much as possible; at the end phase of searching, the dominating fitness-proportionate selection can enforce the EDA searching to exploit the already found promising areas. The proposed adaptive niching EDA is used for solving the protein HP model prediction. In addition, three benchmark functional multimodal optimization problems are also used to evaluate the proposed method.
Preface

The common theme of this thesis is developing some improved methods based on machine learning techniques for solving the prediction problems of gene function and protein structure. The material is organized in seven chapters. Most of the material has been published or considered to publish in journal papers and conference papers.

The material in Chapter 2 can be found in


The material in Chapter 3 can be found in


The material in Chapter 4 can be found in

which has been extended into a journal paper


The material in Chapter 5 can be found in


The material in Chapter 6 can be found in

Acknowledgements

I would like to take the opportunity to thank the following people who have supported me through my PhD experience at Waseda University.

First of all, I am heartily thankful to my supervisor, professor Jinglu Hu, whose encouragement, guidance and support from the initial to the final level enabled me to complete my PhD research. His tireless pursuit of excellence in research, teaching, and scientific writing is truly inspirational. His remarkable trust and support are essential for my study and research.

I am very grateful to Professor Kotaro Hirasawa, Professor Mizuho Iwaihara and Associate Professor Lianming Sun who proofread all the thesis. Many thanks are for their valuable comments.

I have had many valuable experiences at Waseda University during my graduate study, so I would like to acknowledge my classmates in Neural Computing Lab (Boyang Li, Jun Zhang, Yang Chen, Wenxiang Dou, Qiangwei Wang, Yu Cheng, Lan Wang, Bo Zhou, Zhan Shi, Yuling Lin et al.) for giving constructive suggestions to my thesis and generously supporting my academic career.

I would also like to express my thanks to my research group (Long Li, Liangpeng Ma, Weifeng Gu and Feiran Sun). I have learned so much from all of you, from figuring out what research is, to choosing a research agenda, to learning how to present my work. Your constructive criticism and collaboration have been tremendous assets throughout my PhD research.

Finally, I would like to dedicate this work to my parents, Chaogui Chen and Mingfang Kong, my wife Xuefei Hong and my daughter Yi Chen. Without your unending support and encouragement, I would not have finished the degree. Thank you.

Kitakyushu, Japan

Benhui Chen

October 20, 2010
Table of Contents

Abstract i
Preface v
Acknowledgements vii
Table of Contents ix
List of Tables xiv
List of Figures xvi
Glossary xvii
  Notations .......................................................... xvii
  Operators and Functions .............................................. xvii
  Abbreviations ........................................................ xviii

1 Introduction and Motivation 1
  1.1 Machine Learning in Bioinformatics ......................... 1
  1.2 Gene Function Prediction ......................................... 2
    1.2.1 Functional Genomics ....................................... 2
    1.2.2 Gene Function Prediction .................................... 4
  1.3 Protein Structure Prediction on HP Model ....................... 5
    1.3.1 Protein and Protein Folding ................................ 5
    1.3.2 Hydrophobic Polar (HP) Model ............................... 8
  1.4 Challenges ....................................................... 10
    1.4.1 Challenges in Prediction of Gene Function .................. 10
    1.4.2 Challenges in Prediction of Protein HP model ................ 11
  1.5 Goals of the Thesis ............................................. 12
  1.6 Thesis Outlines and Main Contributions ......................... 12
2 Support Vector Machine with Composite Kernel for Classification

2.1 Introduction .................................................. 17
2.2 A Composite Kernel Based SVM (ck-SVM) Model ................. 19
  2.2.1 Formulations of Model ................................... 19
  2.2.2 Properties of Model ..................................... 21
2.3 Estimation of the Proposed ck-SVM Model ......................... 22
  2.3.1 Affinity Propagation Clustering ............................ 22
  2.3.2 Partitioning Based on Separating Boundary Detection ....... 23
  2.3.3 Estimation of the Proposed ck-SVM Model ............... 25
2.4 Experiments and Results ..................................... 25
  2.4.1 Experiment Data Sets ..................................... 25
  2.4.2 Evaluation Metrics ....................................... 26
  2.4.3 Experiment Setting and Results ......................... 27
2.5 Conclusions ................................................. 27

3 Multi-label Classification Based on Label Ranking and Delicate Boundary SVM for Functional Genomics

3.1 Introduction ................................................. 29
3.2 Probabilistic Outputs of SVM Classification and Ranking Based Multi-label Classification ......................... 32
  3.2.1 Probabilistic Outputs for SVMs .......................... 32
  3.2.2 Ranking Based Multi-label Classification ............... 33
  3.2.3 Threshold Selection Strategies .......................... 34
3.3 Proposed Ranking Based Multi-label Classification ............... 34
  3.3.1 Scoring Method Based on SVM with Delicate Decision Boundary .... 35
  3.3.2 Instance-dependent Thresholding Strategy ............... 40
3.4 Experiments .................................................. 42
  3.4.1 Benchmark Datasets ....................................... 42
  3.4.2 Evaluation metrics for multi-label classification ........ 43
  3.4.3 Experimental Setting and Results ........................ 44
3.5 Conclusions ................................................. 46

4 Hierarchical Multi-label Classification Based on Over-sampling and Hierarchy Constraint for Gene Function Prediction

4.1 Introduction ................................................. 47
4.2 Hierarchical Multi-label Classification .......................... 49
  4.2.1 Notation and Definition ................................. 49
  4.2.2 Imbalanced Dataset Learning ............................ 50
  4.2.3 TPR Consistency Ensemble ............................... 51
4.3 Improved HMC Method ....................................... 51
  4.3.1 Main Frame of the Proposed Method ..................... 51
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Part attributes of the yeast genome (S. cerevisiae) benchmark datasets.</td>
<td>5</td>
</tr>
<tr>
<td>1.2</td>
<td>Assignment of residues as Hydrophobic or Polar.</td>
<td>8</td>
</tr>
<tr>
<td>2.1</td>
<td>Properties of experiment datasets</td>
<td>26</td>
</tr>
<tr>
<td>2.2</td>
<td>Descriptions of classes (labels).</td>
<td>26</td>
</tr>
<tr>
<td>2.3</td>
<td>Results of comparing with the RBF kernel and linear SVM for FunCat “Sequence” dataset.</td>
<td>28</td>
</tr>
<tr>
<td>2.4</td>
<td>Results of comparing with the RBF kernel and linear SVM for FunCat “All microarray” dataset.</td>
<td>28</td>
</tr>
<tr>
<td>3.1</td>
<td>Characteristics of the experimental benchmark datasets</td>
<td>43</td>
</tr>
<tr>
<td>3.2</td>
<td>Experimental results for Yeast dataset.</td>
<td>45</td>
</tr>
<tr>
<td>3.3</td>
<td>Experimental results for Genbase dataset.</td>
<td>45</td>
</tr>
<tr>
<td>4.1</td>
<td>Properties of experiment datasets</td>
<td>58</td>
</tr>
<tr>
<td>4.2</td>
<td>Experiment results of different over-sampling strategies based on the FunCat subtree of “Protein fate” (ID=14).</td>
<td>60</td>
</tr>
<tr>
<td>4.3</td>
<td>Experiment results of different ensemble strategies based on the FunCat subtree of “Protein fate” (ID=14).</td>
<td>62</td>
</tr>
<tr>
<td>4.4</td>
<td>Classification results on the four FunCat datasets.</td>
<td>63</td>
</tr>
<tr>
<td>5.1</td>
<td>HP instances used in the experiments.</td>
<td>80</td>
</tr>
<tr>
<td>5.2</td>
<td>Results of comparing with MK-EDA for 2-D HP model.</td>
<td>81</td>
</tr>
<tr>
<td>5.3</td>
<td>Results achieved by different search methods for 2-D HP model.</td>
<td>82</td>
</tr>
<tr>
<td>5.4</td>
<td>Results achieved by different search methods for 3-D HP model.</td>
<td>83</td>
</tr>
<tr>
<td>5.5</td>
<td>Comparing results of the improved backtracking repairing method in 2-D HP model.</td>
<td>84</td>
</tr>
<tr>
<td>6.1</td>
<td>Parameters of the Shekel function ($n = 5$).</td>
<td>97</td>
</tr>
<tr>
<td>6.2</td>
<td>Settings for the three test functions.</td>
<td>97</td>
</tr>
<tr>
<td>6.3</td>
<td>Experimental results for the Kowalik function.</td>
<td>99</td>
</tr>
<tr>
<td>6.4</td>
<td>Experimental results for the Shekel function (Dim=30).</td>
<td>100</td>
</tr>
<tr>
<td>6.5</td>
<td>Experimental results for the Schwefel function.</td>
<td>100</td>
</tr>
<tr>
<td>6.6</td>
<td>Computing costs of the proposed method and the UMDA_c.</td>
<td>100</td>
</tr>
<tr>
<td>6.7</td>
<td>Results of comparing with MK-EDA and the proposed method for 2-D HP model.</td>
<td>101</td>
</tr>
</tbody>
</table>
6.8  Results of different search methods for 2-D HP model. . . . . . . . . . . . . . . . . 102
6.9  Results of different search methods for 3-D HP model. . . . . . . . . . . . . . . . . 102
# List of Figures

1.1 Information flows (from DNA to RNA to proteins) in the central dogma of biology. ...................... 2
1.2 A small part of the hierarchical FunCat classification scheme [79] ................................. 6
1.3 One possible configuration of the sequence $H P H P R H P H P H P R H P H P$ in 2-D HP model. There are four HH topological neighbors (represented by broken lines). 9
1.4 Flow diagram of this thesis .......................................................... 13

2.1 An example of multiple linear separating boundaries .................................................. 19
2.2 An example of partitioning approach. (a) the synthetic data (678 samples) constructed by shifted cosine signals with Gaussian noise; (b) the clustering results of implementing AP clustering onto the original synthetic data; (c) the selected samples (219 samples) by implementing the separation boundary detection; (d) the clustering results of implementing AP clustering onto the selected samples. 24

3.1 Main scheme of the proposed ranking based multi-label classification. ....................... 35
3.2 Main scheme of the SVM based scoring method ...................................................... 36
3.3 Bias model with delicate decision boundary .......................................................... 38
3.4 Proposed instance-dependent threshold selection based on KNN. .............................. 41
3.5 Relation between the parameter $k$ of KNN in threshold selection and the performance of classification for Yeast dataset .................................................. 45
3.6 Comparing results of the bias models for 12-th label in Yeast dataset ......................... 46

4.1 Main frame of the proposed HMC method. ............................................................... 52
4.2 Illustration of the hierarchical SMOTE. ................................................................. 53
4.3 Rooted tree of the FunCat classes of ID=14 (Protein fate). ...................................... 61
4.4 Three multi-label metrics on the FunCat subtree of ID=14 (Protein fate). ...................... 61

5.1 One possible configuration of the sequence $H P H P R H P H P H P H P H P$ in 2-D HP model. There are six HH topological neighbors (represented by broken lines). .................. 68
5.2 Illustration of the protein folding H-Core in 2-D HP model. ...................................... 71
5.3 Two example solutions with same energy (Instance s8 in Tab. 5.1, length: 64, energy: -25). .................................................. 72
5.4 Guided operators for local search. ....................................................................... 73
5.5 Pseudocode of the backtracking repairing algorithm. .............................................. 75
5.6 Illustration of the closed-areas in 2-D HP model. .................................................... 76
5.7 (a) Illustration of closed-area detection. (b) The situation that need to implement backtracking. ................................................................. 77
5.8 Pseudocode of the detection procedure. ........................................ 78
5.9 The best fitness for one representative run of instance s7. ............... 82
5.10 The number of backtracking searching operations for instance s8. ....... 84
5.11 The number of backtracking searching operations for instance s9. ....... 85
5.12 The number of backtracking searching operations for instance s10. ...... 85

6.1 Selection procedure of the adaptive niching EDA. ............................. 91
6.2 Population clustering center exemplars of Schwefel function (dimension=2) optimized by UMDA_c. .................................................. 93
6.3 Framework of the proposed adaptive Niching EDA ............................ 95
6.4 Plot of the Shekel function (dimension=2). ...................................... 98
6.5 Plot of the Schwefel function (dimension=2). ................................... 99
6.6 The number of different individuals in the selected population for one representative run of instance s10. ......................................... 103
6.7 One conformation of the best results for the instance s5 in 3-D HP model (fitness=-29, star denotes H and circle denotes P). ................. 103
6.8 One conformation of the best results for the instance s7 in 3-D HP model (fitness=-49, star denotes H and circle denotes P). ....................... 104
6.9 One conformation of the best results for the instance s8 in 3-D HP model (fitness=-51, star denotes H and circle denotes P). ....................... 104

B.1 The flowchart of Estimation of Distribution Algorithms (EDAs). .......... 122

C.1 Two messages are passed between data points: (a)”responsibilities” r(i,k) are sent from data point i to candidate exemplar k; (b) “availabilities” a(i,k) are sent from candidate exemplar k to data point i. ........................................ 129
Glossary

Some notations may have different meaning locally.

**Notations**

\( x^T \)  
transpose

\( b \)  
bias

\( \Omega \)  
coordinate parameter vector of local linear boundary in Local Linear Multi-SVM model

\( \alpha_k \)  
Lagrange multipliers

\( p \)  
preferences parameter in Affinity Propagation

\( \#SV \)  
number of support vectors

\( \sigma \)  
radius of niching clearing procedure

\( \kappa \)  
capacity of niching clearing procedure

**Operators and Functions**

\( \text{sign}(x) \)  
an odd mathematical function that extracts the sign of \( x \)

\( \mathcal{L}(\cdot) \)  
Lagrangian function in construction of optimization problems

\( K(x_i, x_k) \)  
kernel function in SVM

\( \sum \)  
sum

\( y(x) \)  
classifier function

\( f(x) \)  
decision value function

\( f_P(x) \)  
local linear multi-SVM model

\( \mathcal{R}(x) \)  
output of basis function in local linear multi-SVM model

\( s(i, k) \)  
similarity function in Affinity Propagation

\( r(i, k) \)  
responsibility function in Affinity Propagation

\( a(i, k) \)  
availability function in Affinity Propagation

\( p(y = 1|x) \)  
probabilistic output for SVM classifier

\( C(\cdot) \)  
outputs of multi-label classification based on label ranking

\( R(\cdot) \)  
outputs of ranking method for multi-label classification

\( T(\cdot) \)  
outputs of threshold selection for multi-label classification

\( \varphi_i(x) \)  
the children set of node \( i \) which have a positive prediction for a given example \( x \) in Hierarchical Multi-label Classification
\( E(x) \)  
energy of protein HP model  

\( Fit_c p(x) \)  
fitness value of composite fitness function for protein HP model  

\( p_{MK}(X) \)  
k-order Markov probabilistic model for EDAs  

\( \kappa_i(n_i) \)  
niche capacity selection for the \( i \)-th niche in the \( t \)-th iteration of adaptive nicheng EDA

**Abbreviations**

- **ANN**: Artificial Neural Networks  
- **AP**: Affinity Propagation  
- **BOA**: Bayesian Optimization Algorithm  
- **DAG**: Directed Acyclic Graph  
- **EDA**: Estimation of Distribution Algorithm  
- **EcGA**: Extended compact Genetic Algorithm  
- **EA**: Evolutionary Algorithm  
- **FDA**: Factorized Distribution Algorithm  
- **FunCat**: Functional Catalogue of gene function prediction  
- **GA**: Genetic Algorithms  
- **GO**: Gene Ontology  
- **HMC**: Hierarchical Multi-label Classification  
- **KNN**: \( k \) Nearest Neighbors  
- **LS-SVM**: Least Squares Support Vector Machines  
- **PSO**: Particle Swarm Optimization  
- **PSP**: Protein Structure Prediction  
- **QP**: Quadratic Programming  
- **RBF**: Radial Basis Function  
- **SMOTE**: Synthetic Minority Over-sampling Technique  
- **SRM**: Structural Risk Minimization  
- **SV**: Support Vector  
- **SVM**: Support Vector Machine  
- **SVR**: Support Vector Regression  
- **TPR**: True Path Rule  
- **3-D**: Three Dimensional  
- **2-D**: Two Dimensional
Chapter 1

Introduction and Motivation

1.1 Machine Learning in Bioinformatics

How to extract useful information efficiently from high throughput biological data is one of main challenges in bioinformatics, which requires the development of tools and methods capable of transforming all these heterogeneous data into biological knowledge about the underlying mechanism. Novel computational techniques to analyze biological data in the form of sequences, gene and protein expressions, pathways, and images are becoming vital for understanding diseases and future drug discovery [6]. Machine learning techniques such as Markov models, Support Vector Machines (SVMs), Neural Networks (NNs), Evolutionary Algorithms (EAs) and graphical models have been effective in analyzing biological data because of their capabilities in handling randomness and uncertainty of data noise and in generalization [83, 122].

There are several biological domains where machine learning techniques are applied for knowledge extraction from data, such as genomics, proteomics, microarrays, systems biology, evolution and text mining, etc. Among them, applications in genomics and proteomics are our focuses in this thesis. 1) Genomics is one of the most important domains in bioinformatics. The number of sequences available is increasing exponentially, and these data need to be addressed in order to obtain useful information [75, 14, 17]. Assigning biological functions to the genome sequences is a key challenge in Genomics. Machine learning techniques, such as SVMs and NNs, are always used to predict gene functions from a predefined set of possible functions. 2) Proteins play a very important role in the life process, and their three-dimensional (3-D) structure is a key feature in their functionality. In the proteomic domain, the main application of computational methods is protein structure prediction. Proteins are very complex macromolecules with thousands of atoms and bounds. Hence,
the number of possible structures is huge. This makes protein structure prediction a very complicated combinatorial problem where optimization techniques, such as Genetic Algorithm (GA) and Estimation of distribution algorithm (EDA), are required.

Machine learning consists in programming computers to optimize a performance criterion by using example data or past experience. The optimized criterion can be the accuracy provided by a predictive model (in modeling methods, such as NNs, SVM, etc.), and the value of a fitness or evaluation function (in optimization methods, such as GA, EDAs, etc.) [55, 66]. In a modeling problem, the “learning” term refers to running a computer program to induce a model by using training data or past experience. Machine learning uses statistical theory when building computational models since the objective is to make inferences from a sample. The two main steps in this process are to induce the model by processing the huge amount of data and to represent the model and making inferences efficiently. Optimization problems can be posed as the task of finding an optimal solution in a space of multiple (sometimes exponentially sized) possible solutions.

### 1.2 Gene Function Prediction

#### 1.2.1 Functional Genomics

Functional genomics is a field of bioinformatics that attempts to make use of the vast biology data produced by genomic projects (such as genome sequencing projects) to describe gene (and protein) functions and interactions. The central dogma of biology is that DNA is transcribed into RNA and RNA is translated into proteins. Figure 1.1 shows the relationship between the DNA, RNA and protein. The gene function usually means the function of the products of genes after transcription and translation, which are proteins.
Genes and ORFs

Genes are the units of heredity. They are sections of DNA which encode the information needed to make an organism and determine the attributes of that organism. Gene-finding programs are used to hypothesize where the genes lie in a DNA sequence. When an appropriate stretch of DNA (reasonable length, starting and ending with the right parts, etc.) is found, it is labeled as an Open Reading Frame (ORF) - a putative gene.

DNA

DNA is the molecular code of cells. It is a long chain molecule, consisting of a backbone of alternate sugar and phosphate groups, with a base attached to each sugar. There are four different bases which can be attached, and the sequence of the bases along the backbone makes the code. The bases are Adenine (A), Guanine (G), Cytosine (C), and Thymine (T). From a computer science perspective, DNA is regarded as a long string made up of the four letters A, G, C and T. The main purpose of DNA is to encode and replicate the information needed to make proteins.

The four bases of DNA are used in different combinations to code for the all the 20 amino acids that make proteins. A triplet of DNA bases is used to code for each amino acid. Each triple of DNA is known as a codon. Apart from the codons which are used for amino acids, three of the triples are used to encode “stop” codons, which tell the cellular machinery where to stop reading the code.

DNA is double stranded. It exists as two long chain molecules entwined together in the famous double helix. The two strands have complementary base pairing, so each C in one strand is paired with a G in the other and each A with a T. So when the size of DNA is quoted, it is usually in “base pairs”.

RNA

DNA is translated to proteins via RNA. RNA is a nucleic acid, very similar to DNA but single stranded, and the four bases of RNA are A, G, C and U (Uracil replaces Thymine). RNA is used for several roles in the cell. Its primary role is to take a copy of one of the strands of DNA. This piece of RNA (known as messenger RNA) might then undergo splicing to remove introns, pieces of sequence which are non-coding, which interrupt the coding regions (exons) of a gene. Finally, the sequence of bases of RNA are then translated into amino acids to make the protein. Measurement of the RNA being produced (“expressed”) in a cell can be used to infer which proteins are being produced.
1.2.2 Gene Function Prediction

Determining the functions of genes and proteins is a foundation to understanding the molecular and biochemical processes that sustain health or cause disease, to identifying and validating new drug targets and to developing reliable diagnostics [100]. The completion of several genome projects in the past decade has generated the full genome sequence of many organisms. Assigning biological functions to the sequences has become a key challenge in modern biology. This step is often guided by automatic discovery processes which interact with the laboratory experiments. More precisely, machine learning techniques are often used to predict gene functions from a predefined set of possible functions (e.g., the functions in the FunCat or Gene Ontology). Afterwards, the predictions with highest confidence can be tested in the lab [100].

Many computational methods are used in functional genomics. The most common method of determining the function of a protein is to use a sequence similarity program to infer function by orthologous homology. Unfortunately, the sequence similarity method is always unavailable and unpractical. Many genes are annotated as “putative” or “potential” without any indication of where that information came from. In addition, if the original sequence was annotated wrongly, then this error may be propagated to the future gene annotation by sequence comparisons.

Recently, some general prediction methods are proposed by using data mining and machine learning to induce rules predicting function from a variety of data sources [5, 95, 10]. Some comprehensive statistic and analysis information, such as percentage of amino acids in the protein and molecular weight of the protein, are abstracted from gene sequence. The obtained statistic and analysis information are utilized as sample attributes for data mining and machine learning techniques. An example of the yeast genome (S. cerevisiae) benchmark dataset attributes is showed in Tab. 1.1. These methods are more general and accurate than conventional sequence similarity methods [29].

From the view of machine learning, the gene function predictions can be seen as multi-label or hierarchical multi-label classification tasks. Different from common machine learning tasks, there are two characteristics of the function predictions: 1) a single gene may have multiple functions; 2) the functions are organized in a hierarchy: a gene that is related to some function is automatically related to all its ancestor functions (this is called the hierarchy constraint). In practice, if the hierarchical information of functions can not be obtained or provided, the prediction task is a multi-label classification problem.

There are two main taxonomies of gene functional classes: the Gene Ontology (GO) [5] and the Functional Catalogue (FunCat) [95]. The GO is composed by thousands of functional classes
Table 1.1: Part attributes of the yeast genome (S. cerevisiae) benchmark datasets.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aa-rat-X</td>
<td>real</td>
<td>Percentage of amino acid X in the protein</td>
</tr>
<tr>
<td>seq-len</td>
<td>integer</td>
<td>Length of the protein sequence</td>
</tr>
<tr>
<td>aa-rat-pair-X-Y</td>
<td>real</td>
<td>Percentage of the pair of amino acids X and Y consecutively in the protein</td>
</tr>
<tr>
<td>mol-wt</td>
<td>integer</td>
<td>Molecular weight of the protein</td>
</tr>
<tr>
<td>theo-pl</td>
<td>real</td>
<td>Theoretical pl (isoelectric point)</td>
</tr>
<tr>
<td>atomic-comp-X</td>
<td>real</td>
<td>Atomic composition of X where X is c (carbon), o (oxygen), n (nitrogen), s (sulphur) or h (hydrogen)</td>
</tr>
<tr>
<td>aliphatic-index</td>
<td>real</td>
<td>The aliphatic index</td>
</tr>
<tr>
<td>hydro</td>
<td>real</td>
<td>Grand average of hydrophobicity</td>
</tr>
</tbody>
</table>

structured according to a directed acyclic graph (DAG), and it is set out in three separated ontologies: “Biological Processes”, “Molecular Function” and “Cellular Component”. Indeed a gene can participate to specific biological processes (e.g. cell cycle, metabolism, nucleotide biosynthesis) and at the same time can perform specific molecular functions (e.g. catalytic or binding activities, that occur at the molecular level) in specific cellular components (e.g. mitochondrion or rough endoplasmic reticulum) [5]. The FunCat represents a more simple and concise set of gene functional classes: it consists of 28 main functional categories (or branches) that cover general fields like cellular transport, metabolism and cellular communication/signal transduction. These main functional classes are divided into a set of subclasses with up to six levels of increasing specificity, according to a treelike structure that accounts for different functional characteristics of genes and gene products. Genes may belong at the same time to multiple functional classes, since several classes are subclasses of more general ones, and because a gene may participate in different biological processes and may perform different biological functions [112]. A small part of the FunCat taxonomies is showed in Fig. 1.2.

1.3 Protein Structure Prediction on HP Model

1.3.1 Protein and Protein Folding

Proteins are extremely important molecules which do almost all the work in the cell. They are involved in everything from immunity to muscle structure, transportation, hormones, metabolism,
respiration, repair, and control of genes. Understanding the roles of proteins is the key to understanding how the whole cell operates.

A protein molecule is composed of a string of amino acids (monomers) that are joined together in a series by peptide bonds. There are 20 different amino acids, so the protein can be represented by a string of characters for computational purposes. The structure and shape of the protein molecule (how the long chain of amino acids folds in 3-D space) is relevant to the job of the protein. Much work has been done on protein structure determination, as it gives clues to the protein function. Understanding protein structures is vital to determining the function of a protein and its interaction with DNA, RNA and enzyme. The information about its conformation can provide essential information for drug design and protein engineering.

A protein can be described in terms of its four main structural descriptions as follows. 1) Primary Structure refers to the linear sequence of protein amino acid units. These sequence acts as
instruction for protein, and is an outcome of the human genome project for mapping DNA. 2) **Secondary Structure** describes the protein in terms of regular patterns or substructures in the polypeptide backbone (independent of the amino acid side chains or offshoots from the main backbone of the molecule). These substructures are classified as one of three motifs: alpha-helix, beta-sheet and the random coil. 3) **Tertiary Structure** refers to the three-dimensional shape of the protein molecule. It describes the spatial relationship between the molecules secondary substructures. A protein is not functional until it is folded into its tertiary structure. 4) **Quaternary Structure** refers to structures and molecular relationships when protein molecules form in groups to construct larger molecules.

The protein three-dimensional tertiary structure is defined and folded by the protein primary form. The self-folding procedure requires specific environmental conditions such as the correct pH and in some cases chaperone or helper molecules. Proteins can be unfolded (denatured) using thermal or chemical kinetics, and some proteins can refold into their tertiary structures when the agent is removed. The spontaneous self-folding procedure can occur in microseconds, though large molecules can take minutes or hours to fold.

Protein structure prediction is a research area that focuses on accurately determining the tertiary structure of a protein from its primary structure. The protein tertiary structure can be determined through conventional structure elucidation experimentally, using X-ray crystallography and Nuclear Magnetic Resonance (NMR) spectroscopy to determine the position of atoms within the molecular environment. The conventional biological method is a slow and expensive process. While there are over a million known protein sequences, only a limited number of protein structures are experimentally determined. Hence, prediction of protein structures from protein sequences using computer programs is an important step to unveil proteins’ 3-D conformation and functions [6, 59, 13].

Under specific conditions, the protein sequence folds into a unique native 3-D structure. Each possible protein fold has an associated energy. The thermodynamic hypothesis states that the native structure of a protein is the one for which the free energy achieves the global minimum. Based on this hypothesis, many methods that search for the protein native structure define an approximation of the protein energy and use optimization algorithms that look for the protein fold that minimizes this energy. These approaches mainly differ in the type of energy approximation employed and in the characteristics of the protein modeling. Because of the complexity of protein structure prediction problem, simplified models like Dill’s HP-lattice model [68] have become a major tool for investigating general properties of protein folding.
Table 1.2: Assignment of residues as Hydrophobic or Polar.

<table>
<thead>
<tr>
<th>Amino Acid</th>
<th>1-Letter</th>
<th>Assignment</th>
<th>Amino Acid</th>
<th>1-Letter</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alanine</td>
<td>A</td>
<td>Hydrophobic</td>
<td>Arginine</td>
<td>R</td>
<td>Polar</td>
</tr>
<tr>
<td>Asparagine</td>
<td>N</td>
<td>Polar</td>
<td>Aspartic acid</td>
<td>D</td>
<td>Polar</td>
</tr>
<tr>
<td>Cysteine</td>
<td>C</td>
<td>Hydrophobic</td>
<td>Glutamic acid</td>
<td>E</td>
<td>Polar</td>
</tr>
<tr>
<td>Glutamine</td>
<td>Q</td>
<td>Polar</td>
<td>Glycine</td>
<td>G</td>
<td>Hydrophobic</td>
</tr>
<tr>
<td>Histidine</td>
<td>H</td>
<td>Polar</td>
<td>Isoleucine</td>
<td>I</td>
<td>Hydrophobic</td>
</tr>
<tr>
<td>Leucine</td>
<td>L</td>
<td>Hydrophobic</td>
<td>Lysine</td>
<td>K</td>
<td>Polar</td>
</tr>
<tr>
<td>Methionine</td>
<td>M</td>
<td>Hydrophobic</td>
<td>Phenylalanine</td>
<td>F</td>
<td>Hydrophobic</td>
</tr>
<tr>
<td>Proline</td>
<td>P</td>
<td>Hydrophobic</td>
<td>Serine</td>
<td>S</td>
<td>Hydrophobic</td>
</tr>
<tr>
<td>Threonine</td>
<td>T</td>
<td>Hydrophobic</td>
<td>Tryptophan</td>
<td>W</td>
<td>Hydrophobic</td>
</tr>
<tr>
<td>Tyrosine</td>
<td>Y</td>
<td>Hydrophobic</td>
<td>Valine</td>
<td>V</td>
<td>Hydrophobic</td>
</tr>
</tbody>
</table>

1.3.2 Hydrophobic Polar (HP) Model

A thermodynamical approach to the problem of tertiary structure prediction assumes that a folded protein seeks to minimize Gibbs free energy. One of the simplest models of this assumption further assumes that the major contribution to the free energy of a proteins native conformation is due to interactions between hydrophobic amino acids [36]. That is that amino acids with a hydrophobic residue in a protein sequence tend to push together, and that those amino acids with a hydrophilic residue tend to push to the outside of the molecule [35].

This simple and well studied abstraction is called the hydrophobic polar (HP) model of protein folding [68]. Commonly studied proteins consists of approximately 20 different amino acids (as showed in Tab. 1.2). The HP model discards all amino acid information in the sequence except for two properties, whether the amino acid is hydrophobic \( (H) \) or whether the amino acid is polar \( (P) \)(also known as hydrophilic). Table 1.2 shows the assignment of residues as hydrophobic or polar [36]. The amino acids are then organized on a 2-D or 3-D lattice such that the number of topologically adjacent \( H - H \) connections is maximized. The number of these hydrophobic connections is calculated and multiplied by a free energy constant (typically -1), thus transforming the problem to that of minimizing the free energy of lattice structures. The amino acid sequence is embedded in a lattice as a self-avoiding walk where a single vertex can be occupied by one amino acid and the walk cannot intersect itself.

In HP model, a protein is considered as a sequence \( S \in \{H, P\}^+ \), where \( H \) represents a hydrophobic residues and \( P \) represents a hydrophilic or polar residues. The HP model restricts
the space of conformations to self-avoiding paths on a lattice in which vertices are labeled by the residues. Given a pair of residues, they are considered neighbors if they are adjacent either in the chain (connected neighbors) or in the lattice but not connected in the chain (topological neighbors). Let $\varepsilon_{HH}$ denote the interaction energy between topological neighbor of two $H$ residues, $\varepsilon_{PP}$ for two $P$ residues, $\varepsilon_{HP}$ for a $H$ residue and a $P$ residue. An energy function is defined as the total energy of topological neighbors with $\varepsilon_{HH} = -1$ and $\varepsilon_{PP} = \varepsilon_{HP} = 0$. The HP problem is to find the folding conformation that minimizes the total energy $E(x)$. Figure 1.3 shows the graphical representation of a possible configuration for sequence $HPHPHHPHHPPHPHPHPHP$ in 2-D HP model, hydrophobic residuals are represented by black beads and polar residuals by white beads. The energy that the HP model associates with this configuration is -4. The problem of finding such a minimum energy configuration has been proved to be NP-complete for the 2-D and 3-D lattices [32, 12].

The HP model cannot be used to describe real protein or demonstrate folding of real protein. It is a simple abstraction that reduces all amino acid properties to two binary properties, and all structure properties of the protein to a discrete lattice. Although more complex models have been proposed, the HP model remains a focus of research in computational biology, chemical and statistical physics. By varying the energy function and the bead sequence of the chain (the primary structure), effects on the native state structure and the kinetics (rate) of folding can be explored, and this may provide insights into the folding of real proteins. In particular, the HP model has been used to investigate the energy landscapes of proteins, i.e. the variation of their internal free energy as a function of conformation. In evolutionary computation, the model is still employed as benchmark problem because of its simplicity and its usefulness as a test-bed for new evolutionary optimization.
approaches [96].

1.4 Challenges

For implementing machine learning techniques into difficult bioinformatics applications, the novel characteristics of the applications will require novel modifications of existing algorithms and procedures, and in some invention of entirely new techniques.

1.4.1 Challenges in Prediction of Gene Function

As mentioned in previous sections, the gene function predictions are considered as multi-label or hierarchical multi-label classification tasks (if hierarchical information of functions are provided). Multi-label classification problem is an extension of traditional multi-class classification problem in which the classes are not mutually exclusive and each sample may belong to several classes simultaneously. In this case, samples in training set are associated with a set of labels and the task is to predict the label set for the unseen instance. Hierarchical multi-label classification is a special variant of multi-label classification where the classes are organized in a hierarchy: an example that belongs to one class automatically belongs to all its super-classes.

In this thesis, the task of multi-label (or hierarchical multi-label) classification is transferred into a series of separate basis binary classification tasks, and one SVM classifier is trained to learn one class, then results of all binary classifiers are combined by considering the multi-label relationships between classes (and hierarchical constraints in hierarchical multi-label problem). According to the characteristics of gene function prediction, there are many challenges for classification methods in three different levels:

- For the basis binary SVM classifier, there are some special datasets (such as FunCat Yeast datasets [10] and Genbase motif-based datasets [37]), the nonlinear (RBF-kernel or other non-linear kernels) SVM models can not obtain proper classification results. These tasks are nonlinear problems with characteristics of high noise and large number of input features compared with the relatively small number of training examples, the conventional nonlinear kernel SVM models are severely overfitting at times [10].

- For the multi-label classification problem, the classes are not mutually exclusive and each sample may belong to several classes simultaneously. The main challenge of this problem is that classes are usually overlapped and correlated. Generally, traditional multi-class learning algorithms cannot work with multi-label problem effectively.
• For the hierarchical multi-label classification problem, the task is characterized by hundreds of classes structured according to a predefined hierarchy; and the example distributions of functional classes are usually high degree skewed (imbalanced) with negative more than positive. In such imbalance dataset learning, standard classifiers tend to be overwhelmed by the majority class and ignore the minority one.

1.4.2 Challenges in Prediction of Protein HP model

The problem of protein structure prediction on HP model has been proved to be NP-complete for the 2-D and 3-D lattices. Therefore, a deterministic approaches is always not practical for this problem. Many GA based methods have been proposed to solve the protein structure prediction on HP model in recent years [111, 49, 104, 31, 26]. Among them, the EDAs that use Markov probabilistic model or other probabilistic models proposed by Ref. [96] outperform other population-based methods when solving the HP model folding problem, especially for the long sequence protein instances. But the EDAs based methods still have some disadvantages as follows.

• For most long sequence protein instances, the chance of finding the global optimum is very low, and the algorithm often need be set by very large generation number and population size for finding the global optimum. For some deceptive sequences, those methods can only find the suboptimum solutions.

• In those EDAs based methods, a backtracking method is used to repair invalid individuals sampled by the probabilistic model of EDAs. For a traditional backtracking algorithm, the computational cost of repairing procedure is very heavy for those long sequence instances.

• For those optimization problems with irregular and complex multimodal landscapes, the EDAs still suffer from the drawback of premature convergence similar to other Evolutionary Algorithms (EAs). A simple yet popular explanation for the occurrence of premature convergence is the loss of diversity. But high diversity does not imply a better EA performance. This is tightly coupled to the question of exploration vs. exploitation. Enforcing diversity at the early phases of evolution ensures a broad exploration of the search space. However, at the phases when high exploitation is needed, such promotion could be counterproductive or even destructive. Finding a modifiable balance between exploration and exploitation is a key to this issue.
1.5 Goals of the Thesis

For complicated tasks in bioinformatics, applying machine learning methods simply usually cannot obtain expectable results. The different characteristics and challenges of the tasks will require novel modifications of existing machine learning methods. In this thesis, improved methods based on machine learning methods are developed and applied to the prediction of gene function and protein structure. More precisely, SVM based multi-label and hierarchical multi-label classification methods are developed for solving the gene function prediction problems, and hybrid EDA based methods are proposed for solving the protein HP model problems. According to the characteristics of applications, the prior knowledge of tasks and the information of biological data are extracted by some delicate techniques. The obtained knowledge and information are used to enhance machine learning methods.

The work presented here aims to assess the performances of the improved methods incorporating prior knowledge of application problems. The thesis also shows how the improved methods handle the aforementioned challenges inherent in the gene function prediction and protein structure prediction.

1.6 Thesis Outlines and Main Contributions

This thesis presents my work that has been done over the last three years. It consists of seven chapters. Chapter 1 gives a background and an outline for the whole thesis. Chapter 2 introduces a novel support vector machine with composite kernel for solving classification tasks in gene function prediction. Chapter 3 is devoted to multi-label classification based on label ranking and delicate boundary SVM for Functional Genomics. Chapter 4 introduces a hierarchical multi-label classification based on over-sampling and hierarchy constraint for FunCat gene function prediction. Chapter 5 proposes a hybrid EDA for protein structure prediction based on HP model. Chapter 6 proposes an adaptive niching EDA with balance searching based on clustering analysis. Finally, Chapter 7 gives a summary for the whole thesis. The flow of this thesis is depicted in Fig. 1.4.

This thesis summarizes the research on the predictions of gene function and protein structure. Thesis outlines and main contributions are listed as follows.

Chapter 2 proposes a composite kernel based SVM (ck-SVM) model for solving nonlinear classification tasks in gene function prediction. The proposed ck-SVM method realizes a nonlinear separating boundary by estimating a series of piecewise local linear boundaries. A local linear
multi-SVM model is estimated by a composite kernel with incorporating prior knowledge of training data. Firstly, according to the distribution information of training data, a partitioning approach composed of separating boundary detection and clustering technique is used to obtain local subsets from training data. Secondly, a composite kernel is introduced to realize a local linear multi-SVM model by incorporating prior knowledge mined from each training subsets. Instead of building multiple local SVM models separately, the prior knowledge of local subsets is used to construct a composite kernel, then the local linear multi-SVM model is realized exactly in the same way as a single SVM model by using the composite kernel.

The main contributions related to this composite kernel based SVM model are that:

- A novel composite kernel based SVM is proposed to realize a local linear multiple SVM system. Instead of building multiple local SVM models separately, the local subsets is used to estimate a composite kernel, and the composite kernel is utilized to realize the
local linear multi-SVM model by implementing the structural risk minimization in the same way as a single standard SVM.

- A partitioning approach composed of separating boundary detection and clustering technique is used to obtain local subsets from training data. By considering sample label changes in neighbor samples, the separating boundary detection can select the training samples which near to separating boundary. It can avoid trivial partitioning produced by implementing unsupervised clustering approach on original training data.

**Chapter 3** proposes an improved multi-label classification based on label ranking and delicate decision boundary SVM. Firstly, an improved probabilistic SVM with delicate decision boundary is used as the scoring method to obtain a proper label rank. It can improve the probabilistic label rank by introducing the information of overlapped training samples into learning procedure. Secondly, an instance-dependent thresholding of relating with input instance and label rank is proposed to decide the classification results. It can estimate an appropriate threshold for each testing instance according to the characteristics of instance and label rank.

The main contributions related to this multi-label classification method are that:

- A novel scoring method considering the information of overlapped training samples is proposed to solve the multi-label classification problem. The range of overlapping sample space is identified according to the relative information of two separating surfaces firstly. Then, a bias model with delicate decision boundaries is used to improve the classification accuracy of overlapping samples.

- An instance-dependent thresholding of relating with input instance and label rank is proposed to decide the classification results. By applying $d$-folds validation on training data, a set of target thresholds for all training samples is determined as teachers, then the instance-dependent thresholds for testing instances are obtained by implementing KNN strategy on this teacher threshold set.

**Chapter 4** proposes an improved hierarchical multi-label classification method based on over-sampling and hierarchy constraint for solving the gene function prediction problem. The hierarchical multi-label classification task is transferred into a series of binary SVM classification tasks. Then, two measures are implemented to improve the classifier performance by introducing the hierarchy constraint into learning procedures. Firstly, for imbalanced functional classes, a hierarchical SMOTE is proposed as over-sampling preprocessing to improve
the SVM learning performance. Secondly, an improved True Path Rule (TPR) consistency approach is introduced to ensemble the results of binary probabilistic SVM classifications. It can improve the classification results and guarantee the hierarchy constraint of classes.

The main contributions related to this hierarchical multi-label classification method are that:

- For imbalance classes in gene function prediction datasets, a hierarchical SMOTE approach is proposed to preprocess the imbalanced training subsets for classifiers. It can improve the performance of classifiers by changing the distribution of the imbalanced datasets.

- An improved TPR consistency approach is used to combine the results of binary probabilistic SVM classifiers. A performance weight is introduced into TPR method, it can restrict the poor performance binary classifiers bring an error propagation effect to other classifiers.

Chapter 5 proposes a novel hybrid Estimation of Distribution Algorithm (EDA) to solve the protein structure prediction problem on HP model. Firstly, a composite fitness function containing the information of folding structure core (H-Core) is introduced to replace the traditional fitness function of HP model. The proposed fitness function is expected to select better individuals for probabilistic model of EDA. Secondly, local search with guided operators is utilized to refine found solutions for improving efficiency of EDA. Thirdly, an improved backtracking-based repairing method is proposed to repair invalid individuals sampled by the probabilistic model of EDA. It can significantly reduce the number of backtracking searching operation and the computational cost for long sequence protein.

The main contributions related to this hybrid EDA are that:

- A composite fitness function containing the information of folding structure core (H-Core) is introduced to replace the traditional fitness function of HP model. The proposed fitness function is expected to select better individuals for probabilistic model of EDAs algorithm. It can help to increase the chance of finding the global optimum and reduce the complexity of EDA (population size and the number of generation needed).

- Local search with guided operators is utilized to refine the found solutions for improving efficiency of EDA. The local information of solutions found so far can be helpful for exploitation, while the global information can guide the search for exploring promising areas.
• An improved backtracking-based repairing method is introduced to repair invalid individuals sampled by the probabilistic model of EDAs for the long sequence protein instances. The traditional backtracking repairing procedure will produce heavy computational cost for searching invalid closed-areas of folding structure. To avoid entering invalid closed-areas, a detection procedure for feasibility is introduced when selecting directions for the residues in backtracking searching procedure. It can significantly reduce the number of backtracking searching operation and the computational cost for the long protein sequences.

Chapter 6 proposes an adaptive niching EDA based on Affinity Propagation (AP) clustering analysis. The AP clustering is used to adaptively partition the niches and mine the searching information from the evolution process. The obtained information is successfully utilized to improve the EDA performance by using a balance niching searching strategy. The proposed adaptive niching EDA is used for solving the protein HP model folding. In addition, three benchmark functional multimodal optimization problems based on continuous EDA with single Gaussian probabilistic model are also used to evaluate the proposed method in continuous situation.

The main contributions related to this adaptive niching EDA are that:

• An adaptive niching EDA is proposed, AP clustering is used to adaptively partition the population into niches during a run of EDA. The individuals are clustered before submitting them to niching clearing. A cluster can be seen as a niche, and the niche number and the niche radiuses may vary obviously for different generations.

• A mechanism of niche capacity selection based on the Boltzmann scheme is proposed to realize a balance searching between exploration and exploitation.

Chapter 7 concludes this work, summarizes the thesis and gives suggestions for further research.
Chapter 2

Support Vector Machine with Composite Kernel for Classification

2.1 Introduction

Support Vector Machines (SVMs) have been widely used in different application areas and become the state of the art. Different from most of conventional methods, SVM implements the Structural Risk Minimization (SRM) principle which seeks to minimize an upper bound of the generalization error rather than minimize the training error [113]. This eventually results in remarkable characteristics such as the good generalization performance, the absence of local minima and the sparse representation of solution.

By mapping the input data onto a higher-dimensional feature space in a non-linear fashion and seeking an optimal separating hyperplane in the feature space, SVMs can deal with linearly inseparable classification problems, and the feature mapping can be done implicitly through the kernel trick. Nonlinear SVMs employ sophisticated kernel functions, such as radial basis function (RBF), polynomial functions, etc., to fit datasets with complex decision surfaces. However, as many other nonlinear classification methods, nonlinear kernel SVM models also face the potential overfitting issue when the number of training examples is small due to their large VC dimensions.

In functional genomics, there are some classification tasks (such as FunCat Yeast datasets [10] and Genbase motif-based datasets [37]), the nonlinear (RBF-kernel or other non-linear kernels) SVM models can not obtain proper classification results. These tasks are nonlinear problems with characteristics of high noise and large number of input features compared with the relatively small
number of training examples, the conventional nonlinear kernel SVM models are severely overfitting at times [10]. Composite kernel techniques are always used to deal with some special classification tasks [92, 69, 16, 62, 108], prior knowledge of training data can be introduced into SVM classifier by composite kernel techniques. In this chapter, a new multi-SVM system called composite kernel based SVM (ck-SVM) is proposed for solving those gene function classification tasks.

A nonlinear separating boundary can be approximately seen as an aggregation of piecewise local linear boundaries. The proposed ck-SVM method realizes a nonlinear separating boundary by estimating a series of piecewise local linear boundaries. A local linear multi-SVM system is estimated by a composite kernel with incorporating prior knowledge of training data. 1) According to the distribution information of training data, a partitioning approach composed of separating boundary detection and clustering technique is used to obtain local subsets from training data. By considering sample label changes in neighbor samples, the separating boundary detection can select the training samples which are near to separating boundary. The selected samples are partitioned into local subsets by Affinity Propagation (AP) clustering, and each subset is utilized to capture prior knowledge of corresponding local linear boundary. 2) A composite kernel is introduced to realize a local linear multi-SVM system by incorporating prior knowledge mined from each training subsets. Instead of building multiple local SVM models separately, the prior knowledge of local subsets is used to construct a composite kernel, then the local linear multi-SVM system is realized exactly in the same way as a single SVM model by using the composite kernel. Experimental results on benchmark datasets demonstrate that the proposed method improves the classification performance efficiently.

The proposed method falls in the category of multiple linear SVM based on subspace pattern recognition. In some previous works, Ref. [25] presented Localized Support Vector Machine which builds multiple linear SVM models from training data and each model is designed to classify a particular test example. Ref. [46] introduced Mixtures of Linear SVMs by packaging linear SVMs into a probabilistic formulation and embedding them in the mixture of experts model. Different from those methods, in the proposed ck-SVM method, first, the subsets for capturing local linear knowledge are only clustered from samples which are near to separating boundaries. It can avoid trivial partitioning produced by implementing unsupervised clustering approach on original training data. Second, a composite kernel incorporating prior knowledge is introduced to realize the local linear multi-SVM system by estimating a series of piecewise linear boundaries. Instead of building multiple local SVM models separately, the partition information of local subsets is used to build a composite kernel, and the composite kernel is utilized to realize the local linear multi-SVM system.
2.2 A Composite Kernel Based SVM (ck-SVM) Model

2.2.1 Formulations of Model

In this chapter, the binary classification problem is considered. The classifier is built from a given labeled training dataset of \( N \) samples

\[
(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_N, y_N)
\]

(2.2.1)

where \( x_i \in \mathbb{R}^d \) is the input vector corresponding to the \( i \)-th sample labeled by \( y_i \in \{-1, +1\} \) depending on its class.

As showed in Fig. 2.1, a non-linear separating boundary can be seen as an aggregation of \( M \) piecewise local linear boundaries \( \Omega_j^T x + b_j, j = 1, \ldots, M \). According to the piecewise linear approximation method, the piecewise linear model \( f_P(x) \) can be written compactly as follow.

\[
f_P(x) = \sum_{j=1}^{M} (\Omega_j^T x + b_j)R_j(x) + b
\]

(2.2.2)
where $\mathcal{R}_j(x)$’s are the basis function, $\Omega_j$’s are the coordinate parameter vectors of local linear boundaries. The role of basis function is similar to that of a functional space basis. In some particular situations, they do constitute a functional basis. Typical examples are wavelet basis and RBF basis. The overall performance of the the piecewise linear model is obtained via an interpolation using the basis function $\mathcal{R}(x)$. It also implies that the piecewise linear model $f_P(x)$ can describe any sufficiently smooth nonlinear separating hyperplane function on a compact interval arbitrarily well by merely increasing the value of $M$.

Two parameter vectors $\Phi(x)$ and $\Theta$ are defined as follows.

$$
\Phi(x) = [\mathcal{R}_1(x), x\mathcal{R}_1(x), \ldots, \mathcal{R}_M(x), x\mathcal{R}_M(x)]^T
$$

$$
\Theta = [b_1, \Omega_1^T, \ldots, b_M, \Omega_M^T]^T
$$

(2.2.3)

Introducing parameter vectors $\Phi(x)$ and $\Theta$ (Eq. 2.2.3) into Eq. 2.2.2, the piecewise linear model $f_P(x)$ can be rewritten as:

$$
f_P(x) = \Theta^T \Phi(x) + b
$$

(2.2.4)

By introducing the Structural Risk Minimization principle [33], the piecewise linear model $f_P(x)$ (Eq. 2.2.4) can be written as the QP optimization problem as

$$
\min_{\Theta, b, \xi} J_P = \frac{1}{2} \Theta^T \Theta + c \sum_{k=1}^{N} \xi_k
$$

s.t. \quad \begin{cases}
    y_k[\Theta^T \Phi(x_k) + b] \geq 1 - \xi_k, & k = 1, \ldots, N \\
    \xi_k \geq 0, & k = 1, \ldots, N
\end{cases}

(2.2.5)

The Lagrangian is constructed:

$$
\mathcal{L}(\Theta, b, \xi; \alpha, v) = J_P(\Theta, \xi) - \sum_{k=1}^{N} (\alpha_k y_k[\Theta^T \Phi(x_k) + b] - 1 + \xi_k) - \sum_{k=1}^{N} v_k \xi_k
$$

(2.2.6)

with Lagrange multipliers $\alpha_k \geq 0$, $v_k \geq 0$ for $k = 1, \ldots, N$. The solution is given by the saddle point of the Lagrangian:

$$
\max_{\alpha, v} \min_{\Theta, b, \xi} \mathcal{L}(\Theta, b, \xi; \alpha, v)
$$

(2.2.7)

This leads to

$$
\begin{align*}
\frac{\partial \mathcal{L}}{\partial \Theta} &= 0 \rightarrow \Theta = \sum_{k=1}^{N} \alpha_k y_k \Phi(x_k) \\
\frac{\partial \mathcal{L}}{\partial b} &= 0 \rightarrow \sum_{k=1}^{N} \alpha_k y_k = 0 \\
\frac{\partial \mathcal{L}}{\partial \xi_k} &= 0 \rightarrow 0 < \alpha_k < c, \quad k = 1, \ldots, N
\end{align*}
$$

(2.2.8)
The dual problem becomes
\[
\max_{\alpha} \mathcal{J}_D(\alpha) = -\frac{1}{2} \sum_{k,l=1}^{N} y_k y_l K(x_k, x_l) \alpha_k \alpha_l + \sum_{k=1}^{N} \alpha_k
\]
\[
\text{s.t.} \quad \begin{cases}
\sum_{k=1}^{N} \alpha_k y_k = 0 \\
0 \leq \alpha_k \leq c, \ k = 1, \ldots, N
\end{cases}
\]  
(2.2.9)

where \(K(x_k, x_l)\) is a composite kernel defined by
\[
K(x_k, x_l) = \Phi(x_k)^T \Phi(x_l) = (1 + x_k^T x_l) \sum_{j=1}^{M} R_j(x_k) R_j(x_l)
\]
(2.2.10)

for \(k = 1, \ldots, N\). Hence, the piecewise linear model \(f_P(x)\) (Eq. 2.2.4) is reduced to a standard SVM based on a composite kernel. Finally the nonlinear SVM classifier takes the form
\[
y = \text{sign}\left[\sum_{k=1}^{N} \alpha_k y_k K(x, x_k) + b\right]
\]
(2.2.11)

with \(\alpha_k\) positive real constants which are the solution to a QP problem.

From the optimal solution, it is not necessary to know the parameter vector \(\Phi(x)\) in expansion (Eq. 2.2.3) explicitly, as the solution only depends on the inner product which defines composite kernel \(K(x_k, x_i)\) (Eq. 2.2.10). In order to represent an inner product, the kernel is required to satisfy Mercer’s condition [113]. In Eq. 2.2.10, factors \(R_j(x_k) R_j(x_l)\) are outputs of basis functions. It follows that the kernel \(K(x, x_i)\) satisfies Mercer’s condition because it consists of a sum of products of Mercer kernels, cf. [113].

### 2.2.2 Properties of Model

According to the definitions mentioned in the previous subsection, the proposed ck-SVM model can be seen as a novel local linear multi-SVM system that is skillfully realized by a composite kernel. The prior knowledge of each local linear boundaries are introduced into model by basis function parameters of the composite kernel, and the local linear multi-SVM system is realized exactly in the same way as a single SVM model by using the composite kernel.

When the parameter \(M\) in Eq. 2.2.2 is set as 1 specifically, the ck-SVM model becomes a linear SVM. By increasing the value of \(M\), the model can describe any sufficiently smooth nonlinear separating hyperplane function. Therefore, the ck-SVM is a flexibility controllable support vector
machine model, it can solve nonlinear tasks by introducing multiple local linear boundaries and avoid over-fitting in certain degree by selecting a proper parameter $M$.

In conventional multi-SVM methods, the training data is partitioned into several subsets, and each subset is used to estimate a local SVM model separately, then outputs of local SVM models are combined by a certain ensemble mechanism. However, the local SVM models, estimated by local information of subsets, are always not suitable for describing whole nonlinear separating boundary because data noise and other reasons. And designing an effective ensemble mechanism is a difficult task in practice.

In the proposed model, instead of building multiple local SVM models separately, the partition information of local subsets is used to construct a composite kernel, and the composite kernel is utilized to realize the local linear multi-SVM system by implementing the structural risk minimization in the same way as a single standard SVM. Comparing with conventional multi-SVM methods, the proposed model can properly estimate nonlinear separating boundary by learning from global information of whole training data. In addition, by implementing the structural risk minimization as a single SVM and introducing the interpolation based on basis function, the ck-SVM can overcome some drawbacks of ensemble mechanism in conventional methods.

2.3 Estimation of the Proposed ck-SVM Model

2.3.1 Affinity Propagation Clustering

Affinity Propagation (AP) [45] is a recently introduced clustering method which takes as input a set of measures of similarity between pairs of data points and outputs a set of clusters of the points with their corresponding exemplars. The algorithm takes a matrix of similarity measures between each pair of points $s(i, k)$ as input. Instead of requiring that the number of clusters be predetermined, the AP takes as input a real number $s(k, k)$ for each data point $k$. These values, which are called preferences, are a measure of how likely each point is to be chosen as exemplar.

The algorithm works by exchanging messages between the points until a stop condition is satisfied. There are two types of messages to be exchanged between data points. The responsibility $r(i, k)$, sent from data point $i$ to candidate exemplar point $k$, reflects the accumulated evidence for how well-suited point $k$ is to serve as the exemplar for point $i$, taking into account other potential exemplars for point $i$. The availability $a(i, k)$, sent from candidate exemplar point $k$ to point $i$, reflects the accumulated evidence for how appropriate it would be for point $i$ to choose point $k$ as its exemplar, taking into account the support from other points that point $k$ should be an exemplar.
The availabilities are initialized to zero: \( a(i, k) = 0 \). Then, the parameters are computed and updated using the rules as follows:

\[
\begin{align*}
    r(i, k) &\leftarrow s(i, k) - \max_{k' \text{ s.t. } k' \neq k} \{a(i, k') + s(i, k')\} \quad (2.3.1) \\
a(i, k) &\leftarrow \min\{0, r(k, k) + \sum_{i' \text{ s.t. } i' \notin \{i, k\}} \max\{0, r(i', k)\}\} \quad (2.3.2) \\
a(k, k) &\leftarrow \sum_{i' \text{ s.t. } i' \neq k} \max\{0, r(i', k)\} \quad (2.3.3)
\end{align*}
\]

The AP method has been praised because of its ability to efficiently and quickly handle very large problems.

### 2.3.2 Partitioning Based on Separating Boundary Detection

In order to estimate parameters of the proposed ck-SVM model according to the distribution information of training data, clustering techniques can be used to partition training data into subsets. But our partitioning task is somewhat different from conventional unsupervised clustering. In order to capture the prior knowledge of local linear boundaries properly, the partitioning approach in the ck-SVM should ensure each subset contains training examples from two classes simultaneously. But conventional clustering techniques consider only the proximity between examples and often end up grouping training examples from the same class into the same cluster, because such clusters tend to be pure.

A partitioning approach composed of separating boundary detection and clustering technique is proposed to partition training data into subsets. A separating boundary detection considering sample label changes in neighbor area is used to select samples which are near to the separating boundary firstly. Then, the AP clustering is used to partition the selected samples into subsets for estimating parameters of the ck-SVM model.

An example on synthetic data is used to formulate the proposed partitioning method. As showed in Fig. 2.2(a), a two-class dataset is constructed comprising 339 points in each class, the points are uniformly drawn from some shifted cosine signals in the two-dimensional space and perturbed with Gaussian noise in the vertical direction. Implementing unsupervised clustering technique onto the original data may produce trivial partitioning. The clustering results of implementing AP clustering (the cluster number is set as 3 empirically) onto the synthetic data (678 samples) is showed in Fig. 2.2(b), where samples from different clusters are plotted in different colors. It can be found that the
Figure 2.2: An example of partitioning approach. (a) the synthetic data (678 samples) constructed by shifted cosine signals with Gaussian noise; (b) the clustering results of implementing AP clustering onto the original synthetic data; (c) the selected samples (219 samples) by implementing the separation boundary detection; (d) the clustering results of implementing AP clustering onto the selected samples.

Partitioning tends to grouping samples from the same class into the same cluster. The partitioned subsets cannot be used to capture distribution information of local separating boundaries properly.

To solve above problem, in the proposed partitioning method, a separating boundary detection is introduced to select samples before implementing clustering. Samples around the boundary are selected by detecting changes of class label. The detection procedure is described as follows. For a certain training data sample $P$, consider the $n$ nearest neighbor samples, if one of its $n$ neighbor samples has a different class label, then $P$ is selected as a sample around the separating boundary to be reserved. The result of detection on the training data may lead to a subset of samples that distribute around the separation boundary between classes. It can filter out information that may be regarded as less relevant, while preserving the important structural properties near to the separation.
boundary. The selected result (219 samples) from the synthetic data by implementing the detection is showed in Fig. 2.2(c) and the clustering results on the selected samples is showed in Fig. 2.2(d). It can be found that the subsets of the proposed partitioning is able to capture distribution information in the training data properly.

### 2.3.3 Estimation of the Proposed ck-SVM Model

According to the definition of the proposed ck-SVM described in previous, basis functions is used to interpolate the piecewise linear hyperplanes, and each basis function corresponds to a local linear hyperplane. Therefore, the basis functions should abstract the distribution information of dataset as accurate as possible. In this chapter, a RBF gaussian function is selected as the basis function to capture distribution information of obtained partitioning subsets.

\[
R(x) = e^{-\frac{(x-\mu)^2}{2\sigma^2}}
\]  

(2.3.4)

where \(\mu\) is the subset center, \(\sigma\) the radius of subset and \(\lambda\) the scale parameter.

Another important parameter in the proposed model is the number of basis function \(M\). It is decided by the number of partitioning subsets (clusters). Theoretically, the proposed model \(f_P(x)\) (Eq. 2.2.10) can describe any sufficiently smooth nonlinear separating hyperplane function on a compact interval arbitrarily well by merely increasing the value of \(M\). If a proper parameter \(M\) is selected according with the data distribution (for example, select \(M = 3\) for the cosine synthetic data showed in Fig. 2.2(a)), the proposed model will get a well performance. In practice, the AP clustering used in the proposed model is an adaptive clustering approach, and it also can work by user-specified number of clusters.

### 2.4 Experiments and Results

#### 2.4.1 Experiment Data Sets

In functional genomics, an important problem is predicting the functions of genes (proteins). Currently, the number of protein sequences being stored in central protein databases from labs all over the world is constantly increasing. From these proteins only a fraction has been experimentally analyzed in order to detect their structure and hence their function in the corresponding organism. The reason is that experimental determination of structure is labor-intensive and quite time-consuming. Therefore there is the need for automated tools that can classify new proteins to structural families.
Table 2.1: Properties of experiment datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Attribute</th>
<th>Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence (seq)</td>
<td>478</td>
<td>2580</td>
<td>1339</td>
</tr>
<tr>
<td>All microarray (expr)</td>
<td>551</td>
<td>2488</td>
<td>1291</td>
</tr>
</tbody>
</table>

Table 2.2: Descriptions of classes (labels).

<table>
<thead>
<tr>
<th>FunCat ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.01</td>
<td>Protein folding and stabilization</td>
</tr>
<tr>
<td>14.04</td>
<td>Protein targeting, sorting and translocation</td>
</tr>
<tr>
<td>14.07</td>
<td>Protein modification</td>
</tr>
<tr>
<td>14.10</td>
<td>Assembly of protein complexes</td>
</tr>
<tr>
<td>14.13</td>
<td>Protein/peptide degradation</td>
</tr>
</tbody>
</table>

Generally, gene function predictions always are tasks of multi-label classification [40] or hierarchical multi-label classification [10], and SVMs is often used as efficient basis classifier because of their good generalization ability [40, 115, 10]. The purpose of this chapter is to improve the basis classifier performance for each label (class), so only per-class (binary classification) experiments are implemented for evaluating the proposed method. Two yeast datasets (“Sequence” and “All microarray”) from FunCat are used to evaluate the proposed method. The different datasets describe different aspects of the genes in the yeast genome, and the different sources of data highlight different aspects of gene function. The properties of experiment datasets, including instance number $D$ and attribute number, are listed in Tab. 2.1, the detailed description of each dataset can be referred from Ref. [29, 114]. The datasets are downloaded from the following webpage: http://dtai.cs.kuleuven.be/clus/hmcdatasets/. Five classes (labels) in “protein fate” FunCat class (FunCat $ID = 14$) are selected to implement per-class (binary classification) experiments, the descriptions of labels are listed in Tab. 2.2.

2.4.2 Evaluation Metrics

Three classical evaluation metrics of Precision, Recall and F-score are used to evaluate the efficiency of the proposed method. The three metrics are traditionally defined for a binary classification task with positive and negative classes. Precision is the proportion of positive predictions that are correct.
and recall is the proportion of positive samples that are correctly predicted positive. That is:

\[ \begin{align*}
\text{Precision} & = \frac{TP}{TP + FP} \\
\text{Recall} & = \frac{TP}{TP + FN} \\
F - score & = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\end{align*} \tag{2.4.1} \]

with TP the number of true positives (correctly predicted positive samples), FP the number of false positives (positive predictions that are incorrect), and FN the number of false negatives (positive samples that are incorrectly predicted negative). Note that these measures ignore the number of correctly predicted negative samples because only positive samples are concerned in gene function prediction.

### 2.4.3 Experiment Setting and Results

In our experiments, the LSSVM [107] is taken as a basis. SVM parameters are chosen by cross-validation procedure. In partitioning approach, the neighbor parameter \( n \) is set as 5 for separating boundary detection. About the parameters of AP clustering (defined in Ref. [45]), the maximum number of iterations is set as 1000, and early terminate parameter is set as 100 (i.e., the clustering procedure will be terminated if the estimated exemplars stay fixed for continuous 100 iterations); the damping factor, which may be needed if oscillations occur, is set as 0.9; Euclidean distance is used for the similarity metric of samples.

The RBF-kernel SVM and linear SVM are used to compare with the proposed ck-SVM method. The experimental results for five labels on two datasets are presented in the Tab. 2.3 and Tab. 2.4. It can be found that the RBF-kernel SVMs are severely overfitting, most of testing instances are predicted as negative and values of the recall metrics are very low. The experimental results of all evaluation metrics demonstrate that the proposed method improves the classification performance efficiently.

### 2.5 Conclusions

In this chapter, a composite kernel based SVM method with incorporating prior knowledge is presented. The proposed ck-SVM approximates a nonlinear separating boundary by estimating a series of piecewise linear boundaries. A partitioning approach composed of separating boundary detection and clustering technique is used to obtain local subsets from the training data. The separating
Table 2.3: Results of comparing with the RBF kernel and linear SVM for FunCat “Sequence” dataset.

<table>
<thead>
<tr>
<th>FunCat ID</th>
<th>RBF-kernel SVM</th>
<th>Linear SVM</th>
<th>Proposed ck-SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.01</td>
<td>0.3963</td>
<td>0.0472</td>
<td>0.0843</td>
</tr>
<tr>
<td>14.04</td>
<td>0.5326</td>
<td>0.0825</td>
<td>0.1429</td>
</tr>
<tr>
<td>14.07</td>
<td>0.7253</td>
<td>0.0623</td>
<td>0.1147</td>
</tr>
<tr>
<td>14.10</td>
<td>0.4039</td>
<td>0.0772</td>
<td>0.1296</td>
</tr>
<tr>
<td>14.13</td>
<td>0.4532</td>
<td>0.0451</td>
<td>0.0820</td>
</tr>
</tbody>
</table>

Table 2.4: Results of comparing with the RBF kernel and linear SVM for FunCat “All microarray” dataset.

<table>
<thead>
<tr>
<th>FunCat ID</th>
<th>RBF-kernel SVM</th>
<th>Linear SVM</th>
<th>Proposed ck-SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.01</td>
<td>0.4092</td>
<td>0.1175</td>
<td>0.1826</td>
</tr>
<tr>
<td>14.04</td>
<td>0.4813</td>
<td>0.1281</td>
<td>0.2023</td>
</tr>
<tr>
<td>14.07</td>
<td>0.6947</td>
<td>0.1275</td>
<td>0.2155</td>
</tr>
<tr>
<td>14.10</td>
<td>0.4187</td>
<td>0.0984</td>
<td>0.1594</td>
</tr>
<tr>
<td>14.13</td>
<td>0.4382</td>
<td>0.0946</td>
<td>0.1556</td>
</tr>
</tbody>
</table>

boundary detection can select the training samples which are near to separating boundary by considering sample label changes in neighbor samples. The selected samples are partitioned into local subsets by AP clustering, and each subset is utilized to capture prior knowledge of corresponding local linear boundary. Then, a composite kernel is introduced to realize the local linear model by incorporating prior knowledge mined from each training subsets.

The experimental results demonstrate that the proposed method can solve the special nonlinear classification tasks in gene function prediction, which conventional nonlinear kernel SVMs can not work efficiently.
Chapter 3

Multi-label Classification Based on Label Ranking and Delicate Boundary SVM for Functional Genomics

3.1 Introduction

Multi-label classification problem is an extension of traditional multi-class classification problem in which its classes are not mutually exclusive and each sample may belong to several classes simultaneously. It is increasingly required by many real-world applications. For instance, in text categorization, a document generally has several different topics, such as social, sport and health [76, 99]; in bioinformatics, each gene may be associated with a set of functional classes, such as metabolism, transcription and protein synthesis [40]; in scene classification, each scene image may belong to several semantic classes, such as beach and city [15]. In all these cases, instances in the training set are associated with a set of labels and the task is to predict the label set for the unseen instances. The main challenge of this problem is that classes are usually overlapped and correlated. Generally, traditional multi-class learning algorithms cannot work with multi-label problem effectively.

Label ranking strategy is often used in multi-label classification methods [40, 110, 117]. This strategy requires a real-valued score for each class (label) to order a label rank, the labels of higher score values are more related with the new instance, and then classifies a new instance into the classes that ranking score above a threshold. In a ranking based classification, both the scoring method and the threshold selection can influence the classification results significantly. An ideal ranking based classifier can be imagined as follows, the scoring method can provide a proper label rank (i.e., the labels of higher score values are more related with the instance). And the threshold selection strategy can estimate an appropriate threshold according to the characteristics of instance
and label ranking strategy.

About the scoring methods, since Support Vector Machine (SVM) based methods have good generalization ability in single-label multi-class problem[101, 24], more attention also has been paid on such kind of techniques for multi-label problem. At present, there are mainly two types of SVM-based methods to solve the multi-label problem. One is to consider all samples and their labels simultaneously to construct one optimization formulation, for example, rank-SVM [40] and maximal margin labeling algorithm [63]. But it is time-consuming to solve such large scale optimization problems in numerical computation. The other is to decompose a multi-label problem into many binary class sub-problems and to solve them by using SVM-like methods. Generally speaking, the latter runs faster than the former does.

Nowadays the widely used decomposition techniques include one-versus-rest and one-versus-one strategies. In one-versus-rest SVM methods [40, 110], the multi-label training set is simply divided into \( l \) (the number of labels) binary class subsets, where each positive class consists of those samples with some given label and the other remained samples belong to the corresponding negative class; and then \( l \) traditional binary SVM classifiers are trained. The weakness of this strategy is that binary classifiers are trained independently, ignoring the semantic relationship of labels.

For one-versus-one SVM methods, the first step is to split the training set into \( l \times (l-1)/2 \) binary class subsets. It is noted that in some subsets samples possess double labels (overlapping samples), i.e. belonging to both positive and negative classes at the same time. Some papers show that the performance of classifier can be improved by introducing the information of those double-label training samples into learning procedure effectively. In [102], those double-label samples are processed by the method of triple class support vector machine. In [115], those double-label samples are considered as the third class besides positive and negative classes and so-called parallel support vector machine is designed to cope with such training subsets. The drawback of this strategy is that many binary classifiers need to be built for those datasets with large label number \( l \).

Different from the above methods, in this chapter, a novel scoring method is proposed to solve the multi-label classification problem. Experientially, the basic Binary-SVM method cannot work very well for those overlapping samples because of the complexity of multi-label problem. But there is an important common characteristic of overlapping (double-label) samples in the essentially overlapping classes of two labels. It is that the majority of double-label samples are distributed near to two binary SVM separating surfaces simultaneously. The motivation of the proposed method is that, the range of overlapping sample space is identified according to the relative information of two separating surfaces firstly. Then, a bias model with delicate decision boundaries is used to improve
the classification accuracy of overlapping samples.

The scoring methods are the major focus of research in the multi-label classification. However, the threshold selection is often processed as a unimportant post-processing step and so far few studies have done on this subject. For the document classification applications, in [118] and [70], threshold selection techniques are successfully applied to obtain better performance. A per-class threshold selection strategy has been used to improve the performance of the one-versus-all binary Support Vector Machine (SVM) multi-label classification in [42]. It estimate an appropriate threshold value for each class by optimizing macro-average F-measure.

An improved ranking based multi-label classification method is proposed in this chapter. 1) An improved probabilistic SVM with delicate decision boundary is used as the scoring method. In this method, firstly, utilizing the one-versus-rest strategy and the Platt’s sigmoid method [90], $l$ binary probabilistic SVM classifiers are trained for $l$ labels correspondingly. For example, training the $p$-th ($p \in [1, l]$) binary SVM, the samples labeled by $p$ only are considered as “positive”, all others are considered as “negative”. Secondly, according to the one-versus-one strategy and $l$ binary probabilistic SVMs, $l \ast (l - 1)/2$ bias models are built by corresponding each pair of labels. The proposed bias model can capture the characteristics of double-label samples and give the more delicate decision boundaries for samples of overlapping sample space. It is expected to obtain more proper probabilistic classifier results by introducing the information of double-label training samples into classifier. Finally, biased probabilistic classification results are ranked to produce label ranking. 2) An instance-dependent thresholding of relating with input instance and label rank is proposed to decide the classification results. By applying $d$-folds validation on training data, a set of target thresholds for all training samples is determined as teachers, then the instance-dependent thresholds for testing instances are obtained by implementing KNN strategy on this teacher threshold set. Experimental results on the two Functional Genomics benchmark datasets of Yeast and Genbase show that the proposed method improves the classification performance efficiently, compared with binary SVM method and some existing well-known methods.

The rest parts of the chapter are organized as follows. Section 3.2 gives a brief overview of probabilistic outputs of SVM classification and ranking based multi-label classification. Section 3.3 describes the details of the proposed method based on label ranking and delicate decision boundary SVM. Section 3.4 formulates evaluation metrics of multi-label classification problem and the results of experimental performance evaluation. Finally, the conclusions and future work directions are discussed.
3.2 Probabilistic Outputs of SVM Classification and Ranking Based Multi-label Classification

3.2.1 Probabilistic Outputs for SVMs

SVM is a supervised learning method introduced by Vapnik based on his Statistical Learning Theory and Structural Minimization Principle [113]. The basic idea of using SVM for classification is to find the optimal separating hyperplane between the positive and negative samples. The optimal hyperplane is defined as the one giving the maximum margin between the training samples that are closest to it. Support vectors are the samples that lie closest to the separating hyperplane. Once this hyperplane is found, new samples can be classified simply by determining on which side of the hyperplane they fall.

Given training data, \( T = \{(x_1, y_1), \ldots, (x_m, y_m)\} \), where \( x_i \in \mathbb{R}^n \) is the input pattern, and \( y_i \in \{-1, 1\} \) is the class label for a two-class problem, SVM for classification attempts to find a classifier \( f(X) \), which minimizes the expected misclassification rate. A linear classifier \( f(X) \) in SVM is equivalent to solving a convex quadratic optimization problem bellow:

\[
\min \frac{1}{2} \|W\|^2 + C \sum_{i=1}^{n} \xi_i \quad (3.2.1)
\]

subject to \( y_i (\langle W, X_i \rangle + b) \geq 1 - \xi_i \) and \( \xi_i \geq 0 \), where \( C \) is called the regularization parameter, and is used to balance the complexity of classifier and the classification accuracy on the training set \( T \). This quadratic problem is generally solved through its dual formulation [33]. Simply replacing the involved vector inner-product with a non-linear kernel function converts linear SVM into a more flexible non-linear SVM, which is the essence of the famous kernel trick [116, 19].

Constructing a classifier to produce a posterior probability \( p(\text{class} | \text{input}) \) is very useful in practical recognition situations. Posterior probabilities are also required when a classifier is making a small part of an overall decision, and the classification output must be combined for the overall decision. However, the standard SVM do not provide such probabilities. The Platt’s sigmoid method through training the parameters of an additional sigmoid function to map the SVM outputs into probabilities [90]. The following gives a brief description of this method of probabilistic output for SVM in two classes’ case.

Given training data \( T = \{(x_1, y_1), \ldots, (x_m, y_m)\} \), labeled by \( y_i \in \{-1, 1\} \), the binary SVM can obtain the decision function \( f(x) \) so that sign of \( f(x) \) is the prediction of all the test samples \( x \). Instead of predicting the label, many applications require a posterior class probability \( p(y = 1 | x) \).
In [90], the probability \( p(y = 1|x) \) is approximated by a sigmoid function
\[
p(y = 1|x) = \frac{1}{1 + \exp(Af(x) + B)}
\] (3.2.2)
with parameters \( A \) and \( B \). To estimate the best values of \((A, B)\), any subset of \( m \) training data can be used to solve the following maximum likelihood problem,
\[
\min_{Z=(A,B)} \left\{ -\sum_{i=1}^{l} (t_i \log(p_i) + (1 - t_i) \log(1 - p_i)) \right\}
\]
where \( p_i = \frac{1}{1 + \exp(Af_i + B)} \), \( f_i = f(x_i) \)
\[
t_i = \begin{cases} 
\frac{N_+ + 1}{N_+ + 2} & \text{if } y_i = 1 \\
\frac{N_- + 2}{N_- + 2} & \text{if } y_i = -1 
\end{cases} \quad i = 1, 2, \ldots, m
\] (3.2.3)
where \( N_+ \) means the number of positive-labeled samples, and \( N_- \) means the number of negative-labeled samples.

The method leaves the SVM error function unchanged. Instead, it adds a trainable post-processing step which is trained with regularized binomial maximum likelihood. A Sigmoid function with two parameters is chosen as the post-processing, since it matches the posterior that is empirically observed. The SVM+Sigmoid combination preserves the sparseness of the SVM while producing probabilities that are of comparable quality to the regularized likelihood kernel method.

### 3.2.2 Ranking Based Multi-label Classification

A formal problem statement for multi-label classification is described as follow. The domain of instances is denoted as \( X = IR^d \) and the finite set of labels as \( Y = \{1, 2, \ldots, l\} \). Given a multi-label training set \( T \) of size \( m \), \( T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\} \) \((x_i \in X, y_i \subseteq Y)\) where each instance \( x_i \in X \) is associated with a subset of relevant labels \( y_i \subseteq Y \), the goal of the multi-label classification is to construct a multi-label classifier \( C : X \rightarrow 2^l(l = |Y| \geq 2) \) which, for any given instance \( x_i \), determines all its relevant label subset \( y_i \).

For the ranking based multi-label classification, the learning system is expected to produce a real-valued scoring function of the form \( f(\cdot, \cdot) : X \times Y \rightarrow IR \). It is supposed that, a successful learning system will tend to output larger score values for labels in \( y_i \) than those not in \( y_i \). The corresponding multi-label classifier \( C(\cdot) \) can also be derived from the real-valued scoring function:
\[
C(x_i) = \{y | f(x_i, y) > t, y \in Y\},
\] (3.2.4)
where \( t \) is a value estimated by a certain threshold selection strategy.
3.2.3 Threshold Selection Strategies

There are three commonly used threshold selection strategies for classification [118], including the rank-based method (RCut), the proportion-based assignments (PCut) and the score-based local optimization (SCut). Let \( l \) denotes the number of classes (labels) in the problem, \( d \) denotes the number of instances in the testing (or validation) set, and assume one score is produced by the classifier for each instance-class pair. The threshold selection algorithms are defined as follows.

1. **RCut**, it is a per-instance strategy, for each instance, sort classes by score and assign positive to each of the \( t \) top-ranking classes. The parameter \( t \) is a value (an integer between 1 and \( l \)) can be either specified by the user or automatically tuned using a validation set. That is, the value of \( t \) optimizing the global performance of the classifier on the validation set is fixed when applying the classifier on new instance in the testing set. RCut with \( t = 1 \) is also commonly used for single-label classification.

2. **PCut**, it is a per-class strategy, for each class \( c_j \), sort the testing instance by score and assign positive to each of the \( l_j \) top-ranking instances, where \( l_j = P(c_j) \times x \times d \) is the number of instance assigned to class \( c_j \), and \( P(c_j) \) is the prior probability (estimated using a training set) for an arbitrary instance to be a member of class \( c_j \). The parameter \( x \) is automatically tuned in the same fashion as tuning \( t \) for RCut, by varying the value of \( x \) until the global performance of the classifier is optimized on the validation set.

3. **SCut**, it is also a per-class strategy, score a validation set of documents for each class and tune the threshold over the local pool of score values until the optimal performance of the classifier is obtained for that class. And the per-class thresholds are fixed when applying the classifier to new instance in the testing set.

The three strategies have different properties of information used, suitability, flexibility and overfitting risk. And the choice of threshold selection strategy depends on the characteristics of classifier and application [118].

3.3 Proposed Ranking Based Multi-label Classification

According to the previous analysis, the ranking based multi-label classification consists of two important parts: the scoring method and the threshold selection. And both the scoring and the threshold selection can influence the classification results significantly. The scoring method should
provide a proper label rank, i.e., the labels of high score values are more related with the instance. And the threshold selection strategy can choose appropriate thresholds for the testing instances.

In the improved ranking based multi-label classification, as showed in Fig. 3.1, instead of estimating a static threshold for all testing instances in conventional methods, threshold selection is considered as a function \( T(x_i, R(x_i)) \). The threshold function relies on both the input instance \( x_i \) and the label rank \( R(x_i) \). It is expected to choose appropriate thresholds according to the characteristics of instances and label ranking strategy. The corresponding multi-label classifier \( C(\cdot) \) can be described as:

\[
C(x_i) = T(x_i, R(x_i)) \cdot R(x_i).
\] (3.3.1)

Efforts on both parts are made to improve the classification performance. For the scoring part, an improved probabilistic SVM with delicate decision boundary is used to obtain a proper label ranking results. For the threshold selection part, based on the ideal threshold set of training data estimated by the cross validation method, a KNN learning model is used to realize the threshold function \( T(x_i, R(x_i)) \) related with both the input instance and the label rank.

### 3.3.1 Scoring Method Based on SVM with Delicate Decision Boundary

The goal of the multi-label classification is to construct a multi-label classifier \( c : X \rightarrow 2^l (l = |Y| \geq 2) \) which, for any given sample \( x_i \), determines all its relevant label subset \( y_i \). In this chapter, however, instead of constructing a multi-label classifier, the learning system will produce a real-valued function of the form \( f : X \times Y \rightarrow IR \). It is supposed that, a successful learning system will tend to output larger values (probability) for labels in \( y_i \) than those not in \( y_i \). The corresponding multi-label classifier \( c(\cdot) \) can also be derived from the real-valued function: \( f(\cdot, \cdot): c(x_i) = \{y|f(x_i, y) > t(x_i), y \in Y\} \). where \( t(\cdot) \) is a threshold function.
The main scheme of the proposed method is shown in Fig. 3.2. First of all, \( l \) (the number of label) binary SVMs are built by one-versus-rest strategy, i.e., training the \( p \) (\( p \in [1, l] \)) binary SVM, the samples labeled by \( p \) only are considered as “positive”, all others are “negative”. And the Platt’s sigmoid method is used to obtain the probabilistic outputs of SVMs. Thus, \( l \) probabilistic classifiers are constructed. Then, according to the one-versus-one strategy and probabilistic outputs of \( l \) binary SVMs, \( l \ast (l - 1)/2 \) bias models are built by corresponding each pair of labels. The detailed mechanism of bias model will be described in next subsection, it can capture the characteristics of double-label samples and give a more delicate decision boundary for samples of overlapping sample space. It can be used to improve the performance of probabilistic classifier. Finally, in integration, ranking and final decision procedure, for each sample, there are \( (l - 1) \) probabilistic values for every label, the mean value of \( (l - 1) \) probabilistic values is used as the final probability for this label. Hence, a probabilistic rank of \( l \) values are obtained for all labels. In the testing procedure, a proper threshold estimated based on training set is used to determine the label subset and its size for each testing sample.

**Bias Model with Delicate Decision Boundary**

Now let us consider the basic problem of classifying the essentially overlapping classes of label \( p \) and \( q \). As shown in Fig. 3.3 (a), two separating surfaces are constructed by SVM binary classifiers mentioned in previous subsection. They divide the sample space into four areas: “only \( p \) area (1)”,...
“only q area (2)”, “p and q area (3)” and “no one’s area (4)”. It must be noted that in some cases (e.g. for linearly separable classes, or symmetric classes) one or two of these areas may be empty [88]. And two SVM separating surfaces are constructed in different high dimensional space, in order to summarize the probabilistic relativity of two surfaces, they are analyzed in same dimensional space for convenience.

In practice, there is an important common characteristic of double-label samples in the essentially overlapping classes of two labels. It is that the majority of double-label samples are distributed near to two probabilistic separating surfaces simultaneously. A real example of the Yeast benchmark dataset (label 3-rd and 4-th) is shown in Fig. 3.3 (b). Because of the error of SVMs classifiers, there are many samples (especially the double-label samples) can not be classified correctly. Most of double-label samples are distributed near to the separating surfaces and intermixed with other single-label samples.

The main idea of the proposed bias model is that, for certain pair of labels, the distributing range of overlapping sample space (specified by the four thresholds $\varepsilon_{p+}$, $\varepsilon_{p-}$, $\varepsilon_{q+}$ and $\varepsilon_{q-}$) can be estimated by double-label training samples and relative information of two probabilistic boundaries. Then, training samples in this overlapping sample space are used to estimate two delicate boundaries for two label respectively (bold curves shown in Fig. 3.3 (a)). And the delicate boundaries are used to obtain better classification results of SVMs.

The proposed bias model utilizes the probability value to realize the delicate boundaries. The probabilistic outputs of SVM can be obtained by Sigmoid method mentioned in previous section. Training the first separating surface, the samples labeled by $p$ only are considered as “positive”, all others are “negative”. Trained classifier estimates the probability $r^+_{pq}(x)$ and dual probability $r^-_{pq}(x)$:

$$
r^+_{pq}(x) = P(p \in f(x) \land q \notin f(x)|x \in p \cup q)
$$
$$
r^-_{pq}(x) = 1 - r^+_{pq}(x) = P(q \in f(x) \land p \notin f(x)|x \in p \cup q). \quad (3.3.2)
$$

The second separating surface is trained for class $q$ can estimates the probability $r^+_{qp}(x)$ and dual probability $r^-_{qp}(x)$ similarly:

$$
r^+_{qp}(x) = P(q \in f(x) \land p \notin f(x)|x \in p \cup q)
$$
$$
r^-_{qp}(x) = 1 - r^+_{qp}(x) = P(p \in f(x) \land q \notin f(x)|x \in p \cup q). \quad (3.3.3)
$$
Figure 3.3: Bias model with delicate decision boundary
And the probabilities of a given sample \( x \), which belongs to each of four areas, can be derived:

\[
\begin{align*}
P_1 &= P(x \in "only p") = r^+_p(x)r^-_{qp}(x) \\
P_2 &= P(x \in "only q") = r^-_p(x)r^+_{qp}(x) \\
P_3 &= P(x \in "p and q") = r^+_p(x)r^+_{qp}(x) \\
P_4 &= P(x \in "no one's") = r^-_p(x)r^-_{qp}(x)
\end{align*}
\]

(3.3.4)

The mechanism of bias model is described as follows. Firstly, double-label training samples near to two separating surfaces simultaneously are selected, and their SVM distance values to hyperplane of two labels (denoted as \( \{d_{pi}\} \) and \( \{d_{qi}\} \)) are used to estimate the four range thresholds \( \varepsilon_{p+}, \varepsilon_{p-}, \varepsilon_{q+} \) and \( \varepsilon_{q-} \) of two separating surfaces. Those thresholds can be used to decide the range of overlapping sample space. Secondly, according to the four thresholds, training samples in the range of overlapping sample space are selected to train two delicate boundaries for two label \( p \) and \( q \) respectively. Thirdly, in the testing procedure, the four thresholds and trained delicate boundaries of two separating surfaces are utilized to correct the classification results. For every sample in the testing dataset, if it belong to the overlapping sample space, its final probabilistic outputs are corrected by two trained delicate boundaries of two labels.

**Parameters estimation for the proposed method**

According to mechanism of the proposed bias model described in previous section, the key procedures of bias model are how to estimate the four thresholds and train the delicate boundaries of separating surfaces.

For the first problem, the double-label training samples near to two separating surfaces simultaneously are utilized to estimate them. Take the \( \varepsilon_{p-} \) as the example to introduce the estimation algorithm: select the double label training samples subset (denoted \( S_{ol} \)). For every sample \( i \in S_{ol} \), according to the Eq. 3.3.4 to estimate the probabilities \( P_1, P_2, P_3 \) and \( P_4 \), then calculate the standard deviation value \( r_{std} = \text{stdev}(P_1, P_2, P_3, P_4) \). So the threshold \( \varepsilon_{p-} \) can be calculated by the equation as follow:

\[
\varepsilon_{p-} = \text{mean}(|d_{pi}|), i \in s_{ol} \land d_{pi} < 0 \land r_{std} < \eta
\]

(3.3.5)

where \( d_{pi} \) is the sample’s distance value to label \( p \) SVM hyperplane, \( r_{std} < \eta \) means the distribution of sample simultaneously near to the two separating surfaces of label \( p \) and \( q \) in Fig. 3.3 (a), the \( \eta \) is a constant trained based on the overlapping training data.

For the second problem, training samples in the overlapping sample space decided by four range thresholds (those samples stimulatingly meet two condition of \( \varepsilon_{p-} \leq d_{pi} \leq \varepsilon_{p+} \) and \( \varepsilon_{q-} \leq d_{qi} \leq \varepsilon_{q+} \)) are used to train two delicate boundaries for label \( p \) and \( q \) respectively. Then the four thresholds and trained delicate boundaries of two separating surfaces are utilized to correct the classification results.
\( \varepsilon_{q-} \) and \( \varepsilon_{q+} \) are selected to train two curve functions \( f_p \) and \( f_q \) (delicate boundaries) for two label \( p \) and \( q \) respectively.

Take \( f_p \) as the example to introduce the curve function fitting algorithm. A statistical method and the relative information of two separating surfaces are used to fit the curve function \( f_p \). The decision values for function \( f_p \) is denoted by the distances between the training samples and separation boundary of the label \( q \). As shown in Fig. 3.3 (a), let us consider the decision value region \( [\varepsilon_{q-}, \varepsilon_{q+}] \), the length is \( L \). The whole decision value domain is divided into \( k_i \) intervals, then the distance between two neighbor intervals \( d \) can be calculated as \( d = L/k_i \). Using statistical method, the numbers of samples in overlapping sample space belonging to double-label or single-label respectively in each interval are counted. In the \( j \)-th interval, the number of samples belonging to double-label is denoted as \( Num_d^j \), belonging to single label is denoted as \( Num_s^j \), where \( j = 1, 2, ..., k_i \) is the index of interval. The bias probabilistic value is defined as following:

\[
\text{p}_{de, j} = \frac{Num_d^j}{Num_d^j + Num_s^j}.
\]  

(3.3.6)

Then the data set \{ \( (d_{qj}, p_{de, j}) \) \} is used to fit the delicate decision boundary curve function \( f_p \), \( d_{qj} \) as the input variable and \( p_{ad, j} \) as the value of function.

In the testing procedure, for every sample \( i \) of the test data, if the values of \( d_{pi} \) and \( d_{qi} \) meet two conditions of \( \varepsilon_{p-} \leq d_{pi} \leq \varepsilon_{p+} \) and \( \varepsilon_{q-} \leq d_{qi} \leq \varepsilon_{q+} \) simultaneously, the probabilities are biased as \( p_{pi} = p_{pi} + f_p(d_{qi}) \), \( p_{qi} = p_{qi} + f_q(d_{pi}) \) for the final probabilistic outputs.

### 3.3.2 Instance-dependent Thresholding Strategy

Different from the conventional methods to estimate a static threshold for all testing instances, the proposed instance-dependent thresholding strategy is considered as a function that relies on input instance and label rank. It is expected to choose appropriate thresholds according to the characteristics of instance and label ranking strategy. Based on the ideal threshold set of training data estimated by the cross validation strategy, a KNN learning model is used to realize the threshold function related with the input instance and the label rank.

As showed in Fig. 3.4, firstly, the \( d \)-folds validation method is utilized to estimate the target threshold values for training samples. The training data is split into several training/validation subsets. Training subsets are used to train the classifiers, and estimate the target threshold value for each sample in validation subset by optimizing a certain evaluation measure. Secondly, according to the teacher set of target thresholds for training samples, the KNN strategy is utilized to obtain the
instance-dependent thresholds for testing instances. The outline of the proposed instance-dependent thresholding strategy is described as follows.

1. Split training data into $d$ folds. According to the validation strategy, for fold $i = 1, \ldots, d$.
   
   (a) Train the SVM+Sigmoid multi-label classifier (mentioned in previous subsection) on the $d-1$ folds (except the validation fold $i$).
   
   (b) Based to the trained multi-label classifier, consider the fold $i$ as the testing data, obtain the probabilistic test outputs (score values for each classes) for samples in the fold $i$.
   
   (c) For each sample $I_p$ in the validation fold $i$, sort the score values of classes (labels) as a descending rank $(c^{(1)}, c^{(2)}, \ldots, c^{(l)})$, where $c^{(1)}$ is the label of maximum score and $l$ is the number of labels. Then, respectively take subset \{c^{(1)}\}, \{c^{(1)} \ c^{(2)}\}, \{c^{(1)} \ c^{(2)} \ c^{(3)}\} \ldots \ldots \ldots (from the subset contains the top label, add a new label one by one sequentially) as the candidate result label set for the sample $I_p$ and evaluate it by a certain evaluation measure. Select the best candidate label set \{c^{(1)} \ c^{(2)} \ldots c^{(j)}\}, $1 < j < l$. Then, take the mean of score values of $c^{(j)}$ and $c^{(j+1)}$ as the target threshold value of the sample $I_p$.

2. The target threshold values for all training samples are obtained by validation of Step 1. This target threshold set of training data is utilized as teacher set to estimate the thresholds of testing instances by KNN strategy described in Step 3.

3. In test procedure, for each instance $I_q$ in the testing data, the similarities with all training samples are calculated by a certain distance metric (Euclidean distance, Hamming distance or other measures). The mean value of $k$ nearest samples’ thresholds is assigned as the threshold of the testing instance $I_q$, where $k$ is the parameter of KNN strategy.
The proposed threshold selection is a per-instance strategy similar with the RCut method. But different from RCut method, instead of using a static parameter $t$ as the threshold for all testing instances, the proposed method estimates threshold for different testing instance by a lazy learning method of KNN according to the ideal threshold values of training data. For the $d$-folds validation method in the proposed thresholding strategy, the parameter $d$ should be chosen by considering the trade-off between the performance and computational cost.

### 3.4 Experiments

#### 3.4.1 Benchmark Datasets

Nowadays, the number of protein sequences being stored in central protein databases from labs all over the world is constantly increasing. From these proteins only a fraction has been experimentally analyzed in order to detect their structure and hence their function in the corresponding organism. The reason is that experimental determination of structure is labor-intensive and quite time-consuming. Therefore there is the need for automated tools that can classify new proteins to structural families. With the contribution of modern data analysis techniques, such as machine learning and knowledge discovery, the issue has been approached computationally, thus providing fast and more flexible solutions.

The proposed improved multi-label classification method is used to solve two benchmark biology functional prediction problems of Yeast functional genomics [40] and Genbase motif-based protein classification [37].

**Yeast functional genomics data.** The gene functional classes of the yeast Saccharomyces cerevisiae, which is one of the best-studied organisms. Specifically, the yeast data set studied in [40] is investigated. Each gene is described by the concatenation of microarray expression data and phylogenetic profile and is associated with a set of functional classes whose maximum size can be potentially more than 190. In order to make it easier, Elisseeff and Weston preprocessed the data set where only the known structure of the functional classes is used. Actually, the whole set of functional classes is structured into hierarchies up to four levels deep. In this chapter, the same data set as used in [40] is adopted. In this data set, only functional classes in the top hierarchy are considered.

**Genbase motif-based protein classification data.** This problem is studied in [37]. Any protein chain can be mapped into a representation based on attributes. Such a representation supports the efficient function of data-driven algorithms, which represent instances as classified part of a fixed set
of attributes. A very important issue in the data mining process is the efficient choice of attributes. In this problem, protein chains are represented using a proper motif sequence vocabulary. Suppose the vocabulary contains \( N \) motifs. Any given protein sequence typically contains a few of these motifs. Each sequence is encoded as an \( N \)-bit binary pattern where the \( i \)-th bit is 1 if the corresponding motif is present in the sequence; otherwise the corresponding bit is 0. Each \( N \)-bit sequence is associated with a label that identifies the functional family of the sequence (if known).

Two benchmark datasets are downloaded from website of [110]. The detailed information about them, such as the number of samples, attributes, classes and their average number of labels or classes are listed in Tab. 3.1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Total labels</th>
<th>Total features</th>
<th>Avg. labels</th>
<th>Training/Testing set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast</td>
<td>14</td>
<td>103 (Numeric)</td>
<td>4.25</td>
<td>1500/917</td>
</tr>
<tr>
<td>Genbase</td>
<td>27</td>
<td>1186 (Discrete)</td>
<td>1.35</td>
<td>463/199</td>
</tr>
</tbody>
</table>

### 3.4.2 Evaluation metrics for multi-label classification

Multi-label classification requires different metrics than those used in traditional single-label classification. Let \( D \) be a multi-label evaluation data set, consisting of \(|D|\) multi-label examples \((x_i, y_i), i = 1 \ldots |D|, y_i \subseteq Y\). \( Y = \{1, 2, \ldots, l\} \) is the set of labels. Let \( C \) be a multi-label classifier and \( z_i = C(x_i) \) be the set of labels predicted by \( C \) for example \( x_i \). The following evaluation metrics for label ranking used in [99, 40, 110, 120] are used in this chapter:

**Hamming loss** evaluates how many times a sample label pair is misclassified, i.e., a label not belonging to the instance is predicted or a label belonging to the instance is not predicted, which is defined as:

\[
\text{HammingLoss} = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{1}{l} |y_i \Delta z_i| \tag{3.4.1}
\]

where \( \Delta \) means for the \( XOR \) operation of two sets. Hamming loss evaluates how many times a sample label pair is misclassified, i.e., a label not belonging to the instance is predicted or a label belonging to the instance is not predicted. In \( E1 \), \( \Delta \) means the \( XOR \) operation of two sets. The smaller this measure is, the better the method performs.

Multi-label Precision (MPrec), Multi-label Recall (MRec) and Multi-label F-score (MF-score)
evaluation metrics are defined as follows.

\[
MPrec = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|y_i \cap z_i|}{|z_i|}
\]

\[
MRec = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|y_i \cap z_i|}{|y_i|}
\]

\[
MF-score = \frac{2 \times MPrec \times MRec}{MPrec + MRec}
\] (3.4.2)

There is a common characteristic for above three measures, the larger measure value, the better the classification performance. And the performance is perfect when its value equal 1.

### 3.4.3 Experimental Setting and Results

In our experiments, the LSSVM [107] is taken as a basis. SVM parameters are chosen by cross-validation procedure. In the training procedure, the methods mentioned in Section 3.3 are utilized to train the SVM classifiers, calculate the parameters of sigmoid functions and correction models and estimate the ideal threshold values for all training samples. In the testing procedure, for each test instance, a probabilistic rank of \( l \) values for all labels is obtained by final decision procedure. Then, according to the ideal threshold values of training samples, a proper threshold estimated based on KNN method. And this proper threshold is used to determine the label subset finally.

For the threshold selection, the \( MF-score \) evaluation metric is selected as the optimizing measure for estimating the ideal threshold set of training data. In addition, there are two important parameters for the proposed threshold selection. The first is the parameter \( d \) of \( d \)-fold cross validation, considering the trade-off between the performance and computational consume, 10-folds strategy is used for Yeast dataset and 5-folds for Genbase dataset. The second is the parameter \( k \) of KNN method, by experimentalism, \( k = 4 \) is chosen for Yeast dataset and \( k = 3 \) for Genbase dataset. The Fig. 3.5 shows the relation of the parameter \( k \) and the performance of classification for Yeast dataset.

The following popular multi-label classification methods: C4.5, Naive Bayes and Binary-SVM [99, 110] are utilized to compare with the proposed method. The experiment results for two datasets are presented in the Tab. 3.2 and Tab. 3.3. For Genbase dataset, the nonlinear (RBF-kernel or other non-linear kernels) SVM models are severely overfitting. Therefore, the linear SVM and ck-SVM (proposed in Chapter 2) are used for Genbase dataset in our experiments. The results of the methods of C4.5 and Naive Bayes are cited from [110]. The winning results are marked with bold font.
Figure 3.5: Relation between the parameter $k$ of KNN in threshold selection and the performance of classification for Yeast dataset

Table 3.2: Experimental results for Yeast dataset.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Hamming loss</th>
<th>MRec</th>
<th>MRec</th>
<th>MF-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5</td>
<td>0.259</td>
<td>0.561</td>
<td>0.593</td>
<td>0.5766</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>0.301</td>
<td>0.610</td>
<td>0.531</td>
<td>0.5678</td>
</tr>
<tr>
<td>Binary-SVM (based on RBF kernel)</td>
<td>0.2021</td>
<td>0.5862</td>
<td>0.6332</td>
<td>0.6088</td>
</tr>
<tr>
<td>Proposed (based on RBF kernel)</td>
<td><strong>0.1915</strong></td>
<td><strong>0.6761</strong></td>
<td><strong>0.7106</strong></td>
<td><strong>0.6929</strong></td>
</tr>
</tbody>
</table>

Table 3.3: Experimental results for Genbase dataset.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Hamming loss</th>
<th>MRec</th>
<th>MRec</th>
<th>MF-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5</td>
<td>0.001</td>
<td>0.992</td>
<td>0.995</td>
<td>0.9935</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>0.035</td>
<td>0.273</td>
<td>0.276</td>
<td>0.2745</td>
</tr>
<tr>
<td>Binary-SVM (Based on linear SVM)</td>
<td>0.0011</td>
<td>0.9933</td>
<td><strong>0.9958</strong></td>
<td>0.9945</td>
</tr>
<tr>
<td>Proposed (based on linear SVM)</td>
<td>0.00006</td>
<td>0.9947</td>
<td>0.9950</td>
<td>0.9949</td>
</tr>
<tr>
<td>Proposed (based on ck-SVM)</td>
<td><strong>0.00006</strong></td>
<td><strong>0.9968</strong></td>
<td>0.9942</td>
<td><strong>0.9955</strong></td>
</tr>
</tbody>
</table>
Figure 3.6 shows the MF-score results of the thirteen bias models for 12-th label in Yeast dataset comparing with the Binary-SVM method.

![Figure 3.6: Comparing results of the bias models for 12-th label in Yeast dataset](image)

The experimental results of all evaluation metrics demonstrate that the proposed method improves the performance of classification efficiently.

### 3.5 Conclusions

In a ranking based multi-label classification, both the scoring method and the threshold selection can influence the classification results significantly. In this chapter, an improved ranking based multi-label classification is proposed to solve functional genomics applications. Firstly, the improved probabilistic SVM with correction model is used as the scoring method. It can improve the probabilistic label ranking by introducing the information of multi-label training samples in overlapping sample space into learning procedure. Secondly, an instance-dependent thresholding is proposed to decide the classification results. It can estimate appropriate threshold value according to the characteristics of input instance and label ranking strategy. Experimental results on the two functional genomics benchmark datasets of Yeast and Genbase show that the proposed method improves the performance of classification efficiently, compared with binary SVM method and some existing well-known methods.
Chapter 4

Hierarchical Multi-label Classification Based on Over-sampling and Hierarchy Constraint for Gene Function Prediction

4.1 Introduction

Multi-label classification problem is an extension of traditional multi-class classification problem in which the classes are not mutually exclusive and each sample may belong to several classes simultaneously. In this case, samples in training set are associated with a set of labels and the task is to predict the label set for the unseen instance. Hierarchical multi-label classification (HMC) is a special variant of multi-label classification where the classes are organized in a hierarchy: an example that belongs to one class automatically belongs to all its super-classes (this is called the hierarchy constraint).

Gene function prediction [10] is a difficult task of HMC. It is known that a single gene may have multiple functions, and these functions are organized in a hierarchy. Gene function prediction is characterized by hundreds or thousands of functional classes structured according to a predefined hierarchy; and the example distributions of functional classes are usually high degree skewed (imbalanced) with negative more than positive. There are two hierarchy structures: a directed acyclic graph for the Gene Ontology (GO) [5] and a tree forest for FunCat [95]. The classification on FunCat taxonomy of rooted tree structure is our focus in this chapter.

Many approaches have been proposed to deal with the HMC tasks [93, 94, 105, 10, 91, 100], and the approaches can be categorized as two different research lines. The first research line is to develop a single multi-label learning model that predicts all the classes of an example at once. Clare et al [29] presented a decision tree method for multi-label classification in the context of functional
genomics. A decision tree is used to predict not a single class but a vector of Boolean class variables. Vens et al. [114] proposed a method based on the predictive clustering trees for HMC tasks. The second research line is to develop an ensemble model that consisted of binary classifiers. An HMC task is transferred into a series of separate binary classification tasks, and each classifier is trained to learn one class, then results of all binary classifiers are combined by considering the hierarchical relationships between classes [9, 10, 84, 50, 18, 112]. Barutcuoglu et al. [10] presented a two-step approach where support vector machines (SVMs) are learned for each class separately, and then combined using a Bayesian network model so that the predictions are consistent with the hierarchy constraint. Valentini et al. [112] proposed a True Path Rule (TPR) ensemble method to combine the binary classifications. This method is characterized by a two-way asymmetric flow of information that traverses the graph-structured ensemble: positive predictions for a node influence in a recursive way its ancestors, while negative predictions influence its offspring.

The approaches of the first research line, which most of are based on the decision tree model with a vector of Boolean class variables, can predict all the classes by learning a single model. But for complicated HMC tasks (datasets) with a large number of attributes and classes, it is difficult and unpractical to tackle all classes efficiently by a single learning model. The approaches of the second research line, which transfer an HMC task into many binary classifications, always have advantages for solving such complicated HMC tasks with high dimension attributes. Efficient classification method (such as SVM), which is adept at high dimension learning task, can be used as binary classifier to improve the classification performance.

There are two main challenges that need to be solved for the second research line approaches. Firstly, most of binary classifiers at lower hierarchy levels often need to learn from strongly skewed training datasets, i.e., in typical HMC applications, classes at lower hierarchy levels often have very small frequencies, while (because of the hierarchy constraint) the frequency of classes at higher levels tends to be high. In such imbalance dataset learning, standard classifiers tend to be overwhelmed by the majority class and ignore the minority one. Secondly, the hierarchy constraint of classes is not taken into account in separate binary classifiers, i.e. it is not automatically imposed that an instance belonging to a class should belong to all its super-classes. Most of traditional HMC methods [10, 84, 50, 18, 112] mainly focus on the second challenge by developing ensemble strategies considering the hierarchy constraint. In our research, the two challenges are simultaneously considered: a hierarchical imbalanced dataset preprocessing approach is proposed to enhance the performance of the separate binary classifiers; and an improved ensemble strategy based on classifier performance is introduced to deal with the second challenge.
How to efficiently introduce the information of hierarchy constraint between classes into learning model is a vital problem for the HMC task. Along the second research line, this chapter proposes an improved HMC method based on over-sampling and hierarchy constraint for solving the gene function prediction problem. The HMC task is transferred into binary SVM classification tasks for classes in the hierarchy. Then, two measures are implemented to improve the HMC performance by introducing the hierarchy constraint into learning procedures. 1) A hierarchical over-sampling approach based on Synthetic Minority Over-sampling Technique (SMOTE) is proposed to preprocess the imbalanced training subsets for binary SVM classifiers. It can improve the performance of classifiers by changing the distribution of the imbalanced datasets. 2) An improved TPR consistency approach is used to combine the results of binary probabilistic SVM classifiers. The consistency ensemble can improve the classification results and guarantee the hierarchy constraint of classes (an instance belonging to a class should belong to all its super-classes).

The rest of the chapter is organized as follows. Section 4.2 gives a brief overview of the hierarchical multi-label classification, True Path Rule consistency ensemble and imbalanced datasets learning methods. Section 4.3 formulates the proposed HMC method based on over-sampling and consistency ensemble. Section 4.4 presents the experiments on FunCat benchmark datasets. Finally, the conclusions are introduced.

4.2 Hierarchical Multi-label Classification

4.2.1 Notation and Definition

In a hierarchical multi-label classification problem, an instance $x$ can be assigned to one or more classes of the set $C = \{c_1, c_2, \cdots, c_m\}$. The assignments can be coded through a vector of multi-labels $y = (y_1, y_2, \cdots, y_m) \in \{0, 1\}^m$, by which if $x$ belongs to class $c_i$, then $y_i = 1$, otherwise $y_i = 0$, where the variable $i, 1 \leq i \leq m$, refers to the indices corresponding to the $m$ classes belonging to the set $C$.

The classes of HMC are structured according to a hierarchy and can be represented by a rooted tree, where nodes correspond to classes, and paths to relationships between classes. The class hierarchy is denoted by $(C, \leq_h)$, where $\leq_h$ is a partial order in class tree representing the superclass relationship (for all $c_1, c_2 \in C, c_1 \leq_h c_2$ if and only if $c_1$ is a superclass of $c_2$). Hence the node corresponding to the class $c_i$ can be simply denoted by $i$. The set of children nodes of $c_i$ is represented by $\text{child}(i)$, and the set of its parents by $\text{par}(i)$. Moreover $y_{\text{child}(i)}$ denotes the labels of the children classes of node $c_i$ and analogously $y_{\text{par}(i)}$ denotes the labels of the parent classes of $c_i$. A classifier
$D : X \rightarrow \{0, 1\}^m$ computes the multi-label associated to each instance $x \in X$, and $d_i(x) \in \{0, 1\}$ is the label predicted by the classifier for class $c_i$. For the sake of simplicity, $d_i(x)$ is represented simply by $d_i$ if there is no ambiguity.

### 4.2.2 Imbalanced Dataset Learning

The imbalanced dataset learning problem typically occurs when, in a classification problem, there are much more instances of some classes than others. In such cases, standard classifiers tend to be overwhelmed by the large classes and ignore the small ones [21, 109]. The solutions to imbalanced dataset learning problem can be divided into data and algorithmic levels categories. The methods at data level change the distribution of the imbalanced dataset, and then the balanced dataset are provided to the learner to improve the detection rate of minority class. The methods at the algorithm level modify the existing data mining algorithms or put forward new algorithms to resolve the imbalance problem [52].

At the algorithmic level, solutions include adjusting the costs of the various classes so as to counter the class imbalance, adjusting the probabilistic estimate at the tree leaf (when working with decision trees), adjusting the decision threshold, and recognition-based (i.e., learning from one class) rather than discrimination-based (two class) learning [21].

At the data level, different forms of re-sampling methods were proposed [21]. The simplest re-sampling methods are random over-sampling and random under-sampling. The former augments the minority class by exactly duplicating the examples of the minority class, while the latter randomly takes away some examples of the majority class. However, random over-sampling may make the decision regions of the learner smaller and more specific, thus cause the learner to over-fit. Random under-sampling can reduce some useful information of the data sets. Many improved re-sampling methods are thus presented, such as heuristic re-sampling methods [65], combination of over-sampling and under-sampling methods [11], embedding re-sampling methods into data mining algorithms [51], and so on.

SMOTE (Synthetic Minority Over-sampling Technique) proposed in Ref. [20] is an efficient re-sampling method at the data level. It generates new synthetic examples along the line between the minority examples and their selected nearest neighbors. The advantage of SMOTE is that it makes the decision regions larger and less specific. SMOTE generates synthetic minority examples to over-sample the minority class. For every minority example, its $k$ (which is set to 5 as default in SMOTE) nearest neighbors of the same class are calculated, then some examples are randomly selected from them according to the over-sampling rate. After that, new synthetic examples are
generated along the line between the minority example and its selected nearest neighbors.

4.2.3 TPR Consistency Ensemble

TPR consistency ensemble method has proposed in Ref. [112] for solving the HMC problem. The True Path Rule enforces the consistency of gene function annotations in FunCat taxonomies: “If the child term describes the gene product, then all its parent terms must also apply to that gene product”. In other words, if a gene is annotated with a specific functional term (functional class), then it is annotated with all the “parent” classes, and with all its ancestors in a recursive way. The ensemble method is characterized by a two-way asymmetric flow of information that traverses the tree-structured ensemble: positive predictions for a node influence in a recursive way its ancestors, while negative predictions influence its offspring.

For a given example \( x \), considering the parents of a given node \( c_i \), a classifier that respects the TPR needs to obey the following rules:

\[
\begin{align*}
    d_i = 1 & \Rightarrow d_{\text{par}(i)} = 1 \\
    d_i = 0 & \Rightarrow d_{\text{child}(i)} = 0 
\end{align*}
\]  

(4.2.1)

The TPR ensemble algorithm puts together the predictions made at each node by local “base” classifiers to realize an ensemble that obeys the “true path rule”. Positive decisions may propagate from bottom to top across the graph: they influence the decisions of the parent nodes and of their ancestors in a recursive way, by traversing the graph towards higher level nodes/classes. Negative predictions for a given node (taking into account the local decision of its descendants) are propagated to the descendants, to preserve the consistency of the hierarchy according to the true path rule.

4.3 Improved HMC Method

4.3.1 Main Frame of the Proposed Method

Main frame of the proposed HMC method based on over-sampling and hierarchy constraint ensemble is shown in Fig. 4.1. Firstly, the training dataset is partitioned into \( m \) (the number of classes) subsets for each binary classifier, and the imbalanced subsets are preprocessed by over-sampling strategy to improve the data distribution firstly. Then, the \( m \) binary probabilistic SVM classifiers are trained to predict the classification results. Finally, the probabilistic prediction results of \( m \) binary classifiers are assembled by the consistency strategy of taking into account the hierarchical relationships between classes.
Over-sampling by Hierarchical SMOTE
Classification by \( m \) probabilistic SVMs (SVM+Sigmoid)
Consistency ensemble by TPR method based on classifier performance

\( n \)-folds validation for estimating the classifier performance

Figure 4.1: Main frame of the proposed HMC method.

The SVM+Sigmoid method is used as the basic binary probabilistic classifier. Constructing a classifier to produce a posterior probability \( ps(\text{class} | \text{input}) \) is very useful when a classifier is making a small part of an overall decision, and the classification output must be combined for the overall decision. The standard SVM do not provide such probabilities, and the Platt’s sigmoid method is commonly used to estimate the probabilistic outputs of SVM. It maps the SVM outputs into probabilities through training the parameters of an additional sigmoid function by validation set in training data [90].

\[
ps(y = 1 | x) = \frac{1}{1 + \exp(A f(x) + B)} \tag{4.3.1}
\]

with parameters \( A \) and \( B \). To estimate the best values of \((A, B)\), any validation subset of \( n' \) training data can be used to solve the following maximum likelihood problem,

\[
\min_{Z=(A,B)} \left\{ -\sum_{i=1}^{l} (t_i \log(ps_i) + (1 - t_i) \log(1 - ps_i)) \right\}
\]

\[
p_{si} = \frac{1}{1 + \exp(A f_i + B)}, \quad f_i = f(x_i)
\]

\[
t_i = \begin{cases} 
\frac{N_{+}+1}{N_{+}+2} & \text{if } y_i = 1 \\
\frac{N_{-}+1}{N_{-}+2} & \text{if } y_i = -1 
\end{cases} \quad i = 1, 2, \cdots n' \tag{4.3.2}
\]

where \( N_+ \) means the number of positive-labeled samples, and \( N_- \) means the number of negative-labeled samples. The SVM+Sigmoid combination preserves the sparseness of the SVM while producing probabilities that are of comparable quality to the regularized likelihood kernel method.
4.3.2 Proposed Hierarchical SMOTE

Preprocessing for Training Subsets

In the preprocessing procedure of partitioning the training dataset into \( m \) subsets for binary classifiers. According to the hierarchy relations between classes, the negative examples in subset for each class have been selected in such a way described as follows. The negative examples are not annotated for the class, but belong to the parent class (i.e. positive for the parent class). For the training subset of class \( c_i \), examples that belong to \( c_i \) become the positive examples, and the other examples which belong to \( \text{par}(i) \) but not to \( c_i \) become the negative examples. This negative example selection method is often used in many HMC methods [114, 112], in this way only negative examples that are not too dissimilar to the positive ones are selected as the training subsets for binary classifiers.

Even using the above mentioned partition preprocess, there are still many high degree imbalanced training subsets for binary classifiers. A hierarchical SMOTE is proposed to tackle this problem. SMOTE is an over-sampling approach in which the minority class is over-sampled by creating “synthetic” examples. The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the \( k \) minority class nearest neighbors.

According to the hierarchy constraint characteristic of HMC problem, for the hierarchical class tree, the positive examples belong to training subset of class (node) \( c_i \) automatically belong to all
training subset of its super-classes (nodes) \( \text{par}(i) \). Therefore, in the proposed hierarchical SMOTE method, over-sampling procedures for \( m \) subsets are implemented from bottom to top across the class tree, and the “synthetic” examples created by children classes are automatically reserved and regarded as the “synthetic” examples of their parent classes. For example, as showed in Fig. 4.2, considering the node 2, all “synthetic” examples \( \{e3 \cup e4\} \) created by its children nodes \( \text{child}(2) = \{3, 4\} \) are combined and reserved as its “synthetic” examples firstly, the over-sampling procedure of node 2 only need create the insufficient “synthetic” examples. Detailed procedures of the proposed hierarchical SMOTE are described in next subsection.

Comparing with implementing the basic SMOTE over-sampling for each subsets respectively, the proposed hierarchical SMOTE method has two advantages. Firstly, because of the “synthetic” examples created by children classes are automatically reserved for their parent classes, the proposed method can remarkably reduce the over-sampling operations of “synthetic” examples for whole HMC problem. Secondly, the basic SMOTE method has a drawback in some special situations [2], it can not obtain good performance because of its assumptions about the training set. For instance, the space between two positive instances is assumed to be positive and the neighborhood of a positive instance is also assumed to be positive, but those assumptions may not always be true for some special distribution datasets. In the proposed hierarchical SMOTE method, considering the “synthetic” examples for class \( c_i \), those reserved “synthetic” examples from children classes \( \text{child}(i) \) are all distributed in the positive space of class \( c_i \) definitely, therefore, the proposed method can solve the drawback of basic SMOTE in certain degree.

### Hierarchical SMOTE for Imbalanced Subsets

According to the rooted tree of \( m \) labels, from bottom to top across the tree, implement over-sampling processing for subsets of imbalanced classes. For node \( c_i \) \((1 \leq i \leq m)\) in the class tree, denote the positive (minority) class is \( P \) and the negative (majority) class is \( N \) as follows,

\[
P = \{p_1, p_2, \cdots, p_{pn}\}, \quad N = \{n_1, n_2, \cdots, n_{nn}\}
\]

where \( pn \) and \( nn \) are the number of positive and negative examples in training subset of label \( c_i \). The class \( c_i \) is considered as imbalanced if \( \frac{pn}{nn} \leq \eta \), \( \eta (\eta = 0.6 \text{ is chosen in experiments}) \) is a skewed parameter to evaluate the degree of imbalance. The hierarchical SMOTE method is described as follows.

Step 1, calculate the number of “synthetic” examples need be over-sampled \( O_p = nn \ast \mu - pn \), where \( \mu (\mu = 0.8 \text{ is chosen in experiments}) \) is a cost sensitive parameter of SMOTE, it means the
ratio of positive and negative examples after over-sampling.

Step 2, if $c_i$ is a leaf node in class tree, generate $O_p$ “synthetic” positive examples from the data $P$ as follows. Calculate the parameter vector $S = \{s_1, s_2, \ldots, s_{pm}\}$ according to an average strategy, where $s_j$ is an integer and $O_p = \sum_{j=1}^{pm} s_j$. For each $p_j \in P$, $s_j$ nearest neighbors are randomly selected from its $k$ ($k = \max(S) + 5$ is chosen in experiments) nearest neighbors in $P$.

Firstly, the differences $\text{dif}_q(q = 1, 2, \ldots, s_j)$ between example $p_j$ and its $s_j$ nearest neighbors from $P$ are calculated, then multiply $\text{dif}_q$ by a random number $r_q(q = 1, 2, \ldots, s_j)$ between 0 and 1, finally, $s_j$ new “synthetic” positive examples are generated between $p_j$ and its nearest neighbors:

$$\text{synthetic}_q = p_j + r_q \times \text{dif}_q, q = 1, 2, \ldots, s_j.$$ (4.3.4)

The above procedure is repeated for each $p_j$ in $P$, and $O_p$ “synthetic” positive examples are generated for node $c_i$. This step is similar with SMOTE, more details can be seen in Ref. [20].

Step 3, if $c_i$ is not a leaf node in class tree, combine all “synthetic” examples created by children nodes $\text{child}(i)$ as set $P_{\text{child}}$. If $|P_{\text{child}}| \geq O_p$, randomly select $O_p$ examples from the set $P_{\text{child}}$ as “synthetic” positive examples set of node $c_i$. If $|P_{\text{child}}| < O_p$, generate $O_p - |P_{\text{child}}|$ “synthetic” positive examples for node $c_i$ using similar procedure Step 2, and combine the generated examples and the examples of $P_{\text{child}}$ as “synthetic” positive examples of node $c_i$.

### 4.3.3 Improved TPR Consistency Ensemble

#### Disadvantage of the basic TPR Ensemble

The TPR ensemble method [112] combines the local predictions of the base learners associated to each node with the positive decisions that come from the bottom of the hierarchy, and with the negative decisions that spring from the higher level nodes. More precisely, base classifiers estimate local probabilities $\overline{p_{hi}}(x)$ that a given example $x$ belongs to class $c_i$, the TPR ensemble provides an estimate of the “consensus” global probability $p_{hi}(x)$. The set $\varphi_i(x)$ denotes the children of node $c_i$ which have a positive prediction for a given example $x$:

$$\varphi_i(x) = \{j| j \in \text{child}(i), d_j(x) = 1\}$$ (4.3.5)

The global consensus probability $p_{hi}(x)$ of the ensemble depends both on the local prediction $\overline{p_{hi}}(x)$ and on the prediction of the nodes belonging to $\varphi_i(x)$. As described in Eq. 4.3.6, the decision $d_i(x)$ at node/class $c_i$ is set to 1 if $p_{hi}(x) > t$, and to 0 otherwise (a natural choice for $t$ is 0.5). For the leaf nodes, the Eq. 4.3.6 becomes $p_{hi}(x) = \overline{p_{hi}}(x)$. In this way positive predictions propagate from bottom to top. On the contrary if for a given node $d_i(x) = 0$, then this decision
is propagated to its subtree. To balance the local predictions with the positive predictions coming from the ensemble, it introduces a parent weight \( w, 0 \leq w \leq 1 \), such that if \( w = 1 \) the decision at node \( c_i \) depends only by the local predictor, otherwise the prediction is shared proportionally to \( w \) and \( 1 - w \) between respectively the local parent predictor and the set of its children.

\[
ph_i(x) = w \cdot ph_i(x) + \frac{1 - w}{|\varphi_i(x)|} \sum_{j \in \varphi_i(x)} ph_j(x).
\] (4.3.6)

There is a disadvantage in basic TPR ensemble method. In practice, the performances of different binary SVM classifiers for hierarchy classes are obviously varied for real-world datasets. For some classes at lower level of hierarchy rooted tree, the classifier performance is very poor because of data noise or other reasons. But in the basic TPR consistency ensemble method, as described in Eq. 4.3.6, each classifier in the positive prediction children set \( \varphi_i(x) \) contributes same propagation effect for estimating the local probabilities \( \overline{ph}_i(x) \). Because the propagation effects from children classes are represented by probabilistic values, this equal propagation mechanism may produce an error propagation: if an error prediction occurs in a poor performance child node classifier, i.e., this child classifier predicts a high probabilistic value for a negative instance that not belongs to it. This error prediction will give an obvious effect to other node classifiers because of its high probabilistic value.

**Improved TPR Ensemble Based on Classifier Performance**

To restrict the poor performance binary classifiers bring an error propagation effect to other classifiers. In the improved consistency ensemble method, another weight \( \nu_j \) of classifier performance is introduced to the TPR ensemble as follows:

\[
ph_i(x) = w \cdot ph_i(x) + \frac{1 - w}{|\varphi_i(x)|} \sum_{j \in \varphi_i(x)} \nu_j \cdot ph_j(x)
\]

\[
\nu_j = \frac{E_j}{\sum_{k \in \varphi_i(x)} E_k}, j \in \varphi_i(x)
\] (4.3.7)

where \( E_j \) is the performance evaluation of child node classifier \( j \in \varphi_i(x) \) by a certain metric.

The introduced performance weight \( \nu_j \) are decided by a certain performance metric of classifiers, this classifier performance metric can be estimated by a validation strategy according to the train data. It should be noted that the meaning of two weight \( w \) and \( \nu_j \) are different. The parent weight \( w \) is used to balance the local predictions and the positive predictions coming from the children nodes, Ref. [112] indicates that this parent weight is a global parameter that affect the general
precision/recall characteristics of the ensemble. The performance weight $\nu_j$ in the improved method is used to restrict the poor performance binary classifiers bring an error propagation effect to other classifiers.

Similar to the negative prediction propagation method in Ref. [112], negative predictions for a given node are propagated to its descendants, to preserve the consistency of the hierarchy. In case of a negative decision for a node $c_i$, all the classes belonging to the subtree rooted at $c_i$ are set to negative, and their probabilities are set to $p_{h_i}(x)$ if larger than $p_{h_i}(x)$.

4.4 Experiments

4.4.1 Evaluation Metrics

Considering the multi-label and hierarchy characteristics of HMC problem, two categories of metrics are adopted to evaluate the proposed method.

The first metric category is the classical Precision (Prec), Recall (Rec) and F-score for each class of the hierarchy. F-score metric is used to estimate the classifier performance parameters in the improved TPR ensemble method (Eq. 4.3.7). The three metrics are traditionally defined for a binary classification task with positive and negative classes. Precision is the proportion of positive predictions that are correct, and recall is the proportion of positive examples that are correctly predicted positive. That is:

$$\text{Prec} = \frac{TP}{TP + FP}, \quad \text{Rec} = \frac{TP}{TP + FN}$$

$$F - \text{score} = \frac{2 \times \text{Prec} \times \text{Rec}}{\text{Prec} + \text{Rec}}$$ (4.4.1)

with TP the number of true positives (correctly predicted positive examples), FP the number of false positives (positive predictions that are incorrect), and FN the number of false negatives (positive examples that are incorrectly predicted negative). Note that these measures ignore the number of correctly predicted negative examples.

Moreover, in order to take into account the multi-label characteristics of HMC problem, the second metric category, which includes Multi-label Precision (MPrec), Multi-label Recall (MRec) and Multi-label F-score (MF-score) [40, 110], is also used to evaluate the proposed method. Let $D$ be a multi-label evaluation data set, consisting of $|D|$ multi-label examples $(x_i, y_i), i = 1 \ldots |D|, y_i \subseteq Y$. $Y = \{1, 2, \ldots, k\}$ is the set of labels. Let $z_i$ be the set of labels predicted by HMC classifier for
The three metrics are described as follows:

\[
MP_{\text{rec}} = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|y_i \cap z_i|}{|z_i|}
\]

\[
M_{\text{rec}} = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|y_i \cap z_i|}{|y_i|}
\]

\[
MF - \text{score} = \frac{2 \times MP_{\text{rec}} \times M_{\text{rec}}}{MP_{\text{rec}} + M_{\text{rec}}}
\] (4.4.2)

There is a common characteristic for above-mentioned metrics, the larger measure value, the better the classification performance. The performance is perfect when its value equals 1.

### 4.4.2 Data Sets and Experiment Setting

Four yeast data sets from FunCat are used to evaluate the proposed method. The different data sets describe different aspects of the genes in the yeast genome, and the different sources of data highlight different aspects of gene function. The properties of experiment datasets, including instance number \(D\) and attribute number, are listed in Tab. 4.1, the detailed description of each dataset can be referred from Ref. [29, 114]. The four datasets are downloaded from the following webpage: http://dtai.cs.kuleuven.be/clus/hmcdatasets/.

Two different ensemble strategies are used to compare with the proposed method. The first is Flat ensemble, that does not take into account the hierarchical structure of the data. The second is basic TPR hierarchical ensemble, the parent weight \(w\) in Eq. 4.3.6 is set as 0.5. For each ensemble strategy, linear LSSVM method [107] is taken as base learner (the nonlinear (RBF-kernel or other non-linear kernels) SVM models are severely overfitting for FunCat datasets), the Platt’s sigmoid method is utilized to estimate the probabilistic outputs of SVM, and the threshold \(t\) of all binary classifier output are set as 0.5 in all the experiments. About the parameters of the proposed method, the hierarchical SMOTE parameter setting are described in Subsection 3.2.2; in the improved TPR
ensemble, 5-folds validation strategy on training data is used to estimate the F-score metric as the performance weight $\nu_j$.

4.4.3 Experiment Results and Analysis

To observe the detailed performance of the proposed hierarchical SMOTE and the improved TPR ensemble, firstly, some specific experiments are implemented based on the dataset of Sequence ($D_1$) and the subtree of “protein fate” FunCat class (FunCat ID = 14). The subtree is composed by 20 nodes (as showed in Fig. 4.3). In the dataset of Sequence ($D_1$), there are 625 training examples and 337 testing instances relating with the class subtree of ID = 14. For each class, the number of positive and negative training examples are listed in Tab. 4.2, it can be found that most of classes are significantly imbalanced.

In order to evaluate the performance of the proposed hierarchical SMOTE over-sampling approach, three experiment strategies including Flat ensemble, basic SMOTE over-sampling and the proposed hierarchical SMOTE over-sampling are implemented and evaluated. Experiment results about per-class metrics (Prec, Rec) for 20 classes (nodes) are listed in Tab. 4.2. We can find that two over-sampling strategies can effectively improve the performances of classifiers for imbalanced datasets, and the proposed hierarchical SMOTE over-sampling approach outperforms the basic SMOTE for all classes. The basic SMOTE assumes the space between two positive instances is to be positive and the neighborhood of a positive instance is also to be positive, but those assumptions may not always be true for some special distribution datasets. The proposed hierarchical SMOTE can overcome this drawback in certain degree. According to the strategies of the proposed hierarchical SMOTE, most of the reserved “synthetic” samples from children classes child($i$) are all distributed in the positive space of class $c_i$ definitely.

To observe the improved TPR ensemble approach, two experiment strategies including basic TPR ensemble and the improved method with classifier performance weight are implemented and evaluated. As listed in Tab. 4.3, the experiment results about per-class metrics (Prec, Rec) for 20 classes (nodes) show that the improved TPR outperforms the basic TPR ensemble method. The classifier performance weight in the proposed method can effectively restrict the error propagation effects from poor performance binary classifiers.

To evaluate the performances of combination methods based on different over-sampling and ensemble strategies, four combination strategies including M1 (Flat ensemble), M2 (basic TPR ensemble), M3 (the proposed hierarchical SMOTE over-sampling and Flat ensemble) and the proposed HMC (the proposed hierarchical SMOTE over-sampling and the improved TPR ensemble)
Table 4.2: Experiment results of different over-sampling strategies based on the FunCat subtree of “Protein fate” (ID=14).

<table>
<thead>
<tr>
<th>FunCat ID</th>
<th>Positive</th>
<th>Negative</th>
<th>Flat Prec</th>
<th>Flat Rec</th>
<th>SMOTE Prec</th>
<th>SMOTE Rec</th>
<th>H-SMOTE Prec</th>
<th>H-SMOTE Rec</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.01</td>
<td>61</td>
<td>564</td>
<td>0.1667</td>
<td>0.3333</td>
<td>0.1721</td>
<td>0.3784</td>
<td>0.2407</td>
<td>0.4415</td>
</tr>
<tr>
<td>14.04</td>
<td>158</td>
<td>467</td>
<td>0.3131</td>
<td>0.3690</td>
<td>0.3371</td>
<td>0.3814</td>
<td>0.3558</td>
<td>0.4405</td>
</tr>
<tr>
<td>14.07</td>
<td>306</td>
<td>319</td>
<td>0.6460</td>
<td>0.5474</td>
<td>0.6460</td>
<td>0.5474</td>
<td>0.6460</td>
<td>0.5474</td>
</tr>
<tr>
<td>14.07.01</td>
<td>14</td>
<td>292</td>
<td>0.1020</td>
<td>0.4167</td>
<td>0.2141</td>
<td>0.4072</td>
<td>0.2111</td>
<td>0.4833</td>
</tr>
<tr>
<td>14.07.02</td>
<td>41</td>
<td>265</td>
<td>0.0822</td>
<td>0.2500</td>
<td>0.0978</td>
<td>0.2571</td>
<td>0.1486</td>
<td>0.4583</td>
</tr>
<tr>
<td>14.07.02.01</td>
<td>11</td>
<td>30</td>
<td>0.0714</td>
<td>0.7500</td>
<td>0.1521</td>
<td>0.4860</td>
<td>0.0882</td>
<td>0.7500</td>
</tr>
<tr>
<td>14.07.02.02</td>
<td>27</td>
<td>14</td>
<td>0.0606</td>
<td>0.8571</td>
<td>0.0606</td>
<td>0.8571</td>
<td>0.0606</td>
<td>0.8571</td>
</tr>
<tr>
<td>14.07.03</td>
<td>91</td>
<td>215</td>
<td>0.2619</td>
<td>0.6226</td>
<td>0.2987</td>
<td>0.6233</td>
<td>0.2736</td>
<td>0.6604</td>
</tr>
<tr>
<td>14.07.04</td>
<td>32</td>
<td>274</td>
<td>0.1019</td>
<td>0.5500</td>
<td>0.0925</td>
<td>0.3672</td>
<td>0.1400</td>
<td>0.7000</td>
</tr>
<tr>
<td>14.07.05</td>
<td>44</td>
<td>262</td>
<td>0.0532</td>
<td>0.2174</td>
<td>0.0872</td>
<td>0.2556</td>
<td>0.0860</td>
<td>0.3478</td>
</tr>
<tr>
<td>14.07.07</td>
<td>11</td>
<td>295</td>
<td>0.0121</td>
<td>0.1453</td>
<td>0.0195</td>
<td>0.1670</td>
<td>0.0263</td>
<td>0.3333</td>
</tr>
<tr>
<td>14.07.09</td>
<td>6</td>
<td>300</td>
<td>0.0153</td>
<td>0.1475</td>
<td>0.0205</td>
<td>0.1892</td>
<td>0.0385</td>
<td>0.2500</td>
</tr>
<tr>
<td>14.07.11</td>
<td>49</td>
<td>257</td>
<td>0.0690</td>
<td>0.2857</td>
<td>0.0944</td>
<td>0.3034</td>
<td>0.0909</td>
<td>0.5000</td>
</tr>
<tr>
<td>14.07.11.01</td>
<td>13</td>
<td>36</td>
<td>0.0244</td>
<td>0.2500</td>
<td>0.0215</td>
<td>0.1931</td>
<td>0.0357</td>
<td>0.3750</td>
</tr>
<tr>
<td>14.10</td>
<td>118</td>
<td>507</td>
<td>0.2474</td>
<td>0.4068</td>
<td>0.2850</td>
<td>0.4768</td>
<td>0.2700</td>
<td>0.4576</td>
</tr>
<tr>
<td>14.13</td>
<td>150</td>
<td>475</td>
<td>0.3048</td>
<td>0.4507</td>
<td>0.2715</td>
<td>0.5272</td>
<td>0.3380</td>
<td>0.5070</td>
</tr>
<tr>
<td>14.13.01</td>
<td>112</td>
<td>38</td>
<td>0.1688</td>
<td>0.7407</td>
<td>0.1275</td>
<td>0.4248</td>
<td>0.1688</td>
<td>0.7407</td>
</tr>
<tr>
<td>14.13.01.01</td>
<td>75</td>
<td>37</td>
<td>0.1147</td>
<td>0.7143</td>
<td>0.1147</td>
<td>0.7143</td>
<td>0.1147</td>
<td>0.7143</td>
</tr>
<tr>
<td>14.13.04</td>
<td>16</td>
<td>134</td>
<td>0.0333</td>
<td>0.1429</td>
<td>0.0314</td>
<td>0.1762</td>
<td>0.0556</td>
<td>0.2857</td>
</tr>
<tr>
<td>14.13.04.02</td>
<td>6</td>
<td>16</td>
<td>0.0127</td>
<td>0.2542</td>
<td>0.0127</td>
<td>0.2542</td>
<td>0.0127</td>
<td>0.2542</td>
</tr>
</tbody>
</table>

are implemented and evaluated. Experiment results about multi-label metrics (MPrec, MRec and MF-score) for the whole subtree dataset are showed in Fig.4.4. The results show that the proposed HMC method outperforms other three strategies. The combination of two proposed strategies (the hierarchical SMOTE over-sampling and the improved TPR with classifier performance weight) can effectively enhance the HMC performance.

Finally, the three ensemble strategies experiment results on four datasets are listed in Tab. 4.4. The results of the proposed HMC method based on ck-SVM (proposed in Chapter 2) are also listed. It can be found that the proposed method significantly outperforms the basic TPR ensemble method and the Flat ensemble method.
Figure 4.3: Rooted tree of the FunCat classes of ID=14 (Protein fate).

Figure 4.4: Three multi-label metrics on the FunCat subtree of ID=14 (Protein fate).
Table 4.3: Experiment results of different ensemble strategies based on the FunCat subtree of “Protein fate” (ID=14).

<table>
<thead>
<tr>
<th>FunCat ID</th>
<th>TPR Prec</th>
<th>TPR Rec</th>
<th>Improved TPR Prec</th>
<th>Improved TPR Rec</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.01</td>
<td>0.2016</td>
<td>0.3792</td>
<td>0.2578</td>
<td>0.4356</td>
</tr>
<tr>
<td>14.04</td>
<td>0.3228</td>
<td>0.3929</td>
<td>0.3521</td>
<td>0.4568</td>
</tr>
<tr>
<td>14.07</td>
<td>0.5490</td>
<td>0.8263</td>
<td>0.6705</td>
<td>0.8947</td>
</tr>
<tr>
<td>14.07.01</td>
<td>0.2000</td>
<td>0.4167</td>
<td>0.2176</td>
<td>0.5116</td>
</tr>
<tr>
<td>14.07.02</td>
<td>0.0923</td>
<td>0.2500</td>
<td>0.2275</td>
<td>0.4835</td>
</tr>
<tr>
<td>14.07.02.01</td>
<td>0.1667</td>
<td>0.5000</td>
<td>0.2483</td>
<td>0.5758</td>
</tr>
<tr>
<td>14.07.02.02</td>
<td>0.0794</td>
<td>0.3571</td>
<td>0.1829</td>
<td>0.7683</td>
</tr>
<tr>
<td>14.07.03</td>
<td>0.2857</td>
<td>0.6038</td>
<td>0.2837</td>
<td>0.6487</td>
</tr>
<tr>
<td>14.07.04</td>
<td>0.0824</td>
<td>0.3500</td>
<td>0.1285</td>
<td>0.6982</td>
</tr>
<tr>
<td>14.07.05</td>
<td>0.0685</td>
<td>0.2174</td>
<td>0.0920</td>
<td>0.4276</td>
</tr>
<tr>
<td>14.07.07</td>
<td>0.0121</td>
<td>0.1453</td>
<td>0.0303</td>
<td>0.3333</td>
</tr>
<tr>
<td>14.07.09</td>
<td>0.0153</td>
<td>0.1475</td>
<td>0.0657</td>
<td>0.3150</td>
</tr>
<tr>
<td>14.07.11</td>
<td>0.0749</td>
<td>0.2786</td>
<td>0.1267</td>
<td>0.546</td>
</tr>
<tr>
<td>14.07.11.01</td>
<td>0.0169</td>
<td>0.1436</td>
<td>0.0465</td>
<td>0.3542</td>
</tr>
<tr>
<td>14.10</td>
<td>0.2474</td>
<td>0.4068</td>
<td>0.3043</td>
<td>0.4792</td>
</tr>
<tr>
<td>14.13</td>
<td>0.2516</td>
<td>0.5634</td>
<td>0.3357</td>
<td>0.5547</td>
</tr>
<tr>
<td>14.13.01</td>
<td>0.1825</td>
<td>0.4630</td>
<td>0.1864</td>
<td>0.6932</td>
</tr>
<tr>
<td>14.13.01.01</td>
<td>0.1221</td>
<td>0.7429</td>
<td>0.1237</td>
<td>0.7265</td>
</tr>
<tr>
<td>14.13.04</td>
<td>0.0378</td>
<td>0.1865</td>
<td>0.0567</td>
<td>0.2487</td>
</tr>
<tr>
<td>14.13.04.02</td>
<td>0.0127</td>
<td>0.2542</td>
<td>0.0476</td>
<td>0.5000</td>
</tr>
</tbody>
</table>

4.5 Conclusions

In this chapter, an improved HMC method based on over-sampling and consistency ensemble is proposed for solving the gene function prediction problem. According to the characteristics of HMC task, two measures are implemented to improve the HMC performance by introducing the hierarchy constraint into learning procedures.

Firstly, in typical HMC applications, classes at lower hierarchy levels often have very small frequencies, while the frequency of classes at higher levels tends to be high. Therefore, most of training datasets for binary classifiers at lower hierarchy levels are always imbalanced. For such imbalance dataset learning, a hierarchical SMOTE approach is proposed to preprocess the imbalanced training subsets for binary SVM classifiers. It can improve the performance of classifiers by changing the distribution of the imbalanced datasets.
Table 4.4: Classification results on the four FunCat datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>MPrec</th>
<th>MRec</th>
<th>MF-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Flat</td>
<td>0.4771</td>
<td>0.3049</td>
<td>0.3720</td>
</tr>
<tr>
<td></td>
<td>TPR</td>
<td>0.6056</td>
<td>0.3568</td>
<td>0.4490</td>
</tr>
<tr>
<td></td>
<td>Proposed</td>
<td>0.6832</td>
<td>0.4362</td>
<td>0.5324</td>
</tr>
<tr>
<td></td>
<td>Proposed + ck-SVM</td>
<td>0.6905</td>
<td>0.4487</td>
<td>0.5439</td>
</tr>
<tr>
<td>D2</td>
<td>Flat</td>
<td>0.4598</td>
<td>0.2876</td>
<td>0.3598</td>
</tr>
<tr>
<td></td>
<td>TPR</td>
<td>0.5529</td>
<td>0.3624</td>
<td>0.4378</td>
</tr>
<tr>
<td></td>
<td>Proposed</td>
<td>0.6513</td>
<td>0.4292</td>
<td>0.5174</td>
</tr>
<tr>
<td></td>
<td>Proposed + ck-SVM</td>
<td>0.6597</td>
<td>0.4472</td>
<td>0.5331</td>
</tr>
<tr>
<td>D3</td>
<td>Flat</td>
<td>0.4694</td>
<td>0.2917</td>
<td>0.3598</td>
</tr>
<tr>
<td></td>
<td>TPR</td>
<td>0.5883</td>
<td>0.3342</td>
<td>0.4263</td>
</tr>
<tr>
<td></td>
<td>Proposed</td>
<td>0.6728</td>
<td>0.4318</td>
<td>0.5260</td>
</tr>
<tr>
<td></td>
<td>Proposed + ck-SVM</td>
<td>0.6704</td>
<td>0.4342</td>
<td>0.5270</td>
</tr>
<tr>
<td>D4</td>
<td>Flat</td>
<td>0.4628</td>
<td>0.2932</td>
<td>0.3590</td>
</tr>
<tr>
<td></td>
<td>TPR</td>
<td>0.5785</td>
<td>0.3427</td>
<td>0.4304</td>
</tr>
<tr>
<td></td>
<td>Proposed</td>
<td>0.6593</td>
<td>0.4186</td>
<td>0.5121</td>
</tr>
<tr>
<td></td>
<td>Proposed + ck-SVM</td>
<td>0.6685</td>
<td>0.4276</td>
<td>0.5216</td>
</tr>
</tbody>
</table>

Secondly, the hierarchy constraint of classes is not taken into account in separate binary classifiers, i.e. it is not automatically imposed that an instance belonging to a class should belong to all its super-classes. For this problem, an improved TPR consistency approach is used to combine the results of binary probabilistic SVM classifiers. It can improve the classification results and guarantee the hierarchy constraint of classes.

Four yeast data sets from FunCat are used to evaluate the proposed method. Experiment results show that the proposed method is an efficient hierarchical multi-label classification method, it significantly outperforms the basic TPR ensemble and the Flat ensemble.
Chapter 5

Protein Structure Prediction on HP Model Using a Hybrid EDA

5.1 Introduction

Protein structure prediction (PSP) is one of the most important problems in computational biology. A protein is a chain of amino acids (also called as residues) that folds into a specific native tertiary structure under certain physiological conditions. Understanding protein structures is vital to determining the function of a protein and its interaction with DNA, RNA and enzyme. The information about its conformation can provide essential information for drug design and protein engineering. While there are over a million known protein sequences, only a limited number of protein structures are experimentally determined. Hence, prediction of protein structures from protein sequences using computer programs is an important step to unveil proteins’ three dimensional conformation and functions.

Because of the complexity of the PSP problem, simplified models like Dill’s HP-lattice [68] model have become the major tools for investigating general properties of protein folding. In HP model, 20-letter alphabet of residues is simplified to a two-letter alphabet, namely $H$ (hydrophobic) and $P$ (polar). Experiments on small protein suggest that the native state of a protein corresponds to a free energy minimum. This hypothesis is widely accepted, and forms the basis for computational prediction of a protein’s conformation from its residue sequence. The problem of finding such a minimum energy configuration has been proved to be NP-complete for the bi-dimensional (2-D) [32] and tri-dimensional (3-D) lattices [12]. Therefore, a deterministic approach is always not practical for this problem.

Many genetic algorithm (GA) based methods have been proposed to solve the PSP on HP model
in recent years [111, 49, 104, 31, 26]. However, it has been acknowledged that the crossover operators, particularly one-point crossover and uniform crossover, in a conventional GA do not perform well for this problem [64, 44]. On the other hand, R. Santana et al. (2008) pointed out that the evolutionary algorithms able to learn and use the relevant interactions that may arise between the variables of the problem can perform well for this kind of problems. Estimation of distribution algorithm (EDA) is known as one of such kind of evolutionary algorithms.

In the EDAs [67, 71], instead of using conventional crossover and mutation operations, probabilistic models are used to sample the genetic information in the next population. The use of probabilistic models, especially, models taking into account bivariate or multivariate dependencies between variables, allows EDAs to capture genetic tendencies in the current population effectively. In brief, these algorithms construct, in each generation, a probabilistic model that estimates the probability distribution of the selected solutions. Dependency regulars are then used to generate next generation solutions during a simulation step. It is expected that the generated solutions share a number of characteristics with the selected ones. In this way, the search leads to promising areas of the search space.

In Ref. [96], the EDAs that use Markov probabilistic model or other probabilistic models outperform other population-based methods when solving the HP model folding problem, especially for the long sequence protein instances. But those methods have three obvious disadvantages as follow. 1) For most long sequence protein instances, the chance of finding the global optimum is very low, and the algorithm often need be set by very large generation number and population size for finding the global optimum. 2) For some deceptive sequences, those methods can only find the suboptimum solutions. 3) In those methods, a backtracking method is used to repair invalid individuals sampled by the probabilistic model of EDAs. For a traditional backtracking algorithm, the computational cost of repairing procedure is very heavy for those long sequence instances.

This chapter proposes a hybrid method to solve above problems of the EDAs based method for HP model protein folding. Firstly, a composite fitness function containing the information of folding structure core (H-Core) is introduced to replace the traditional fitness function of HP model. The proposed fitness function is expected to select better individuals for probabilistic model of EDAs algorithm. It can help to increase the chance of finding the global optimum and reduce the complexity of EDA (population size and the number of generation needed). Secondly, local search with guided operators is utilized to refine the found solutions for improving efficiency of EDA. The local information of solutions found so far can be helpful for exploitation, while the global information can guide the search for exploring promising areas. Local search with guided operators generates
offspring through combination of global statistical information and the location information of solutions found so far. Thirdly, an improved backtracking-based repairing method is introduced to repair invalid individuals sampled by the probabilistic model of EDAs for the long sequence protein instances. The traditional backtracking repairing procedure will produce heavy computational cost for searching invalid closed-areas of folding structure. In the improved method, to avoid entering invalid closed-areas, a detection procedure for feasibility is introduced when selecting directions for the residues in backtracking searching procedure. It can significantly reduce the number of backtracking searching operation and the computational cost for the long protein sequences.

The rest of the chapter is organized as follows. Section 5.2 gives a brief overview of protein HP model and the EDAs. Section 5.3 describes the proposed hybrid EDA for HP model protein folding. It includes the proposed composite fitness function and local search with guided operators. Section 5.4 formulates the improved backtracking repairing algorithm for invalid solutions. Section 5.5 presents the experiment results of the introduced method. Finally, the conclusions and further work directions are given.

5.2 Protein HP Model and Problem Representation

5.2.1 Protein Folding and HP model

Proteins are macromolecules made out of 20 different residues. A residue has a peptide backbone and a distinctive side chain group. The peptide bond is defined by an amino group and a carboxyl group connected to an alpha carbon to which a hydrogen and side chain group is attached. Residues are combined to form sequences which are considered as the primary structure of proteins. The secondary structure is the locally ordered structure brought about via hydrogen bounding mainly within the peptide backbone. The most common secondary structure elements in proteins are the alpha helix and the beta sheet. The tertiary structure is the global folding of a single polypeptide chain.

Under specific conditions, a protein sequence folds into a unique native 3-D structure. Each possible protein fold has an associated energy. The thermodynamic hypothesis states that the native structure of a protein is the one for which the free energy achieves the minimum. Based on this hypothesis, many methods are proposed to search for the protein native structure by defining an approximation of the protein energy and utilizing the optimization methods. These approaches mainly differ in the type of energy approximation employed and in the characteristics of the protein modeling.
The well-known Dills HP model is used in this chapter. The HP model takes into account the hydrophobic interaction as the main driving force in protein folding. In HP model, each amino acid is represented as a bead, and connecting bonds are represented as lines. In this approach, a protein is considered as a sequence $S \in \{H, P\}^+$, where $H$ represents a hydrophobic residue and $P$ represents a hydrophilic or polar residue. The HP model restricts the space of conformations to self-avoiding paths on a lattice in which vertices are labeled by the residues.

Given a pair of residues, they are considered neighbors if they are adjacent either in the chain (connected neighbors) or in the lattice but not connected in the chain (topological neighbors). Let $\varepsilon_{HH}$ denote the interaction energy between topological neighbor of two $H$ residues, $\varepsilon_{PP}$ for two $P$ residues, $\varepsilon_{HP}$ for a $H$ residue and a $P$ residue. An energy function is defined as the total energy of topological neighbors with $\varepsilon_{HH} = -1$ and $\varepsilon_{PP} = \varepsilon_{HP} = 0$. The HP problem is to find the folding conformation that minimizes the total energy $E(x)$. Figure 5.1 shows the graphical representation of a possible configuration for sequence $HPHPPHPPHPHPH$ in 2-D HP model, hydrophobic residuals are represented by black beads and polar residuals by white beads. The energy that the HP model associates with this configuration is -6.

Although more complex models have been proposed, the HP model remains a focus of research in computational biology, chemical and statistical physics. By varying the energy function and the bead sequence of the chain (the primary structure), effects on the native state structure and the kinetics (rate) of folding can be explored, and this may provide insights into the folding of real proteins. In particular, the HP model has been used to investigate the energy landscapes of proteins, i.e. the variation of their internal free energy as a function of conformation. In evolutionary computation, the model is still employed because of its simplicity and its usefulness as a test-bed for new evolutionary optimization approaches [96].
5.2.2 Problem Representation

In EDAs [67], there are neither crossover nor mutation operators. Instead, the new population is sampled from a probability distribution, which is estimated from a database that contains the selected individuals from the previous generation. Thus, the interrelations between the different variables that represent the individuals are explicitly expressed through the joint probability distribution associated with the individuals selected at each generation.

Essentially EDAs assume that it is possible to build a model of the promising areas of the search space, and use this model to guide the search for the optimum. In EDAs, modeling is achieved by building a probabilistic graphical model that represents a condensed representation of the features shared by the selected solutions. Such a model can capture different patterns of interactions between subsets of the problem variables, and can conveniently use this knowledge to sample new solutions.

In the algorithm of protein folding optimum, one of the important problems is how to present a specific conformation. To embed a hydrophobic pattern \( S \in \{H, P\}^+ \) into a lattice, there are three methods of Cartesian Coordinate, Internal Coordinate and Distance Matrix [64] as follows.

1) Cartesian Coordinate: The position of residues is specified independently from other residues.
2) Internal Coordinate: The position of each residue depends upon its predecessor residues in the sequence. There are two types of internal coordinate: absolute directions where the residue directions are relative to the axes defined by the lattice, and relative directions where the residue directions are relative to the direction of the previous move.
3) Distance Matrix: The location of a given residue is computed by means of its distance matrix.

Krasnogor et al. (1999) [64] performed an exhaustive comparative study using evolutionary algorithms (EAs) with relative and absolute directions. The experimental results show that relative directions almost always outperform absolute directions over square and cubic lattice, while absolute directions have better performances when facing triangular lattices. Experimental evidence suggests internal coordinates with relative directions should be used. However, in general, it is difficult to assess the effectiveness of direction encoding on an EAs performance.

In this chapter, the representation of internal coordinates with relative direction is used. The position of each residue depends upon the previous move. Relative direction representation presents the direction of each residue relative to the main chain next turn direction. This representation can reduce the direction number of each position. For 2-D HP model, the set of direction is left, right and forward (L, R, F). And it is left, right, forward, up and down, (L, R, F, U, D) for the 3-D HP
model. For example, by relative direction representation, the representation of the protein structure shown in Fig. 5.1 is \( s = (RFRLLRFFRLR) \).

It can be noted that the backward direction is not used, because the backward direction will cause overlap in this representation. Thus, this representation can reduce the position collision in a certain degree to guarantee the self-avoiding walk folding procedure. There are other advantages of the relative direction representation. One is that the sequence conformation can be presented as one dimension array. The most important is that the change of a start direction will not influence the structure of other parts in sequence.

5.2.3 Probabilistic Model of EDA

It is very important for EDAs to select an appropriate probabilistic model according to a given application problem. The probabilistic model is represented by conditional probability distributions for each variable and estimated from the genetic information of selected individuals in the current generation. Therefore, the type of probabilistic model also influences the number and strength of the interactions learned by the model.

In Ref. [96], three probabilistic models for EDAs are proposed to solve the HP model problem: \( k \)-order Markov model, tree model and mixtures of trees model. In practice, the \( k \)-order Markov model is an appropriate probabilistic model for the HP model problem, where \( k \geq 0 \) is a parameter of the model. It can effectively embody the self-avoiding folding characteristics of the HP model problem, because it is assumed that positions of adjacent residues are related in the protein folding procedure.

The \( k \)-order Markov model can encode the dependencies between the move of a residue and the moves of the previous residues in the sequence, and this information can be used in the generation of solutions. It is described as follows. The joint probability mass function of \( X \) is denoted as \( p(X) \). And use \( p(X_i = x_i | X_j = x_j) \) or the simplified form \( p(x_i | x_j) \) to denote the conditional probability distribution of \( X_i = x_i \) given \( X_j = x_j \).

In the \( k \)-order Markov model, the value of variable \( X_i \) depends on the values of the previous \( k \) variables. The joint probability distribution can be factorized as follows:

\[
p_{MK}(X) = p(x_1, \cdots, x_{k+1}) \prod_{i=k+2}^{n} p(x_i | x_{i-1}, \cdots, x_{i-k}) \tag{5.2.1}
\]

Since the structure of the Markov model is given, it can be used to construct the probabilistic
model through computing the marginal and conditional probabilities of the set of selected individuals and to sample the new generation. To sample a new solution, first variables in the factor \((x_1, \cdots, x_{k+1})\) are generated and the rest of variables are sampled according to the order specified by the Markov factorization.

5.3 A Hybrid EDA for Protein Folding Based on HP Model

5.3.1 Proposed Composite Fitness Function

In order to increase the chance of finding the global optimum and reduce the complexity of EDA (population size and the number of generation needed), a composite fitness function containing the information of folding structure core is introduced to replace the traditional fitness function of HP model.

It is well known that the energy potential in the HP model reflects the fact that hydrophobic residues have a propensity to form a hydrophobic core. The Hs (hydrophobic residues) form the protein core and the Ps (hydrophilic or polar residues) tend to remain in the outer surface. As shown in Fig. 5.2, the inner kernel, called the H-Core [56], is compact and mainly formed of Hs while the outer kernel consists mostly of Ps. The H-Core Center is called HCC. The H-Core is a rectangle-like area in 2-D lattice and cube-like space in 3-D lattice. The coordinates of HCC can be calculated by follows equations.

\[
x_{HCC} = \frac{1}{n_H} \sum_{i=1}^{n_H} x_i, \quad y_{HCC} = \frac{1}{n_H} \sum_{i=1}^{n_H} y_i \\
z_{HCC} = \frac{1}{n_H} \sum_{i=1}^{n_H} z_i
\]

(5.3.1)
where \( n_H \) is the sum of hydrophobic residues in solution, \( x_i, y_i \) and \( z_i \) (for 3-D HP model) are the coordinates of hydrophobic residues position in lattice. The number of Hs in inner kernel H-Core (denoted as \( N_{HC}(x) \)) can be calculated through search surrounding rectangle area (cube space for 3-D HP model) of HCC.

The number of Hs in inner kernel H-Core is an important characteristic for the folding solution. It also reflects the optimum degree of solution. In practice, as showed in Fig. 5.3, for two solutions with same basic HP model energy \( E(x) \) (defined by the number of topological neighbor residues in lattice), the solution with bigger H-Core has more similar to the optimum solution, and it also has more biology significance. The two possible solutions of the Instance s8 (length is 64) have same basic HP model energy (-25), but they have different \( N_{HC}(x) \) values (9 and 15 respectively). Obviously, the solution Fig. 5.3(b) has more similar to the optimum solution.

In the proposed method, a novel composite fitness function containing the information of H-Core for the \( k \)-order Markov EDA is introduced as follow.

\[
\text{Fit}_{cp}(x) = \omega(-E(x)) + (1 - \omega)N_{HC}(x) \tag{5.3.2}
\]

where, \( E(x) \) is the total energy of the interaction between topological neighbor residues of HP model \( (\varepsilon_{HH} = -1, \varepsilon_{PP} = \varepsilon_{HP} = 0) \). \( N_{HC}(x) \) is the number of Hs in inner kernel H-Core. \( \omega \) is weight parameter of the fitness function, and \( \omega > 0.5 \) is chosen in practice, because the interaction energy \( E(x) \) is the dominant characteristic of protein folding solution.
5.3.2 Local Search with Guided Operators

An efficient evolutionary algorithm should make use of both the local information of solutions found so far and the global information about the search space. The local information of solutions found so far can be helpful for exploitation, while the global information can guide the search for exploring promising areas. The search in EDAs is mainly based on the global information, but local search is an exploitation method based on local information. Therefore, it is worthwhile investigating whether combining local search with EDA could improve the performance of the EDA [121, 120].

Local search with a set of guided operators is implemented in the proposed hybrid EDA. Some of these operations have been utilized as mutations in the previous GA and ant colony optimizations studies of protein folding [104]. But in this chapter, “guided operators” means that those operations are implemented only under some special conditions.

Take 2-D HP model as example, the special conditions defined as follow. 1) Guided operation should guarantee the validity of individual, i.e. it can not produce position collision in lattice. If the positions of residues in lattice are changed, the object positions must be empty. 2) Guided operation should follow a basic principle that make Hs as near as possible to the HCC and Ps far away from the HCC according to the relative position in lattice, as shown in Fig. 5.4.

The way of choosing individuals to implement local search is described as follow. In each iteration procedure of EDAs, use the composite fitness function (described by Eq. 5.3.2) to sort the selection individuals. According to the distribution of individuals’ fitness, randomly select some individuals (the number is a certain percentage of the population) in each fitness domain to implement...
the local search with guided operators.

EDAs extract globally statistical information from the previous search and then build a probabilistic model for modeling the distribution of best solutions visited in the search space. However, the information of the locations of best individual solutions found so far is not directly used for guiding the further search. Local search with guided operator generates offspring through combination of global statistical information and the location information of solutions found so far. The resultant solution can (hopefully) fall in or close to a promising area which is characterized by the probabilistic model.

5.4 Improved Backtracking-based Repairing Method

5.4.1 Backtracking Method

In HP model, a collision is the embedding of two different peptides onto the same vertex of the lattice. As each member of the initial population of EA based method is randomly generated, it may represent an illegal conformation resulting one or more collisions when embedded. Similarly, crossover and mutation operations of GAs, sampling from probabilistic model operation of EDAs and other genetic operations may produce additional collisions.

In Ref. [31], a backtracking algorithm was introduced to repair the positional collisions. It utilizes backtracking strategy to search feasible positions for collision residues in folding procedure. This particular algorithm constitutes a simple yet efficient approach for the purposed task. Its pseudocode is showed in Fig. 5.5.

The algorithm receives three parameters. The first one is $\lambda$, a table containing the allowed moves for each residue in the protein; thus, $\lambda_k$ is a list of allowed moves for $(k + 1)$-th residue and $\lambda_{k,r}$ is the $r$-th move. Although $\lambda$ may contain in principle the full set of moves, in general $|\lambda|$ will not be the same for every $k$. The second parameter $s$ is a partial conformation involving $|s|$ residues. As to the third parameter, it is a Boolean flag used to finalize the execution of the algorithm as soon as a feasible conformation is found. Notice finally that the operator :: represents the sequence concatenation operator.

5.4.2 Disadvantage of Traditional Backtracking-based Method

The basic backtracking method mentioned in the previous section has been shown to be a simple and efficient means of positional collision repairing for protein folding. But in practice, the repairing computational cost is very heavy for long sequence instances of more than 50 residues.
1. R-Backtracking (↓ λ:MOVE[], ↓↑ s:MOVE[],
↑ SolutionFound:bool).

2. if Feasible(s) then

3. if |s| = n – 1 then

4. SolutionFound ← TRUE

5. else

6. SolutionFound ← FALSE

7. i ← 1

8. while ¬SolutionFound ∧ (i ≤ |λ|_{|s|}) do

9. if Detect − fea (⟨λ|s,i⟩, s) then

10. s’ ← s :: ⟨λ|s,i⟩

11. R-Backtracking (λ, s’, SolutionFound)

12. endif

13. i ← i + 1

14. if SolutionFound then

15. s ← s’

16. endif

17. endwhile

18. endif

19. else

20. SolutionFound ← FALSE

21. endif

Figure 5.5: Pseudocode of the backtracking repairing algorithm.
In the generic search procedure of protein folding, especially for the long protein sequences, the EAs based algorithm will produce a lot of valid and invalid individuals that contain closed-areas (or closed-spaces in 3-D circumstance). The Fig. 5.6 shows a valid 2-D individual’s conformation contains two closed-areas. When the basic backtracking method is used to repair invalid individuals contain closed-areas, some invalid closed-areas made by backtracking searching folding procedure will produce computational cost wastes.

Take the 2-D circumstance as example, as showed in Fig. 5.7(a), there is a closed-area formed by residues from 1 to \( n \). If the folding procedure select right (R) as the next direction for \( n+1 \) residue, it will enter the closed-area. Thus, even if the size of this closed-area can not satisfy the length of remain residues (called it as invalid closed-area), the traditional backtracking method will still search all empty position in closed-area by backtracking operation. This will produce a large number of computational cost wastes. According to our experiment, this phenomenon takes place with a high probability in repairing procedures for long sequence proteins.

### 5.4.3 Improved Method

To solve the above problem, in the improved method, a detection for feasibility is introduced. The detection procedure is implemented before selecting direction for next residue to avoid entering an invalid closed-area (The procedure \( \text{Detect \ – \ fea} \) in Line 9 of Fig. 5.5).

The pseudocode of the introduced detection algorithm is shown in Fig. 5.8. The main idea of the detection procedure is described as follow.
1) The current boundaries in lattice is defined as shown in Fig. 5.7(a), the scale of boundary coordinates is larger one position than current filled area and will be changed with current folding procedure. For example, four current boundaries of the 2-D solution shown in Fig. 5.6 are $x = -11$ (left), $x = 4$ (right), $y = 13$ (up) and $y = -4$ (down). They can be used to check the folding procedure whether enter a closed-area. If the detection meet the current boundaries, the folding procedure will not enter a closed-area.

2) A search approach, similar to Floodfill strategy, is utilized to count possible empty positions connected to the detected direction $\langle \lambda_{|s|,i} \rangle$, i.e. those empty positions which could be arrived through this direction. The Floodfill-like search approach count and label the possible empty positions based on a queue $Q$. If the queue $Q$ becomes empty, it means that all possible empty positions are labeled.

3) In the operation of detection procedure, if the current boundaries are met or the number of counted empty positions is larger than the length of remain residues, it means that the folding will not enter a closed-area or entered closed-area is not invalid. Under such circumstance, the detected direction $\langle \lambda_{|s|,i} \rangle$ could be chosen for the next residue.

For long length protein sequences, there are many invalid closed-area in folding procedure. The improved method can significantly reduce the computational cost. Although the detection procedure has some computational cost, it is far less than the cost of backtracking searching operations for invalid closed-areas.
1. \[ \text{Detect-fea (} \downarrow \langle \lambda \rangle, \downarrow s:\text{MOVE[]} \text{):bool.} \]
2. \[ \text{Calculate current boundaries according to } s \]
3. \[ \text{Set label for every positions in } s \text{ (i.e. not empty).} \]
4. \[ \text{if } \text{Feasible}(s::\langle \lambda \rangle) \text{ then} \]
5. \[ \text{counter = 0 and set an empty queue } Q \]
6. \[ \text{Add } \langle \lambda \rangle \text{ to the end of } Q \]
7. \[ \text{while } Q \text{ is not empty do} \]
8. \[ \text{x=first element of } Q \]
9. \[ \text{if position } x \text{ is unlabeled} \]
10. \[ \text{Set label for position } x \]
11. \[ \text{counter = counter + 1} \]
12. \[ \text{endif} \]
13. \[ \text{if (position } x \text{ meet the current boundaries) or (counter is larger than the length of remain residues)} \]
14. \[ \text{Return TRUE} \]
15. \[ \text{endif} \]
16. \[ \text{Remove the first element of } Q \]
17. \[ \text{if west neighbor of } x \text{ is unlabeled} \]
18. \[ \text{Set label for } \text{west} - x \]
19. \[ \text{Add } \text{west} - x \text{ to the end of } Q \]
20. \[ \text{endif} \]
21. \[ \text{Check and process other three (five for 3D) neighbor positions of } x \text{ using similar strategies Step (17)-(20)} \]
22. \[ \text{endwhile} \]
23. \[ \text{endif} \]
24. \[ \text{Return FALSE} \]

Figure 5.8: Pseudocode of the detection procedure.
The main reason of the improvement is that the improved method can significantly reduce the number of backtracking operation. The folding procedure implements backtracking operation only under few special circumstances. As shown in Fig. 5.7(b), if the folding procedure has selected the right (R) direction for the \( n + 1 \) residues. But at \( n + i \) position, the folding procedure produce two sub-closed-areas and all of two are invalid closed-area for remain residues. The folding procedure should implement a backtracking operation under this situation. It will back to the \( n + i - 1 \) residue and search other possible directions.

5.5 Experiments

5.5.1 Problem Benchmark

Eleven benchmark HP instances are used to evaluate the searching capability of the proposed method. The first nine instances are taken from the Tortilla 2-D HP Benchmarks \(^1\), and the last two are taken from Ref. [96]. In Tab. 5.1, \( E^\ast \) is the optimal or best-known energy value, \( H_i, P_i \) and \((\cdots)_i\) indicate \( i \) repetitions of the relative symbol or subsequence. It is important to highlight that most randomly generated amino acid sequences do not behave like natural proteins, because the latter are products of natural selection. Likewise, most randomly generated sequences of \( H \) and \( P \) residues in the HP model do not fold to a single conformation [96].

5.5.2 Results of the Hybrid EDA for HP Model

In order to test the effects of the composite fitness function and the local search with guided mutation, different experiments are implemented by using one of them independently. The composite fitness function can help to reduce the complexity of EDA, it can obtain same results with basic \( k \)-order Markov EDA (MK-EDA) by using less population size and generation number. The local search with guided mutation can help to obtain the global optimum for some instances. But it seems that combination of two strategies can get much better results in practice. The performance of MK-EDA for \( k \in \{2, 3, 4\} \) are investigated, and the algorithm performs very well when \( k = 3 \).

In the experiments of the hybrid EDA, all algorithms use a population size of 2000 individuals. Truncation selection is used as selection strategy. In this strategy, individuals are ordered by fitness, and the best \( T \ast \) \( \text{PopSize} \) are selected where \( T \) is the truncation coefficient. The parameter \( T = 0.15 \) is used in our algorithms. The best elitism scheme is also implemented in algorithms, the set of selected solutions in the current generation are passed to the next generation. The stop

\(^1\)http://www.cs.sandia.gov/tech-reports/compbio/tortilla-hp-benchmarks.html
criteria considered are a maximum number of generation \(G = 1000\) or that the number of different individuals in the population falls below 5. For the protein instances of \(s_6\) to \(s_{11}\), the improved backtracking-based method is used to repair the invalid solutions.

The results of the proposed method comparing with the MK-EDA for the 2-D HP Model shown in Tab. 5.2. It includes the best solution and the percentage of times the best solution has been found in 100 experiments. The results of MK-EDA are also obtained by our experiments with same EDA parameters \((Pop = 2000, G = 1000\) and \(T = 0.15\)) as the proposed method. The experiment results show that the proposed method has more chance to find global optimum or suboptimum solution for long sequences. The MK-EDA cannot find the global optimum of the deceptive sequences and long sequences \(s_7, s_{10}, s_9\), but the proposed method can find the global optimum of the sequences \(s_7, s_{10}, s_{11}\), and can find the second best solution for sequence \(s_11\). Figure 5.9 shows the best fitness for one representative run of the instance \(S7\).

The performance of the proposed method comparing with the best results achieved with other evolutionary and Monte Carlo optimization algorithms is shown in Tab. 5.3 (2-D HP model) and Tab. 5.4 (3-D HP model). The results of other method are cited from Ref. [96, 56]. The experiment results show that none of the algorithms are able to outperform the rest of algorithms for all the instances. The PERM is one of the best contenders in all cases except \(s_8\) in which its result is very poor. It shows that the proposed method is very competitive with the other existing algorithms.

<table>
<thead>
<tr>
<th>No.</th>
<th>Size</th>
<th>(E^*)</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>20</td>
<td>-9</td>
<td>(HPHP2HP2PHP2PHP2PHP)</td>
</tr>
<tr>
<td>s2</td>
<td>24</td>
<td>-9</td>
<td>(HP2(HP2)6H2)</td>
</tr>
<tr>
<td>s3</td>
<td>25</td>
<td>-8</td>
<td>(P2HP2(H2P4)3H2)</td>
</tr>
<tr>
<td>s4</td>
<td>36</td>
<td>-14</td>
<td>(P3H2P2H2P3H7P2H2P4H2P2HP)</td>
</tr>
<tr>
<td>s5</td>
<td>48</td>
<td>-23</td>
<td>(P2H(P2H2)2P3H10P6(P2P2)2P2H2H5)</td>
</tr>
<tr>
<td>s6</td>
<td>50</td>
<td>-21</td>
<td>(H2(PH)3PH1P(PH3)2PHP4(PH)4H)</td>
</tr>
<tr>
<td>s7</td>
<td>60</td>
<td>-36</td>
<td>(P3H3PHP3P3H10PHP4P3H6PHP2PHP)</td>
</tr>
<tr>
<td>s8</td>
<td>64</td>
<td>-42</td>
<td>(H12(PH)2(P2H2)2P2H(P2H2)2P2H(P2H2)2P2(HP)2H12)</td>
</tr>
<tr>
<td>s9</td>
<td>85</td>
<td>-53</td>
<td>(H4P4H12P6(H12P3)3H(P2H2)2P2HPH)</td>
</tr>
<tr>
<td>s10</td>
<td>100</td>
<td>-48</td>
<td>(P6HPHP2P3HP3PH3P2H2P2P2P2PH3PH10PH2PH7P11H7)</td>
</tr>
<tr>
<td>s11</td>
<td>100</td>
<td>-50</td>
<td>(P3H2P2H4P2H3(PH2)3H2P6HP2P6HP2HPH2PH11P2H3P)</td>
</tr>
</tbody>
</table>

These results show that the proposed method is very competitive with the other existing algorithms.
for the PSP on HP models. It should be noted that all fitness values of the proposed method in the comparing results are calculated by basic HP-model fitness definition. The composite fitness function is only used in optimization procedure of EDA.

<table>
<thead>
<tr>
<th>No.</th>
<th>E*</th>
<th>Proposed Method</th>
<th>MK-EDA</th>
<th>Percentage</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>-9</td>
<td>-9</td>
<td>-9</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>s2</td>
<td>-9</td>
<td>-9</td>
<td>-9</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>s3</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>s4</td>
<td>-14</td>
<td>-14</td>
<td>-14</td>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td>s5</td>
<td>-23</td>
<td>-23</td>
<td>-23</td>
<td>22</td>
<td>7</td>
</tr>
<tr>
<td>s6</td>
<td>-21</td>
<td>-21</td>
<td>-21</td>
<td>92</td>
<td>57</td>
</tr>
<tr>
<td>s7</td>
<td>-36</td>
<td>-36</td>
<td>-35</td>
<td>24</td>
<td>12</td>
</tr>
<tr>
<td>s8</td>
<td>-42</td>
<td>-42</td>
<td>-42</td>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>s9</td>
<td>-53</td>
<td>-53</td>
<td>-52</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>s10</td>
<td>-48</td>
<td>-48</td>
<td>-48</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>s11</td>
<td>-50</td>
<td>-49</td>
<td>-48</td>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>

### 5.5.3 Results of Comparing Computational Cost

The hybrid EDA is an improved method based on the MK-EDA. The detailed computational cost analysis of the MK-EDA method can be found in the Ref. [96]. Comparing with the MK-EDA, there are three modifications in the proposed method: 1) the composite fitness function; 2) the local search with guided operations; 3) the improved backtracking-based repairing method for the long protein instances. As far as the computational cost is concerned, modifications of 1) and 2) will produce some additional computational cost. The modification 3) can significantly reduce the repairing costs for EDA invalid individuals.

To demonstrate the computational cost of the hybrid EDA comparing with MK-EDA, some practical experiments in 2-D are implemented. Two comparing methods are implemented with same parameters (population:1000, generation:100, the truncation selection of parameter $T = 0.15$) and same computational environment². Because there are few closed-areas existing in short protein folding, the improved backtracking-based repairing method can not improve the EDA efficiency for short instances. In the comparing experiments for the short instances of $s1$ to $s5$, same basic

² All experiments are performed on the computers with Intel Xeon 2.20 GHz processor, and 1 GB of RAM.
Figure 5.9: The best fitness for one representative run of instance $s7$.

Table 5.3: Results achieved by different search methods for 2-D HP model.

<table>
<thead>
<tr>
<th>No.</th>
<th>Proposed Method $H(X)$</th>
<th>MK-EDA $H(X)$</th>
<th>GA $H(X)$</th>
<th>NewACO $H(X)$</th>
<th>PERM $H(X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>-9</td>
<td>-9</td>
<td>-9</td>
<td>-9</td>
<td>-9</td>
</tr>
<tr>
<td>s2</td>
<td>-9</td>
<td>-9</td>
<td>-9</td>
<td>-9</td>
<td>-9</td>
</tr>
<tr>
<td>s3</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
<td>-8</td>
</tr>
<tr>
<td>s4</td>
<td>-14</td>
<td>-14</td>
<td>-14</td>
<td>-14</td>
<td>-14</td>
</tr>
<tr>
<td>s5</td>
<td>-23</td>
<td>-23</td>
<td>-22</td>
<td>-23</td>
<td>-23</td>
</tr>
<tr>
<td>s6</td>
<td>-21</td>
<td>-21</td>
<td>-21</td>
<td>-21</td>
<td>-21</td>
</tr>
<tr>
<td>s7</td>
<td>-36</td>
<td>-35</td>
<td>-34</td>
<td>-36</td>
<td>-36</td>
</tr>
<tr>
<td>s8</td>
<td>-42</td>
<td>-42</td>
<td>-37</td>
<td>-42</td>
<td>-38</td>
</tr>
<tr>
<td>s9</td>
<td>-53</td>
<td>-52</td>
<td>-51</td>
<td>-53</td>
<td>-53</td>
</tr>
<tr>
<td>s10</td>
<td>-48</td>
<td>-47</td>
<td>-47</td>
<td>-48</td>
<td>-48</td>
</tr>
<tr>
<td>s11</td>
<td>-49</td>
<td>-48</td>
<td>-47</td>
<td>-50</td>
<td>-50</td>
</tr>
</tbody>
</table>
Table 5.4: Results achieved by different search methods for 3-D HP model.

<table>
<thead>
<tr>
<th>No.</th>
<th>Proposed Method ( H(X) )</th>
<th>MK-EDA ( H(X) )</th>
<th>Hybrid GA ( H(X) )</th>
<th>IA ( H(X) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>-11</td>
<td>-11</td>
<td>-11</td>
<td>-11</td>
</tr>
<tr>
<td>s2</td>
<td>-13</td>
<td>-13</td>
<td>-11</td>
<td>-13</td>
</tr>
<tr>
<td>s3</td>
<td>-9</td>
<td>-9</td>
<td>-9</td>
<td>-9</td>
</tr>
<tr>
<td>s4</td>
<td>-18</td>
<td>-18</td>
<td>-18</td>
<td>-18</td>
</tr>
<tr>
<td>s5</td>
<td>-29</td>
<td>-29</td>
<td>-28</td>
<td>-28</td>
</tr>
<tr>
<td>s6</td>
<td>-30</td>
<td>-29</td>
<td>-22</td>
<td>-23</td>
</tr>
<tr>
<td>s7</td>
<td>-49</td>
<td>-48</td>
<td>-48</td>
<td>-41</td>
</tr>
<tr>
<td>s8</td>
<td>-51</td>
<td>-50</td>
<td>-46</td>
<td>-42</td>
</tr>
</tbody>
</table>

backtracking repairing methods are used in two comparing methods. For the long instances of \( s_6 \) to \( s_{10} \), the improved backtracking-based repairing method is used in the hybrid EDA.

The number of backtracking searching operation and computer CPU-Time are recorded. The average backtracking searching operations and the CPU-Times of 10 runs are shown in Tab. 5.5. According to the results of the short instances of \( s_1 \) to \( s_5 \), it can be found that the local search operations and the composite fitness calculation in the hybrid EDA produce some additional computational costs. But it is not very serious. The results of the long instances of \( s_6 \) to \( s_{10} \) show that the proposed repairing method can significantly reduce the repairing costs. It not only covers the additional computational costs caused by local search and composite fitness calculation, but also improves the algorithm efficiency remarkably.

The backtracking searching operations of each generation for sequence \( s_8 \) (the length is 64), \( s_9 \) (the length is 85) and \( s_{10} \) (the length is 100) are also be counted and shown in Fig. 5.10, Fig. 5.11 and Fig. 5.12. It can be found that the improved backtracking-based repairing method can significantly reduce the number of backtracking searching operation. And the longer the protein sequence length is, the more remarkable the improvement achieves.

5.6 Conclusions

In this chapter, a novel hybrid EDA method is introduced to solve the HP model problem. For the basic \( k \)-order Markov EDA, it has very low chance to find the general optimum for those long sequence and deceptive protein instances. A composite fitness function containing information of folding structure core is introduced to replace the traditional fitness function of HP model. It can
Table 5.5: Comparing results of the improved backtracking repairing method in 2-D HP model.

<table>
<thead>
<tr>
<th>No.</th>
<th>Size</th>
<th>AVG-Backtracking operation (MK-EDA, Proposed Method)</th>
<th>AVG-CPUTime (Hour) (MK-EDA, Proposed Method)</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>20</td>
<td>2.1492E+4, 2.1496E+4</td>
<td>0.2181, 0.2309</td>
</tr>
<tr>
<td>s2</td>
<td>24</td>
<td>2.5715E+4, 2.5715E+4</td>
<td>0.2659, 0.2761</td>
</tr>
<tr>
<td>s3</td>
<td>25</td>
<td>2.6834E+4, 2.7044E+4</td>
<td>0.2774, 0.2805</td>
</tr>
<tr>
<td>s4</td>
<td>36</td>
<td>3.8898E+4, 3.8797E+4</td>
<td>0.3059, 0.3203</td>
</tr>
<tr>
<td>s5</td>
<td>48</td>
<td>5.2577E+5, 5.2617E+5</td>
<td>0.4341, 0.4659</td>
</tr>
<tr>
<td>s6</td>
<td>50</td>
<td>5.7842E+6, *5.4887E+6</td>
<td>0.6249, 0.5768</td>
</tr>
<tr>
<td>s7</td>
<td>60</td>
<td>7.4545E+6, *6.7478E+6</td>
<td>0.9276, 0.7516</td>
</tr>
<tr>
<td>s8</td>
<td>64</td>
<td>9.6731E+6, *7.2236E+6</td>
<td>1.1276, 0.8661</td>
</tr>
<tr>
<td>s9</td>
<td>85</td>
<td>2.0829E+8, *1.1196E+7</td>
<td>12.3077, 1.6086</td>
</tr>
<tr>
<td>s10</td>
<td>100</td>
<td>2.6531E+8, *1.9765E+7</td>
<td>15.7125, 2.0007</td>
</tr>
</tbody>
</table>

Figure 5.10: The number of backtracking searching operations for instance s8.
Figure 5.11: The number of backtracking searching operations for instance s9.

Figure 5.12: The number of backtracking searching operations for instance s10.
help to select better individuals for probabilistic model of EDA algorithm. In addition, local search with guided operators is utilized to refine found solutions for improving efficiency of EDA.

For the disadvantage of heavy computational cost of the traditional backtracking method which used to repair the invalid individuals in population. It will produce heavy computational cost for searching invalid closed-areas of folding structure. An improved method is introduced to reduce the repairing computational cost for the long protein sequences. A detection procedure for feasibility is added to avoid entering invalid closed-areas when selecting directions for the residues. Thus, it can significantly reduce the number of backtracking searching operation and the computational cost for long sequence protein. It can be noted that the improved backtracking repairing method can be used in all EA based PSP methods that need to repair invalid individuals. And the underlying mutations are implemented for individuals in repairing procedure.

Experimental results demonstrate that the proposed method outperform the basic EDA method. At the same time, the proposed method is very competitive with other existing algorithms for the PSP on HP models. Further research is needed to determine more efficient local search strategies and probabilistic models of EDA for protein HP model problem.
Chapter 6

Protein HP Model Folding Based on Adaptive Niching EDA with Balance Searching

6.1 Introduction

Estimation of Distribution Algorithms (EDAs) [67] are a new class of evolutionary algorithms. Such approaches, instead of using conventional crossover and mutation operations, use probabilistic models to sample the genetic information in the population. The use of probabilistic models, especially those models taking into account bivariate or multivariate dependencies between variables, allows EDAs to capture genetic tendencies in current population effectively. In brief, these algorithms construct, in each generation, a probabilistic model that estimates the probability distribution of selected solutions. Dependency regulars are then used to generate next generation solutions during a sampling step. It is expected that the generated solutions share a number of characteristics with the selected ones. In this way, the search leads to promising areas of the search space [67].

By explicitly handling interactions between different variables that represent the individuals, the EDA based methods can get much better results than other population-based methods when solving some optimization problems. But for those optimization problems with irregular and complex multimodal landscapes, the EDAs still suffer from the drawback of premature convergence similar to other Evolutionary Algorithms (EAs). A simple yet popular explanation for the occurrence of premature convergence is the loss of diversity. But high diversity does not imply a better EA performance. This is tightly coupled to the question of exploration vs. exploitation [74, 23]. Enforcing diversity at the early phases of evolution ensures a broad exploration of the search space. However, at the phases when high exploitation is needed, such promotion could be counterproductive or
even destructive. Finding a modifiable balance between exploration and exploitation is a key to this issue [3, 41, 106].

For solving complex optimization problems, an efficient EA searching procedure with a balance between exploration and exploitation can be imagined as follows. At the beginning phase, the searching procedure should cover the search space as much as possible to avoid the premature convergence. But at the end phase of searching, the exploitation should be enforced to find good solutions. In other words, the degree of exploration should be monotonically decreasing, while the degree of exploitation should be monotonically increasing during the searching procedure. Inspired by this thought, in this chapter, an adaptive niching EDA based on Affinity Propagation (AP) clustering analysis [45] is proposed to improve the performance of EDA. As a machine learning method, the AP clustering is used as an intelligent and efficient way to mine the information of EDA search procedure. It is not only used to partition the niches for population adaptively, but also used to mine the searching history information of EDA to guide the EDA searching.

The clearing procedure [89] is used as a basic niching frame in the proposed method. 1) AP clustering is used to adaptively partition the population into niches during a run of EDA. The individuals are clustered before submitting them to niching clearing. A cluster can be seen as a niche, and the niche number and the niche radiuses may vary obviously for different generations. 2) A mechanism of niche capacity selection based on the Boltzmann scheme is proposed to realize a balance searching between exploration and exploitation. A niche novelty metric is proposed to record the searching history information by detecting the overlapping distributions of the niches in successive generations. The metric is used to evaluate the degree of search space that has been explored by the previous searching. Two different selection strategies, novelty-proportionate selection based on the novelty metric and fitness-proportionate selection based on the best fitness in a niche, are assembled by a Boltzmann weight in the niche capacity selection mechanism. Tuned by the Boltzmann weight, at the beginning phase of searching, the dominating novelty-proportionate selection can guide the EDA searching to cover the search space as much as possible; at the end phase of searching, the dominating fitness-proportionate selection can enforce the EDA searching to exploit the already found promising areas.

The proposed adaptive niching EDA is used for solving the protein HP model folding. For the protein HP model problems, the \( k \)-order Markov EDA proposed in Ref. [96] outperforms other population-based methods. But for most of the long protein sequences, the chance of finding the global optimum is very low, and for some deceptive sequences, it can only find the suboptimum solutions. In this chapter, the proposed method is used to improve the performance of the \( k \)-order
Markov EDA for protein HP problem. In addition, three benchmark functional multimodal optimization problems based on continuous EDA with single Gaussian probabilistic model are also used to evaluate the proposed method in continuous situation. Experiment results show that the proposed adaptive niching EDA is an efficient method.

The rest of the chapter is organized as follows. Section 6.2 briefly introduces the premature convergence and the AP clustering. Section 6.3 formulates the proposed adaptive niching EDA based on AP clustering analysis. Section 6.4 carries out experiments on solving the optimization problem of HP model protein folding based on a $k$-order Markov probabilistic model. Finally, Section 6.5 presents the conclusions.

### 6.2 Premature Convergence and Niching Clearing

#### 6.2.1 Premature Convergence

A critical problem when dealing with EAs is the phenomenon of premature convergence. Premature convergence occurs when the population of an evolution searching reach a suboptimal state where genetic operators can no longer produce offspring which outperform their parents. In that case, the search process will likely be trapped in a region containing a non-global optimum. A simple yet popular explanation for the occurrence of this phenomenon is the loss of diversity. In that context the diversity refers to a (genetic) variation of the population members [48].

Various enhanced strategies for diversity maintenance have been proposed which target different stages of the evolution process. Niching method is one of the well-known strategies proposed to reduce the effects of genetic drift resulting from the selection mechanism, to allow the formation and the maintenance of different solutions and to prevent the EA from being trapped in local optima [48]. In the niching method, the analogy with nature is straightforward. As in an ecosystem there are different subsystems (niches) that contain many diverse species (subpopulations). The number of elements in a niche is determined by its resources and by the efficiency of each individual in taking profit of these resources. Using this analogy, it is possible for a EA to maintain its population diversity during the generations.

#### 6.2.2 Niching Clearing Procedure

Among niching methods, fitness sharing based methods are widely known and used [27, 28, 103, 57, 34]. In sharing algorithms, the number of individuals in a niche is limited to the caring capacity of the niche. Individuals within a niche must share the fitness payoff offered by their niche. By
treating fitness as a resource and sharing it among all niche members it restricts the maximum number of individuals that can be supported by the niche. When a niche becomes overcrowded, payoff is spread too thin and it in each individual’s best interests to seek out another less crowded niche where they can get more payoff (fitness). Niches are a subdividing of the environment, or the fitness landscape. A population can be thought to specialize in different areas of the environment, where the idea of sharing resources between individuals induces a powerful niching behavior. It has been said [57] that through sharing, a form of localized competition for local resources leads to implicit cooperation between subpopulations (niches).

The clearing method [89] is very similar to the fitness sharing but is based on a concept of limited resources of the environment. Instead of sharing the available resources among all individuals of a subpopulation as in the fitness sharing, the clearing procedure supplies those limited resources only to the best members of each subpopulation.

As well as sharing, clearing is applied after the fitness evaluation and before selection, using a dissimilarity measure to separate the individuals in subpopulations. The capacity $\kappa$ of a niche specifies the maximum number of individuals that a niche can accept. Thus, the clearing preserves the fitness of the $\kappa$ best individuals (dominant individuals) of a niche and resets the fitness of others that belong to the same subpopulation (dominated individuals). As in the sharing methods, individuals belong to the same niche (or subpopulation) if their distance in the search space is less than a dissimilarity threshold $\sigma$ (clearing radius). The clearing can be coupled with elitism strategies naturally to preserve the best elements of niches during the generations. If the two parameters of the niche radius and the niche capacity are correctly estimated, the clearing method can be considered as an excellent niching method [98].

### 6.3 Adaptive Niching EDA with Balance Searching

The clearing procedure is used as a basic niching frame in the proposed adaptive niching EDA. To implement the niching clearing in the selection procedure of EDAs, the individuals are firstly grouped into many niches by a certain partition strategy. A static radius $\sigma$ is used to partition the niches in a conventional niching clearing method. Then, the $\kappa$ best individuals are selected from each niche respectively and a combination of selected individuals is used to estimate probabilistic model of EDA. The capacity parameter $\kappa$ is defined as the maximum number of winners that a niche can accept. In the conventional method, the capacity $\kappa$ is set as a static value for all niches, and it can be set in a range from 1 to the population size, so that the niching effect can be, respectively,
maximized or minimized as convenient.

In the proposed adaptive niching EDA, as showed in Fig. 6.1, a balance niching searching strategy based on adaptive clustering is introduced into the selection procedure to improve the performance of EDA. 1) The values of the clearing radius $\sigma$ are determined adaptively by AP clustering during a run of EDA. The individuals are clustered before submitting them to the niching clearing and a cluster can be seen as a niche. 2) A niche capacity $\kappa$ selection mechanism based on the Boltzmann scheme is utilized to realize a balance searching. Two different selection strategies of considering exploration and exploitation are assembled by the Boltzmann scheme in the selection mechanism. A combination of the selected individuals from all niches is used to estimate the probabilistic model of EDA.

### 6.3.1 A Mechanism of Niche Capacity Selection

In each iteration of the niching EDA, a combination of the selected individuals from all niches is used to estimate a probabilistic model to capture genetic tendencies in the current population. For a certain niche, the number of winners (the $\kappa$ selected individuals) represents the degree of EDA searching for this niche area. Therefore, the niche capacity selection can be utilized to tune the searching behavior of EDA. An efficient EDA with a balance searching should have the behaviors described as follows. The degree of exploration is monotonically decreasing, while the degree of exploitation is monotonically increasing during the search run. To realize an efficient searching procedure with a balance between exploration and exploitation, a mechanism of niche capacity
selection based on the Boltzmann scheme is defined as follows.

\[
\kappa_t(n_i) = M \times \left[ \omega_t \times E_i(n_i) + (1 - \omega_t) \times E_r(n_i) \right]
\]

\[
\omega_t = \left[ \frac{e^{\left(\frac{t}{T_{\text{max}}} \right)} - 1}{e - 1} \right]^\alpha, \quad 1 \leq i \leq c_{l_t}
\]  

(6.3.1)

where \(M\) is the number of selected individuals in each iteration, \(n_i\) denotes the \(i\)-th niche, \(\omega_t\) the Boltzmann weight of the \(t\)-th iteration, \(T_{\text{max}}\) the maximum number of iterations, \(\alpha\) the scaling factor used to tune the Boltzmann scheme, \(c_{l_t}\) the cluster number of the \(t\)-th iteration population, \(E_i(n_i)\) and \(E_r(n_i)\) the selection probabilities of the \(i\)-th niche in the \(t\)-th iteration defined by certain exploitation and exploration strategies, respectively. The detail definitions of \(E_i(n_i)\) and \(E_r(n_i)\) will be described in the following subsections.

The round value \(\lfloor \kappa_t(n_i) \rfloor\) is used as the winner number of niche \(n_i\). A combination of the winners from all niches is used as the selected individuals (the number is \(\sum_{i=1}^{c_{l_t}} \lfloor \kappa_t(n_i) \rfloor\)) to estimate the probabilistic model of the \(t\)-th generation.

In the proposed mechanism of niche capacity selection, two different selection strategies are assembled by a Boltzmann weight \(\omega_t\) related with the generation \(t\). Tuned by the Boltzmann weight \(\omega_t\), at the beginning phase of EDA, the searching procedure explores the search areas as more as possible by dominating the exploration strategy; at the end phase of searching, the searching exploitation is enforced to refine the already found solutions by dominating the exploitation strategy.

### 6.3.2 The Exploration Probability \(E_r(n_i)\)

According to the definition of the niche capacity selection mechanism described in the previous subsection, the exploration strategy (represented by the probability \(E_r(n_i)\)) is expected to preserve a diversity in the population of EDA. It should guide the EDA searching to cover the solution space as much as possible to avoid premature convergence. Selecting the same number of winners from all niches is a simple exploration strategy, i.e., the niche capacities for all niches are set as the average value decided by the selected population size and the cluster number of current population.

But the average selection is not a good strategy suitable to the adaptive niching scheme. Because in the adaptive niching scheme, the individuals in the population are separated into niches adaptively; in addition, the niche number and the niche radiuses vary obviously in different generations. The varied niche radiuses may produce a problem of irregular niche distribution, i.e., at some generations, too many niches are distributed in a search space that has already been explored by the previous searching. A simple experiment can be used to explain this problem. An experiment of
optimizing the 2-dimension Schwefel function (Eq. 6.4.3 and the function landscape showed in Fig. 6.5) is implemented by the continuous Univariate Marginal Distribution Algorithm (UMDA$_C$) [67]. The function domain is [-500 500] and the population size is 50. The populations of four different searching phases (generations of 1-5, 11-15, 21-25 and 31-35) are stored and analyzed by AP clustering. The obtained cluster center exemplars are showed in Fig. 6.2. From the distributions of cluster center exemplars in different generations, it can be found that the adaptive niching scheme often produces the irregular niche distribution problems. To solve this problem, two levels of diversity preservation are considered in the adaptive niching scheme: one is the diversity preservation at individual level; the other is the one at niche (subpopulation) level.

In the proposed exploration searching strategy, the population clustering information is utilized to record the EDA searching history for solving the irregular niche distribution problem. From the overlapping information of the cluster center exemplar distribution in successive generations,
the search space parts which have been explored by the previous searching can be found. A niche novelty metric is proposed to record the overlapping information, and it is used to evaluate the degree of the corresponding niche explored by the previous searching. The niches with smaller novelty metric values will be punished moderately at the exploration searching strategy to realize the diversity preservation at the niche level.

The niche novelty metric \( \text{nov}_t(n_i) \) of niche \( n_i \) in the \( t \)-th generation is defined as follows.

\[
\text{nov}_t(n_i) = \beta^{O_t(n_i)}, \quad 0 < \beta < 1
\]  

(6.3.2)

where \( n_i \) denotes the \( i \)-th niche, \( \beta \) is the penalty parameter for niche novelty, and \( O_t(n_i) \) the overlapping parameter of niche \( n_i \) in the \( t \)-th generation.

Let us describe a niche \( n_i \) at generation \( t \) by its center \( C_t(n_i) \) and radius \( \sigma_t(n_i) \) where the center determining its position and the radius determining its area. If the center of a niche \( n_i \) of \( t \) generation is inside the niche \( n_j \) of \( t-1 \) generation, that is, the distance between two centers \( C_{t-1}(n_j) \) and \( C_t(n_i) \) is less than the radius \( \sigma_{t-1}(n_j) \), the niche \( n_i \) of \( t \) generation is said to be overlapped by the niche \( n_j \) of \( t-1 \) generation.

Set \( O_t(n_i) = 0 \) as the initial value. If the niche \( n_i \) of \( t \) generation is one of the \( k (k > 1) \) niches that are overlapped by the niche \( n_j \) of \( t-1 \) generation, then set \( O_t(n_i) = O_{t-1}(n_j) + 1 \) with a probability of \( (k-1)/k \). That is, among the \( k \) niches only one keeps 0, while other \( k-1 \) niches are set to \( O_{t-1}(n_j) + 1 \).

The niche novelty metric reflects the information of search space explored by the EDA. The exploration searching probability \( E_{r,t}(n_i) \) is defined by a novelty-proportionate strategy as follows.

\[
E_{r,t}(n_i) = \frac{\text{nov}_t(n_i)}{\sum_{j=1}^{cl_t} \text{nov}_t(n_j)}
\]  

(6.3.3)

where \( cl_t \) is the cluster number of the \( t \)-th iteration population.

### 6.3.3 The Exploitation Probability \( E_{i,t}(n_i) \)

A popular fitness-proportionate method is used as the exploitation strategy in the proposed adaptive niching EDA. The exploitation searching probability \( E_{i,t}(n_i) \) is defined as follows.

\[
E_{i,t}(n_i) = \frac{f_t(n_i)}{\sum_{j=1}^{cl_t} f_t(n_j)}
\]  

(6.3.4)

where \( f_t(n_i) \) is the maximum fitness value of the \( i \)-th niche in the \( t \)-th iteration.

The fitness-proportionate niche capacity selection is defined based on the best fitness value in a niche. It can guide the EDA to search the already found promising areas of searching space.
6.3.4 Framework of the Proposed Method

Following a typical EDA framework, as showed in Fig. 6.3, the major steps of the proposed adaptive niching EDA can be summarized as follows.

1) Generate the first population of $N$ individuals randomly. The individuals should be initialized to cover different areas in the search space.

2) Repair invalid individuals and evaluate each individual of the current population. Then the individuals of current population is grouped into clusters by using AP method based on a certain similarity metric.

3) Calculate the novelty metric for each cluster (niche) by Eq. 6.3.2 and evaluate each niche by its best solution found. According to the proposed niche capacity $\kappa$ selection mechanism
of the Boltzmann scheme defined by Eq. 6.3.1, a number of \( \sum_{i=1}^{c_t} \kappa_i(n_i) \) individuals are selected from the niches obtained in Step 2).

4) Estimate the probabilistic model from the selected individuals.

5) Generate a new population of individuals based on the probabilistic model obtained in Step 4). The elitist preservation strategy is implemented, so a set of good solutions in the current generation is passed to the next generation.

6) Repeat Steps 2) to 5) until certain stop criterion is met (e.g. a maximum number of generations, a homogeneous population, or no improvement after a certain number of generations).

6.4 Experiments

6.4.1 Experiments on Continuous Benchmark Functions

To evaluate the proposed method for continuous optimization problems, the proposed adaptive niching EDA is applied to solving three benchmark functional multimodal problems firstly. In cases of continuous EDAs, Gaussian model is the most widely used probabilistic model. But there are difficulties in solving those multimodal functions with irregular and complex landscapes, since conventional EDAs are said to be easily misled by such kinds of deceptive landscapes and then converge to poor results.

Benchmark Functions

The following three benchmark functions from [119] are used to evaluate the proposed method.

1) Kowalik function

\[
f(x) = \sum_{i=1}^{11} \left( a_i - \frac{x_1(b_i^2 + b_1x_2)}{b_i^2 + b_1x_3 + x_4} \right)^2
\]  

(6.4.1)

where \(-5 \leq x_i \leq 5\), and \(a_i\) is [0.1957 0.1947 0.1735 0.1600 0.0844 0.0627 0.0342 0.0323 0.0235 0.0246], \(b_i^{-1}\) is [0.25 0.5 1 2 4 6 8 10 12 14 16].

2) Shekel function

\[
f(x) = \sum_{i=1}^{n} \frac{1}{((x - \bar{a}_i)(x - \bar{a}_i)^T + \bar{c}_i)}^{-1}
\]  

(6.4.2)

where \(n\) is set to 5 in the experiment and the parameters of \(a_{ij}\) and \(c_i\) are listed in Table 6.1.
Table 6.1: Parameters of the Shekel function \((n = 5)\)

<table>
<thead>
<tr>
<th>(i)</th>
<th>(a_1)</th>
<th>(a_2)</th>
<th>(a_3)</th>
<th>(a_4)</th>
<th>(\cdots)</th>
<th>(c_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>(\cdots)</td>
<td>0.1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>(\cdots)</td>
<td>0.2</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>(\cdots)</td>
<td>0.2</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>(\cdots)</td>
<td>0.4</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>7</td>
<td>3</td>
<td>7</td>
<td>(\cdots)</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Table 6.2: Settings for the three test functions

<table>
<thead>
<tr>
<th></th>
<th>Dimension</th>
<th>Domain</th>
<th>Type</th>
<th>Optimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kowalik</td>
<td>4</td>
<td>[-5 5]</td>
<td>Min.</td>
<td>0.0003075</td>
</tr>
<tr>
<td>Shekel ((n=5))</td>
<td>30</td>
<td>[0 10]</td>
<td>Max.</td>
<td>10.0134</td>
</tr>
<tr>
<td>Schwefel</td>
<td>30</td>
<td>[-500 500]</td>
<td>Min.</td>
<td>-12569.5</td>
</tr>
</tbody>
</table>

3) Schwefel function.

\[
f(\vec{x}) = \sum_{i=1}^{n} -x_i \sin(\sqrt{|x_i|})
\]

Table 6.2 lists the specifications of three test functions. And the particular two-dimension cases for the Shekel and Schwefel are showed in Fig. 6.4 and Fig. 6.5.

Experiment Results

Three EDA based methods of UMDA\(_C\), CEGDA\(_{BGe}\) [72] and NichingEDA [38] are used to compare with the proposed method. Among them, the results of CEGDA\(_{BGe}\) and NichingEDA are cited from their original papers, the results of the UMDA\(_C\) are obtained by our experiments. In Ref. [72, 38], the maximal evaluation number for CEGDA\(_{BGe}\) and NichingEDA are \(4 \times 10^5\) and \(5 \times 10^5\), respectively; all results have been averaged over 30 independent runs.

In our experiments, the maximal evaluation number for UMDA\(_C\) and the proposed method are set as \(4 \times 10^5\) (population: 1000 and generation: 400). A Gaussian probabilistic model similar to the UMDA\(_C\) is used in the proposed method. All algorithms use a truncation selection size of 500. Their initial populations are sampled randomly within the function domain, and the elitism is used in the evolution. For those invalid individuals sampled in each generation, a repairing procedure is used to change the values of each out of range variable to the minimum (respectively maximum)
bounds of the function domain. The penalty parameter $\alpha$ in Eq. 6.3.1 for the niche novelty and the scaling factor $\beta$ in Eq. 6.3.2 for the Boltzmann scheme are set as: $\alpha = 2$ and $\beta = 0.95$, respectively. About the parameters of AP clustering (defined in Ref. [45]), Euclidean distance is used as the similarity metric of individuals; the maximum number of iterations is set as 1000, and early terminate parameter is set as 100 (i.e., the clustering procedure will be terminated if the estimated exemplars stay fixed for continuous 100 iterations); the damping factor, which may be needed if oscillations occur, is set as 0.9.

The results on three functions are listed in Table 6.3, Table 6.4 and Table 6.5. The best values, mean values and standard errors are computed based on the best solutions that an algorithm has found in 30 independent runs. The results show that the proposed method outperforms other methods for solving the three functional optimizations. To observe the computational overheads of the proposed method, the CPU-Time costs (the computers with Intel Xeon 2.20 GHz processor, and 1 GB of RAM) are listed in Table 6.6.

### 6.4.2 Experiments on Protein HP Model

In this section, the proposed adaptive niching EDA is applied to solving a real complicated discrete EDA optimization problem, the HP model protein folding based on a $k$-order Markov (MK-EDA)
Figure 6.5: Plot of the Schwefel function (dimension=2).

Table 6.3: Experimental results for the Kowalik function

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Best</th>
<th>Mean</th>
<th>Std</th>
</tr>
</thead>
<tbody>
<tr>
<td>UMDA&lt;sub&gt;c&lt;/sub&gt;</td>
<td>0.00085553</td>
<td>0.0031</td>
<td>0.0017</td>
</tr>
<tr>
<td>NichingEDA</td>
<td>0.00030750</td>
<td>0.00030874</td>
<td>2.7899e-006</td>
</tr>
<tr>
<td><strong>Proposed</strong></td>
<td>0.00030750</td>
<td>0.00030791</td>
<td>4.173e-008</td>
</tr>
</tbody>
</table>

probabilistic model.

Experiment Setting

In this chapter, the representation of internal coordinates with relative direction [64] is used. The position of each residue depends upon the previous move. Relative direction representation presents the direction of each residue relative to the main chain next turn direction. This representation can reduce the direction number of each position. For 2-D HP model, the set of direction is left, right and forward (L, R, F). It is left, right, forward, up and down, (L, R, F, U, D) for the 3-D HP model.

During the course of EDA search procedure, initialization and sampling from probabilistic model operations may produce a lot of invalid individuals including positional collisions. A positional collision is the embedding of two different residues onto the same vertex of the lattice. The
Table 6.4: Experimental results for the Shekel function (Dim=30)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Best</th>
<th>Mean</th>
<th>Std</th>
</tr>
</thead>
<tbody>
<tr>
<td>UMDA&lt;sub&gt;c&lt;/sub&gt;</td>
<td>5.1325</td>
<td>4.3108</td>
<td>0.9326</td>
</tr>
<tr>
<td>NichingEDA</td>
<td>0.3550</td>
<td>0.2230</td>
<td>0.0564</td>
</tr>
<tr>
<td>CEGDA&lt;sub&gt;BGe&lt;/sub&gt;</td>
<td>10.0134</td>
<td>10.0134</td>
<td>8.8818e-015</td>
</tr>
<tr>
<td>Proposed</td>
<td>10.0134</td>
<td>10.0134</td>
<td>4.7622e-015</td>
</tr>
</tbody>
</table>

Table 6.5: Experimental results for the Schwefel function

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Best</th>
<th>Mean</th>
<th>Std</th>
</tr>
</thead>
<tbody>
<tr>
<td>UMDA&lt;sub&gt;c&lt;/sub&gt;</td>
<td>-5928.24</td>
<td>-5424.81</td>
<td>202.437</td>
</tr>
<tr>
<td>NichingEDA</td>
<td>-8733.51</td>
<td>-8005.39</td>
<td>270.755</td>
</tr>
<tr>
<td>CEGDA&lt;sub&gt;BGe&lt;/sub&gt;</td>
<td>-10773.1</td>
<td>-6760.35</td>
<td>2624.33</td>
</tr>
<tr>
<td>Proposed</td>
<td>-11064.82</td>
<td>-8741.61</td>
<td>2124.33</td>
</tr>
</tbody>
</table>

improved backtracking-based repairing method proposed in Ref. [22] is used to repair invalid individuals for the long sequence protein instances. In this repairing method, to avoid entering invalid closed-areas, a detection procedure for feasibility is introduced when selecting directions for the residues in backtracking searching procedure. It can significantly reduce the number of backtracking searching operation and the computational cost for the long protein sequences.

\[ p_{MK}(X) = p(x_1, \cdots, x_{k+1}) \prod_{i=k+2}^{n} p(x_i | x_{i-1}, \cdots, x_{i-k}). \quad (6.4.4) \]

The \( k \)-order Markov model of EDA is described by Eq. 6.4.4. It can encode the dependencies between the move of a residue and the moves of the previous residues in the sequence, and this information can be used in the generation of solutions. To sample a new solution, first variables in the factor \( (x_1, \cdots, x_{k+1}) \) are generated and the rest of variables are sampled according to the order specified by the Markov factorization.

Table 6.6: Computing costs of the proposed method and the UMDA<sub>c</sub>.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Kowalik</th>
<th>Shekel</th>
<th>Schwefel</th>
</tr>
</thead>
<tbody>
<tr>
<td>UMDA&lt;sub&gt;c&lt;/sub&gt; (second)</td>
<td>11.27</td>
<td>13.34</td>
<td>14.95</td>
</tr>
<tr>
<td><strong>Proposed</strong>  (second)</td>
<td>1551.65</td>
<td>1618.73</td>
<td>1692.39</td>
</tr>
</tbody>
</table>
Experiment Results

In our experiments, the benchmark protein instances (s4 to s11 in Chapter 5 Tab. 5.1) from the *Tortilla* 2-D HP Benchmarks and Ref. [96] are used to evaluate the searching capability of the proposed method.

Table 6.7: Results of comparing with MK-EDA and the proposed method for 2-D HP model.

<table>
<thead>
<tr>
<th>No.</th>
<th>$E^*$</th>
<th>Runs</th>
<th>Proposed method</th>
<th>MK-EDA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Best Perc. Mean Time(H)</td>
<td>Best Perc. Mean Time(H)</td>
</tr>
<tr>
<td>s4</td>
<td>-14</td>
<td>100</td>
<td>-14 32% -13.87 1.5324</td>
<td>-14 5% -11.05 0.5321</td>
</tr>
<tr>
<td>s5</td>
<td>-23</td>
<td>100</td>
<td>-23 49% -22.72 1.9113</td>
<td>-23 7% -20.45 0.7112</td>
</tr>
<tr>
<td>s6</td>
<td>-21</td>
<td>100</td>
<td>-21 96% -20.83 2.5884</td>
<td>-21 57% -18.32 0.9125</td>
</tr>
<tr>
<td>s7</td>
<td>-36</td>
<td>100</td>
<td>-36 34% -35.02 3.1270</td>
<td>-35 12% -32.65 1.5278</td>
</tr>
<tr>
<td>s8</td>
<td>-42</td>
<td>100</td>
<td>-42 28% -41.71 4.5327</td>
<td>-42 4% -36.24 2.1232</td>
</tr>
<tr>
<td>s11</td>
<td>-50</td>
<td>15</td>
<td>-49 6.67% -47.81 15.9712</td>
<td>-48 6.67% -44.38 9.5449</td>
</tr>
</tbody>
</table>

The results of MK-EDA are also obtained by our experiments, the maximal evaluation number of the proposed method and MK-EDA are set as $2 \times 10^6$, all algorithms use a population size of 2000 individuals. Truncation selection parameter $T = 0.15$, i.e., the best $T \ast PopSize$, are selected for learning probabilistic model. The penalty parameter $\alpha$ in Eq. 6.3.1 for the niche novelty and the scaling factor $\beta$ in Eq. 6.3.2 for the Boltzmann scheme are set as: $\alpha = 1$ and $\beta = 0.80$. The stop criteria considered are a maximum number of generation $G = 1000$ or that the number of different individuals in the population falls below 5. The performance of MK-EDA for $k \in \{2, 3, 4\}$ are investigated in our experiments, and the algorithm performs very well when $k = 3$. In the AP clustering, Hamming distance is used as the similarity metric of individuals, other parameters are set same as the previous section experiments.

The results of the proposed method compared with the MK-EDA for the 2-D HP Model are listed in Table 6.7. It includes the best solution, the percentage of times the best solution has been found, the mean fitness of selected solutions in the last generation has been found and computing cost (CPU-Time). The experiment results show that the proposed method has more chance to find global optimum or suboptimum solution for protein sequences than MK-EDA. The MK-EDA cannot find the global optimum of the deceptive sequences and long sequences $s7$, $s9$, $s10$ and $s11$. 
The proposed method also can find the global optimum of the sequences $s_7$, $s_9$ and $s_{10}$, and can find the second best solution for sequence $s_{11}$. But the proposed method need more computing cost than the MK-EDA because of the clustering and niching operations. Figure 6.6 shows the number of different individuals in the selected population for one representative run of instance $s_{10}$ in 2-D model. One 3-D conformation of the best results for sequences $s_5$, $s_7$ and $s_8$ are shown in Fig. 6.7, Fig. 6.8 and Fig. 6.9 respectively.

Table 6.8: Results of different search methods for 2-D HP model.

<table>
<thead>
<tr>
<th>No.</th>
<th>Proposed $H(X)$</th>
<th>MK-EDA $H(X)$</th>
<th>GA $H(X)$</th>
<th>NewACO $H(X)$</th>
<th>PERM $H(X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>s4</td>
<td>-14</td>
<td>-14</td>
<td>-14</td>
<td>-14</td>
<td>-14</td>
</tr>
<tr>
<td>s5</td>
<td>-23</td>
<td>-23</td>
<td>-22</td>
<td>-23</td>
<td>-23</td>
</tr>
<tr>
<td>s6</td>
<td>-21</td>
<td>-21</td>
<td>-21</td>
<td>-21</td>
<td>-21</td>
</tr>
<tr>
<td>s7</td>
<td>-36</td>
<td>-35</td>
<td>-34</td>
<td>-36</td>
<td>-36</td>
</tr>
<tr>
<td>s8</td>
<td>-42</td>
<td>-42</td>
<td>-37</td>
<td>-42</td>
<td>-38</td>
</tr>
<tr>
<td>s9</td>
<td>-53</td>
<td>-52</td>
<td>-51</td>
<td>-53</td>
<td></td>
</tr>
<tr>
<td>s10</td>
<td>-48</td>
<td>-47</td>
<td>-47</td>
<td>-48</td>
<td></td>
</tr>
<tr>
<td>s11</td>
<td>-49</td>
<td>-48</td>
<td>-47</td>
<td>-50</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.9: Results of different search methods for 3-D HP model.

<table>
<thead>
<tr>
<th>No.</th>
<th>Proposed $H(X)$</th>
<th>MK-EDA $H(X)$</th>
<th>Hybrid GA $H(X)$</th>
<th>IA $H(X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>s4</td>
<td>-18</td>
<td>-18</td>
<td>-18</td>
<td>-18</td>
</tr>
<tr>
<td>s5</td>
<td>-29</td>
<td>-29</td>
<td>-28</td>
<td>-28</td>
</tr>
<tr>
<td>s6</td>
<td>-30</td>
<td>-29</td>
<td>-22</td>
<td>-23</td>
</tr>
<tr>
<td>s7</td>
<td>-49</td>
<td>-48</td>
<td>-48</td>
<td>-41</td>
</tr>
<tr>
<td>s8</td>
<td>-51</td>
<td>-50</td>
<td>-46</td>
<td>-42</td>
</tr>
</tbody>
</table>

The performance of the proposed method compared with the best results achieved with other evolutionary and Monte Carlo optimization algorithms is shown in Table 6.8 (2-D HP model) and Table 6.9 (3-D HP model). The results of other methods are cited from Ref. [96]. The experiment results show that none of the algorithms are able to outperform the rest of algorithms for all the instances. The PERM (Pruned-Enriched Rosenbluth Method) [60] is a biased chain growth algorithm with re-sampling (“population control”) with depth-first implementation. It is one of the best
Figure 6.6: The number of different individuals in the selected population for one representative run of instance \( s10 \).

Figure 6.7: One conformation of the best results for the instance \( s5 \) in 3-D HP model (fitness=-29, star denotes \( H \) and circle denotes \( P \)).
Figure 6.8: One conformation of the best results for the instance s7 in 3-D HP model (fitness=-49, star denotes H and circle denotes P).

Figure 6.9: One conformation of the best results for the instance s8 in 3-D HP model (fitness=-51, star denotes H and circle denotes P).
contenders in all cases except s8 in which its result is very poor. This shows that our method is very competitive with other existing algorithms for the PSP on lattice HP models.

6.5 Conclusions

This chapter proposed a novel adaptive niching EDA based on AP clustering analysis. The clearing procedure is utilized as a basic niching frame in the proposed method. The AP clustering is not only used to partition the niches for population adaptively, but also used to mine the searching history information of EDA. A niche novelty metric is introduced to evaluate the degree of search space that has been explored by detecting the niche overlapping information of continuous generations. A Boltzmann scheme niche capacity selection is proposed to improve the EDA performance by a balance searching between exploration and exploitation. At the beginning phase of the search, the dominating novelty-proportionate selection based on the niche novelty metric can guide the EDA searching to cover the search space as much as possible. At the end of the search process, the dominating fitness-proportionate selection based on the best fitness value in a niche can enforce the EDA searching to exploit the already found promising areas.

Two different categories of optimization problems are used to evaluate the proposed adaptive niching EDA. Experiment results show that the proposed adaptive niching EDA is an efficient method for solving the optimization problems with irregular and complex multimodal landscapes.
Chapter 7

Conclusions

7.1 Summary

The gene function prediction and protein structure prediction are two important domains where machine learning techniques are applied in bioinformatics. Generally, the first is a classification problem and the second is an optimization problem. Supervised and unsupervised classification methods are often used to predict gene function, such as Clustering, SVM, NN, etc. Evolutionary algorithm (EA) based methods are the main optimization technologies for protein structure prediction, such as genetic algorithm (GA), estimation distribution algorithms (EDAs), particle swarm optimization (PSO), etc.

For the complicated tasks, such as the hierarchical multi-label gene function classification and the long protein sequence structure prediction, applying machine learning methods simply usually cannot obtain expectable results. The different characteristics of the applications will require novel modifications of existing machine learning methods.

In this thesis, improved machine learning methods are explored for solving complicated genomics and proteomic applications. More precisely, SVM based multi-label and hierarchical multi-label classification methods are developed for solving the gene function prediction problems, and novel EDAs based methods are proposed for solving the protein HP model problems. The information of the biology data is extracted and used to improve the performances of machine learning methods, and some delicate techniques are also combined according to the characteristics of application problem.

The following are the main conclusions of this thesis:

1. A composite kernel based SVM (ck-SVM) model can be used for solving nonlinear classification tasks in gene function prediction, which conventional nonlinear kernel SVMs can not
work efficiently.

2. A multi-label classification based on label ranking and delicate decision boundary SVM can be used for solving multi-label gene function classification.

3. A hierarchical multi-label classification (HMC) method based on over-sampling and hierarchy constraint can be used for solving the FunCat gene function prediction problem.

4. A hybrid Estimation of EDA can be used for solving the protein structure problem on HP model.

5. An adaptive niching EDA with balance searching based on clustering analysis can be used to enhance the performance of EDA.

7.2 Topics for Future Research

Although a lot of progress has been made, there are still many aspects that need further investigations.

1. More characteristics and properties of the proposed composite kernel based SVM are need be investigated, such as the efficient subset partition methods and composite kernel estimation techniques.

2. Applications of the proposed hierarchical multi-label method in other domains should be studied, such as in text classification and object recognition.

3. Applications of the proposed adaptive niching EDA in other protein structure prediction problems should be researched, such as in protein side chain placement problem and tag single nucleotide polymorphism (SNP) selection.
Appendix A

Support Vector Machines for Classification

This appendix presents an introduction of the support vector machines, a new generation learning system based on advances in statistical learning theory. SVMs deliver state-of-the-art performance in some real-world applications.

A.1 Introduction

A.1.1 Maximal Margin Classifiers

Consider the class of hyperplanes \( w^T x + b = 0 \), \( w \in \mathbb{R}^n \), \( b \in \mathbb{R} \), corresponding to a decision function as follow.

\[
  f(x) = \text{sign}(w^T x + b) \tag{A.1.1}
\]

The basic idea of learning from samples is to recognize the pattern of a class by examining the training points corresponding to that class. Unseen instances are assumed to lie somewhere around the known training samples. Therefore, a hyperplane should be chosen such that a small shift of the data does not result in prediction change. If the distance between the separating hyperplane and the training samples becomes too small, even test instances very close to the training samples may be classified incorrectly.

Vapnik and Chervonenkis presumed that the generalization ability depends on the distance between the hyperplane and the training points. They introduced the Generalized Portrait, a learning algorithm for separable problems, by constructing a hyperplane which maximally separates the
classes (maximum margin) as follow.

\[
\max_{w,b} \min \{ \|x - x_k\| : x \in \mathbb{R}^n, w^T x + b = 0, k = 1, \ldots, N \} \tag{A.1.2}
\]

To understand how this hyperplane can be constructed in an efficient way, some definitions are listed.

**Definition A.1.1.** (Separability)
A training dataset \( \mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N) : x_k \in \mathbb{R}^n, y_k \in \{-1, +1\}\} \) is called separable by a hyperplane \( w^T x + b = 0 \) if there exist both a unit vector \( w (\|w\| = 1) \) and a constant \( b \) such that the following equalities hold:

\[
w^T x_k + b \geq +1 \quad \text{for} \quad y_k = +1 \tag{A.1.3}
\]
\[
w^T x_k + b \leq -1 \quad \text{for} \quad y_k = -1 \tag{A.1.4}
\]

The hyperplane defined by \( w \) and \( b \) is called a *separating hyperplane*.

**Definition A.1.2.** (Margin)
Consider separating hyperplane \( H \) as defined by \( w^T x + b = 0 \).

1. the margin \( \zeta_k(w, b) \) of a training point \( x_k \) is defined as the distance between \( H \) and \( x_k \):

\[
\zeta_k(w, b) = y_k(w^T x + b) \tag{A.1.5}
\]

2. the margin \( \zeta_\mathcal{D}(w, b) \) of a set of vectors \( \mathcal{D} = x_1, \ldots, x_k \) is defined as the minimum distance from \( H \) to the vectors in \( \mathcal{D} \):

\[
\zeta_\mathcal{D}(w, b) = \min_{x_k \in \mathcal{D}} \zeta_k(w, b) \tag{A.1.6}
\]

Now consider the unit vector \( w^* \) and the constant \( b^* \) which maximize the margin of the training set \( \zeta(w, b) \) and also satisfy the condition (A.1.3) and (A.1.4). This pair of \( (w^*, b^*) \) defines the hyperplane which separates the positive from the negative examples with the largest margin. This hyperplane is also called *maximal margin hyperplane* or *optimal separating hyperplane*.

**Definition A.1.3.** (Optimal separating hyperplane)
The Optimal hyperplane of a training set \( \mathcal{D} \) is defined by

\[
(w^*, b^*) = \arg \max_{w,b} \zeta_\mathcal{D}(w, b) \tag{A.1.7}
\]

Vapnik proves on the uniqueness of the optimal separating hyperplane in [113]. With a unique \( w^* \), \( b^* \) can be described by

\[
b^* = \frac{1}{2} (\min_{k \in I_+} w^* x_k - \max_{k \in I_-} w^* x_k) \tag{A.1.8}
\]

where \( I_+ = \{k | y_k = +1\} \) and \( I_- = \{k | y_k = -1\} \).
A.1.2 Construction of the Optimal Hyperplane

Given a training dataset \( D = \{(x_1, y_1), \ldots, (x_N, y_N) : x_k \in \mathbb{R}^n, y_k \in \{-1, +1\}\} \), the optimal separating hyperplane is the solution to the optimization problem as follow.

\[
\begin{align*}
\max & \quad \zeta_D(w, b) \\
\text{subject to} & \quad \zeta_D(w, b) > 0 \\
& \quad \|w\| = 1
\end{align*}
\]

(A.1.9)

However, solving (A.1.9) is not straightforward, it involves nonlinear constraints, while the objective function itself is neither linear nor quadratic. The optimization problem can be rewritten equivalently as follow.

\[
\begin{align*}
\min & \quad \frac{1}{2}\|w\|^2 \\
\text{subject to} & \quad w^T x + b \geq +1 \quad \text{for} \quad y_k = +1 \\
& \quad w^T x + b \leq -1 \quad \text{for} \quad y_k = -1
\end{align*}
\]

(A.1.10)

Vector \( w_0 \) that solves problem (A.1.10) is related to the vector \( w^* \) solving problem (A.1.9) by the equality [113]:

\[
w^* = \frac{w_0}{\|w_0\|}
\]

(A.1.11)

This implies that the construction of the optimal hyperplane first needs to solve problem (A.1.10) with linear constraints, where the constraints can be simplified as follow.

\[
y_k(w^T x_k + b) - 1 \geq 0, k = 1, \ldots, N
\]

(A.1.12)

The Lagrangian for this problem is

\[
\mathcal{L}(w, b; e; \alpha) = \frac{1}{2} w^T w - \sum_{k=1}^{N} \alpha_k \{y_k [w^T x_k + b] - 1\}
\]

(A.1.13)

with Lagrange multipliers \( \alpha_k \geq 0 \) for \( k = 1, \ldots, N \). The solution is characterized by the saddle point of the Lagrangian

\[
\max_{\alpha} \min_{w, b} \mathcal{L}(w, b; \alpha)
\]

(A.1.14)

This leads to

\[
\begin{align*}
\frac{\partial \mathcal{L}}{\partial w} = 0 & \quad \rightarrow w = \sum_{k=1}^{N} \alpha_k y_k x_k \\
\frac{\partial \mathcal{L}}{\partial b} = 0 & \quad \rightarrow \sum_{k=1}^{N} \alpha_k y_k = 0
\end{align*}
\]

(A.1.15)
with resulting classifier

\[ y(x) = \text{sign}\left[ \sum_{k=1}^{N} \alpha_k y_k x_k^T x + b \right] \]  

Elimination of \( w \) from (A.1.13) with (A.1.15) gives the following Quadratic Programming (QP) problem as the dual problem in the Lagrange multipliers \( \alpha_k \)

\[
\max_{\alpha} J_D(\alpha) = -\frac{1}{2} \sum_{k,l=1}^{N} y_k y_l x_k^T x_l \alpha_k \alpha_l + \sum_{k=1}^{N} \alpha_k
\]

such that \( \sum_{k=1}^{N} \alpha_k y_k = 0 \)

Note that this problem is solved in \( \alpha = [\alpha_1 \ldots \alpha_N] \), and not in \( w \). There are some interesting properties of this QP problem as follows [107].

1. Global and unique solution:
   The matrix related to the quadratic term in \( \alpha \) in this quadratic form is positive definite or positive semidefinite. In the case that the matrix is positive definite (all eigenvalues strictly positive) the solution \( \alpha \) to this QP problem is global and unique [43]. When the matrix is positive semidefinite (all eigenvalues are positive with some zero eigenvalues) then the solution is global but not necessarily unique. It may happen that the solution of \((w, b)\) is unique but \( \alpha \) not. In terms of \( w = \sum_k \alpha_k y_k x_k \) this means that there may exist equivalent expansions of \( w \) which require fewer support vectors.

2. Sparseness:
   An interesting property is that many of the resulting \( \alpha_k \) values are equal to zero. Hence the obtained solution is sparse. This means that in the resulting classifier the sum should be taken only over non-zero \( \alpha_k \) values instead of all training data points:

\[ y(x) = \text{sign}\left[ \sum_{k=1}^{\#SV} \alpha_k y_k x_k^T x + b \right] \]  

where \( \#SV \) is the number of support vectors.

3. Geometric meaning of support vectors:
   The support vectors obtained from the QP problem are located close to the decision boundary.

4. Non-parametric/parametric issues:
   Note that the size of the solution vector \( \alpha \) grows with the number of data points \( N \). It means,
the dual problem actually corresponds to a non-parametric approach, while in the primal problem it is parametric because the size of $w$ is fixed independently of the number of data points. As a consequence, for large datasets it might be advantageous to solve the primal problem but in higher dimensional input space it is better to solve the dual problem as the size of the solution vector $\alpha$ does not depend on the dimension $n$ of the input space.

### A.2 Optimal Hyperplane for Linearly Nonseparable Case

The SVM solution of a linearly separable classification problem is explained in the previous subsection. However, most of classification problems with noisy data are not simple linear separation problems. No feasible solution to the margin maximization problem can be found, due to the fact that the objective function (i.e. the dual Lagrangian) is growing arbitrarily large.

To generalize the previous result with this case, the separability constraints is relaxed for tolerating some misclassifications in the overlapping region. For each violation of the constraints will be punished with a misclassification penalty.

Cortes and Vapnik in 1995 gave the extension of linear SVM to the nonseparable case [30]. It is realized by introducing positive slack variable $\xi_k$ in the constraints. The inequalities are transformed to

$$y_k[w^T x_k + b] \geq 1 - \xi_k, k = 1, \ldots, N \quad (A.2.1)$$

The smallest slack variable satisfies the condition as follow.

$$\xi_k = \max\{0, 1 - y_k[w^T x_k + b]\} \quad (A.2.2)$$

It measures how many points fail to have a margin of $1/\|w\|$. The values of $\xi_k$ indicate where $x_k$ lies compared to the separating hyperplane.

1. $\xi_k \geq 1$: $y_k[w^T x_k + b] < 0$, misclassification;

2. $0 < \xi_k < 1$: $x_k$ is classified correctly, but lies inside the margin;

3. $\xi_k = 0$: $x_k$ is classified correctly, and lies outside the margin or exactly on the margin boundary.
In the primal weight space the optimization problem becomes

$$
\min_{w,b,\xi} J_P = \frac{1}{2} w^T w + c \sum_{k=1}^{N} \xi_k \tag{A.2.3}
$$

such that

$$
\begin{align*}
&\{ \ y_k[w^T x_k + b] \geq 1 - \xi_k, \ k = 1, \ldots, N \\
&\xi_k \geq 0, \ k = 1, \ldots, N
\end{align*} \tag{A.2.4}
$$

where $c$ is a positive real constant. Then the following Lagrangian is considered.

$$
L(w, b, \xi; \alpha, v) = J_P(w, \xi) - \sum_{k=1}^{N} \alpha_k (y_k[w^T x_k + b] - 1 + \xi_k) - \sum_{k=1}^{N} v_k \xi_k \tag{A.2.5}
$$

with Lagrange multipliers $\alpha_k \geq 0$, $v_k \geq 0$ for $k = 1, \ldots, N$. The second set of Lagrange multipliers is needed due to the additional slack variables $\xi_k$. The solution is given by the saddle point of the Lagrangian:

$$
\max_{\alpha,v} \min_{w,b,\xi} L(w, b, \xi; \alpha, v) \tag{A.2.6}
$$

which gives the dual QP problem after substituting (A.2.6) into (A.2.4) as follow.

$$
\max_{\alpha} J_D = \frac{1}{2} \sum_{k,l=1}^{N} y_k y_l x_k^T x_l \alpha_k \alpha_l + \sum_{k=1}^{N} \alpha_k \tag{A.2.7}
$$

such that

$$
\begin{align*}
&\sum_{k=1}^{N} \alpha_k y_k = 0 \\
&0 \leq \alpha_k \leq c, \ k = 1, \ldots, N
\end{align*} \tag{A.2.8}
$$

Comparing with the linearly separable case (A.2.1), this problem has additional box constraints.

### A.3 Kernel-based Classifiers

In the previous sections, it can be found that linear classifiers in the input space can easily be solved by standard optimization techniques. Although linear classifiers are easy to handle, they sometimes pose severe restrictions on the learning task. The target classification application may be too complex to be expressed as a linear combination of the given attributes. This problem can be overcome by an approach called kernel technique, which was introduced as the method of potential functions by Aizerman et al. [1]
Assumed $x \in D \subseteq \mathbb{R}^n$ denotes a real valued random input vector, $y \in \{-1, +1\}$ discrete real valued random output variable and $\Omega \subseteq \mathbb{R}^{n_H}$ a high dimensional feature space. The SVM method basically maps the input vector $x$ into the high dimensional feature space $\Omega$ through some nonlinear mapping $\varphi : D \rightarrow \Omega$. In this feature space, one consider the linear function

$$f(x) = \text{sign}[w^T \varphi(x) + b] \quad (A.3.1)$$

This linear function may generalize well in solving classification problems, however, it remains a problem to solve the calculation in the high dimensional feature space. Interestingly, no explicit construction of the nonlinear mapping $\varphi(x)$ is needed. This is motivated by the following result. The inner product in the feature space $\varphi(x_k)^T \varphi(x_l)$ can be replaced with the corresponding kernel $K(x_k, x_l)$ satisfying Mercer’s condition as follow [78].

Let $K \in L^2(C), g \in L^2(C)$ where $C$ is a compact subset of $\mathbb{R}^{d}$. To guarantee that a continuous symmetric function $K$ has an expansion

$$K(t, z) = \sum_{k=1}^{\infty} a_k \phi(t) \phi(z) \quad (A.3.2)$$

with positive coefficients $a_k > 0$, it is necessary and sufficient that the condition

$$\int_C \int_C K(t, z) g(t) g(z) dtdz \geq 0 \quad (A.3.3)$$

be valid for all $g \in L^2(C)$.

Suppose $w = \sum_{k=1}^{N} \alpha_k \varphi(x_k)$, and based on the Mercer theorem above, the linear function in the feature space (A.3.1) has the following equivalent representation in input space

$$f(x) = \text{sign}\left[\sum_{k=1}^{N} \alpha_k K(x, x_k) + b\right], b \in \mathbb{R}, \alpha_k \in \mathbb{R} \quad (A.3.4)$$

where $x_k$ are vectors and $K(x, x_k)$ is a given function satisfying Mercer’s condition. While in the primal space the parameter $w$ may have a range over an “infinite dimensional” parameter set, in the dual problem the dimension of the parameter vector $\alpha$ grows with the number of data points.

Using Mercer’s theorem to replace the inner product $\varphi(x_k)^T \varphi(x_l)$ with its corresponding kernel $K(x_k, x_l)$ is often called the kernel trick. It enables us to work in a huge dimensional feature space without actually having to do explicit computations in this space. Computations are done in another space after applying this kernel trick. In the case of support vector machines, one starts from a formulation in the primal weight space with a high dimensional feature space by applying
transformations \( \varphi(\cdot) \). The solution is calculated not in this primal weight space, but in the dual space of Lagrange multipliers after applying the kernel trick. In this way classification is done implicitly in a high dimensional feature space rather than in the original input space.

### A.4 Nonlinear SVM Classifiers

By combining the idea of an optimal separating hyperplane with a kernel induced mapping to a high dimensional feature space, the idea from linear is extended to nonlinear classifiers. One can formally replace \( x \) by \( \varphi(x) \) and apply the kernel trick where possible. However, notice that \( \varphi(x) \) can be infinite dimensional, and hence also the \( w \) vector. While for linear SVM one can in fact equally well solve the primal problem in \( w \) as the dual problem in the support values \( \alpha \), this is no longer the same for the nonlinear SVM case because in the primal problem the unknown \( w \) can be infinite dimensional.

With slight modification, the nonlinear case can be described as follow.

\[
\begin{align*}
w^T \varphi(x_k) + b & \geq +1 \quad \text{for} \quad y_k = +1 \\
\quad & (A.4.1) \\
\quad w^T \varphi(x_k) + b & \leq -1 \quad \text{for} \quad y_k = -1 \\
\quad & (A.4.2)
\end{align*}
\]

which is equivalent to

\[
y_k[w^T \varphi(x_k) + b] \geq 1, \ k = 1, \ldots, N \\
\quad & (A.4.3)
\]

in the case of separable data. The classification can be written as

\[
y_k = \text{sign}[w^T \varphi(x) + b] \\
\quad & (A.4.4)
\]

The optimization problem becomes

\[
\min_{w,b,\xi} \mathcal{J}_P = \frac{1}{2} w^T w + c \sum_{k=1}^{N} \xi_k \\
\quad \text{such that} \quad \left\{ \begin{array}{l}
y_k[w^T \varphi(x_k) + b] \geq 1 - \xi_k, \ k = 1, \ldots, N \\
\xi_k \geq 0, \ k = 1, \ldots, N
\end{array} \right. \\
\quad & (A.4.5)
\]

The Lagrangian is constructed:

\[
\mathcal{L}(w, b, \xi; \alpha, v) = \mathcal{J}_P(w, \xi) - \sum_{k=1}^{N} (\alpha_k y_k[w^T \varphi(x_k) + b] - 1 + \xi_k) - \sum_{k=1}^{N} v_k \xi_k \\
\quad & (A.4.6)
\]

with Lagrange multipliers \( \alpha_k \geq 0, v_k \geq 0 \) for \( k = 1, \ldots, N \). The solution is given by the saddle point of the Lagrangian:

\[
\max_{\alpha,v} \min_{w,b,\xi} \mathcal{L}(w, b, \xi; \alpha, v) \\
\quad & (A.4.7)\]
This leads to
\[
\begin{align*}
\frac{\partial L}{\partial w} &= 0 \rightarrow w = \sum_{k=1}^{N} \alpha_k y_k \varphi(x_k) \\
\frac{\partial L}{\partial b} &= 0 \rightarrow \sum_{k=1}^{N} \alpha_k y_k = 0 \\
\frac{\partial L}{\partial \xi_k} &= 0 \rightarrow 0 < \alpha_k < c, k = 1, \ldots, N
\end{align*}
\]  
(A.4.8)

The dual problem becomes
\[
\max_{\alpha} J_D(\alpha) = -\frac{1}{2} \sum_{k,l=1}^{N} y_k y_l K(x_k, x_l) \alpha_k \alpha_l + \sum_{k=1}^{N} \alpha_k
\]
(A.4.9)

such that
\[
\begin{align*}
\sum_{k=1}^{N} \alpha_k y_k &= 0 \\
0 &\leq \alpha_k \leq c, k = 1, \ldots, N
\end{align*}
\]

In this quadratic form, the kernel trick is applied
\[
K(x_k, x_l) = \varphi(x_k)^T \varphi(x_l)
\]
(A.4.10)

for \(k = 1, \ldots, N\). Finally the nonlinear SVM classifier takes the form
\[
y(x) = \text{sign} \left[ \sum_{k=1}^{N} \alpha_k y_k K(x, x_k) + b \right]
\]
(A.4.11)

with \(\alpha_k\) positive real constants which are the solution to a QP problem. The next problem is the determination of the constant \(b\). Karush-Kuhn-Tucker (KKT) complementarity conditions state that the product of the dual variables and the constraints should be zero at the optimal solution [43]. Therefore, using the KKT conditions, it yields
\[
\begin{align*}
\frac{\partial L}{\partial w} &= 0 \rightarrow w = \sum_{k=1}^{N} \alpha_k y_k \varphi(x_k) \\
\frac{\partial L}{\partial b} &= 0 \rightarrow \sum_{k=1}^{N} \alpha_k y_k = 0 \\
\frac{\partial L}{\partial \xi_k} &= 0 \rightarrow c - \alpha_k - v_k = 0 \\
\alpha_k \{y_k [w^T \varphi(x_k) - 1 + \xi_k]\} &= 0, k = 1, \ldots, N \\
v_k \xi_k &= 0, k = 1, \ldots, N \\
\alpha_k &\geq 0, k = 1, \ldots, N \\
v_k &\geq 0, k = 1, \ldots, N
\end{align*}
\]  
(A.4.12)

From \(v_k \xi_k = 0\), the solutions \(w^*, b^*, \xi^*, \alpha^*, v^*\) to this problem is that \(\xi_k^* = 0\) for \(\alpha_k^* \in (0, c)\). Hence
\[
y_k [w^T \varphi(x_k) + b] - 1 = 0 \quad \text{for} \quad \alpha_k \in (0, c)
\]  
(A.4.13)

which means that one can take any training data point for which \(0 \leq \alpha_k \leq c\) and use that equation to compute the bias term \(b\).

There are some properties of the nonlinear SVM classifier and its solution as follows [107].
1. Kernel functions:

Four common choices of kernels are

(a) $K(x, z) = x^T z$ (linear SVM))
(b) $K(x, z) = (\tau + x^T z)^d$ (polynomial SVM of degree $d$)
(c) $K(x, z) = \exp(-\|x - z\|^2 / \sigma^2)$ (RBF kernel)
(d) $K(x, z) = \tanh(\kappa_1 x^T z + \kappa_2)$ (Multilayer Perceptron/MLP)

The Mercer condition holds for all $\sigma$ values in the RBF kernel case and $\tau$ values in the polynomial case but not for all possible choices of $\kappa_1, \kappa_2$ in the MLP case.

2. Global and unique solution:

As in the linear SVM case the solution to the convex QP problem is again global and unique provided that one chooses a positive definite kernel for $K(\cdot, \cdot)$. This choice guarantees that the matrix involved in the QP problem is positive definite as well, and the kernel trick is applicable. For a positive semidefinite kernel the solution to the QP problem is global but not necessarily unique.

3. Sparseness:

As for the linear SVM classifier case many $\alpha_k$ values are equal to zero in the solution vector. In the dual space the nonlinear SVM classifier takes the form

$$y(x) = \text{sign} \left[ \sum_{k=1}^{\#SV} \alpha_k y_k K(x, x_k) + b \right] \quad (A.4.14)$$

where the sum is taken over the non-zero $\alpha_k$ values which correspond to support vectors $x_k$ of the training data set.

4. Geometric meaning of support vectors:

The support vectors obtained from the QP problem are located close to the nonlinear decision boundary.

5. Non-parametric/parametric issues:

Both the primal and the dual problem have neural network representation interpretations. The problem in the primal weight space is parametric, while the dual is non-parametric. Note that in the dual problem the size of the QP problem is not influenced by the dimension $n$ of the input space.
A.5 Conclusions

Support Vector Machines (SVM) is a method of calculating the optimal separating hyperplane in the feature space. Optimal separating hyperplane is defined as the maximum-margin hyperplane in the higher dimensional feature space.

The use of the maximum-margin hyperplane is motivated by statistical learning theory, which provides a probabilistic test error bound which is minimized when the margin is maximized.

The parameters of the maximum-margin hyperplane are derived by solving a quadratic programming (QP) optimization problem. There exists several specialized algorithms for quickly solving the QP problem that arises from SVMs.

Kernel trick is used for solving the nonlinear classification. In the kernel trick, each dot product used in a linear algorithm is replaced with a non-linear kernel function. This causes the linear algorithm to operate in a different space. For SVMs, using the kernel trick makes the maximum margin hyperplane be fit in a feature space. The feature space is a non-linear map from the original input space, usually of much higher dimensionality than the original input space. In this way, non-linear SVMs can be created.
Appendix B

Estimation of Distribution Algorithms (EDAs)

This appendix presents a brief introduction of the Estimation of Distribution Algorithms (EDAs) and two typical EDA methods.

B.1 Introduction

In EDAs [67], there are neither crossover nor mutation operators. Instead, the new population is sampled from a probability distribution, which is estimated from a database that contains the selected individuals from the previous generation. Thus, the interrelations between the different variables that represent the individuals are explicitly expressed through the joint probability distribution associated with the individuals selected at each generation.

Figure B.1 illustrates the flow chart for an EDA approach. Initially, a random solutions is generated. These solutions are evaluated using an objective function. An objective function evaluates how accurate each solution is for the problem. Based on this evaluation, a subset of solutions is selected. Hence, solutions with better function values have a bigger chance of being selected. Then, a probabilistic model of the selected solutions is built, and a new set of solutions is sampled from the model. The process is iterated until the optimum has been found or another termination criterion is fulfilled [4].

In order to explain the behavior of this heuristic, a common outline for EDAs is listed as follow.

1) Generate the first population of $M$ individuals and evaluate each of them. Usually, this generation is made assuming a uniform distribution on each variable.

2) $N$ individuals are selected from the set of $M$, following a given selection method.
3) An $l$ (size of the individual) dimensional probabilistic model that shows the interdependencies among the variables is induced from the $N$ selected individuals.

4) Finally, a new population of individuals is generated based on the sampling of the probability distribution learnt in the previous step.

5) Steps of 2 to 4 are repeated until some stop criterion is met (e.g. a maximum number of generations, a homogeneous population, or no improvement after a certain number of generations).

Essentially EDAs assume that it is possible to build a model of the promising areas of the search space, and use this model to guide the search for the optimum. In EDAs, modeling is achieved by building a probabilistic graphical model that represents a condensed representation of the features shared by the selected solutions. Such a model can capture different patterns of interactions between subsets of the problem variables, and can conveniently use this knowledge to sample new solutions.

Probabilistic modeling gives EDAs an advantage over other evolutionary algorithms that do not employ models, such as GAs. These algorithms are generally unable to deal with problems where there are important interactions among the problems’ components. This, together with EDAs’ capacity to solve different types of problems in a robust and scalable manner, has led to EDAs sometimes also being referred to as competent GAs [47, 86]. EDAs can be seen as a development
of GAs. By recombining a subset of selected solutions, GAs are able to process the information learned during the search, and to orient the exploration to promising areas of the search space. Nevertheless, it has been proved that GAs experience limitations in their capacity to deal with problems where there are complex interactions between different components of the solutions. In these scenarios, EDAs can exhibit a better performance [44, 77, 85].

EDAs can be broadly divided according to the complexity of the probabilistic models used to capture the interdependencies between the variables: univariate, bivariate and multivariate approaches [4]. Univariate EDAs, such as PBIL [7], cGA [54] and UMDA [82], assume that all variables are independent and factorize the joint probability of the selected solutions as a product of univariate marginal probabilities. Consequently, these algorithms are the simplest EDAs and have also been applied to problems with continuous representation.

The bivariate models can represent low order dependencies between the variables and be learnt using fast algorithms. MIMIC [61], the bivariate marginal distribution algorithm BMDA [87], dependency tree-based EDAs [8] and the tree-based estimation of distribution algorithm (Tree-EDA) [97] are all members of this subclass. The latter two use tree and forest-based factorizations, respectively. They are recommended for problems with a high cardinality of the variables and where interactions are known to play an important role. Trees and forests can also be combined to represent higher-order interactions using models based on mixtures of distributions.

Multivariate EDAs factorize the joint probability distribution using statistics of order greater than two. As the number of dependencies among the variables is higher than in the above categories, the complexity of the probabilistic structure, as well as the computational effort required to find the structure that best suits the selected solutions, is greater. Therefore, these approaches require a more complex learning process. Some of the popular Multivariate EDAs are the Factorized Distribution Algorithm (FDA) [81], the Bayesian optimization algorithm (BOA) [87] and the extended compact Genetic Algorithm (EcGA) [54].

Since several EDAs have been proposed with a variety of models and learning algorithms, the selection of the best EDA to deal with a given optimization problem is not always straightforward. One criterion that could be followed in this choice is to trade off the complexity of the probabilistic model against the computational cost of storing and learning the selected model. Both issues are also related to the problem dimensionality (i.e. number of variables) and to the type of representation (e.g. discrete, continuous, mixed).

The simple models generally have minimal storage requirements, and are easy to learn. However, they have a limited capacity to represent higher order interactions. On the other hand, more
complex models, which are able to represent more involved relationships, may require sophisticated
data structures and costly learning algorithms. The impact that the choice between simple and more
complex models has in the search efficiency will depend on the addressed optimization problem. In
some cases, a simple model can help to reach non-optimal but acceptable solutions in a short time.
In other situations, e.g. deceptive problems, an EDA that uses a simple model could move the search
away from the area of promising solutions. Another criterion that should be taken into considera-
tion to choose an EDA is whether there is any previous knowledge about the problem structure, and
which kind of probabilistic model is best suited to represent this knowledge [4].

B.2 Univariate Marginal Distribution Algorithm (UMDA)

Univariate Marginal Distribution Algorithm (UMDA) is a computing model proposed by MUhlen-
bein in 1998 [82]. Since variable domain can be discrete and continuous, there exist two kinds of
UMDAs [67]. One is in discrete domain, the other in continuous domain [81]. Both of them sup-
pose that variables are independent. The discrete one estimates the joint probability distribution of
the selected individuals very simply. It is factorized as a product of independent univariate marginal
distributions. That is:

\[
p_t(x) = p(x|D_{t-1}^{sel}) = \prod_{i=1}^{n} p_t(x_i)
\]  

(B.2.1)

Each univariate marginal distribution is estimated from marginal frequencies:

\[
p_t(x_i) = \frac{\sum_{j=1}^{n} \delta_j(X_i = x_i|D_{t-1}^{sel})}{N}
\]

\[
\delta_j(X_i = x_i|D_{t-1}^{sel}) = \begin{cases} 
1 & \text{if in the } j \text{-th case of } D_{t-1}^{sel}, X_i = x_i \\
0 & \text{otherwise}
\end{cases} 
\]

(B.2.2)

where \(N\) is the size of subpopulation, \(D_{t-1}^{sel}\) the subpopulation composed by the best \(N\) individuals
in \(t\)-th generation, \(p_t(x)\) the probability of individual \(x \in (x_1, x_2, \ldots, x_n)\) existing in \(t\)-th generation, \(x_i\) the value of \(i\)-th variable in individual \(x\).

UMDA in continuous domain uses the density function to generate new populations. It is factor-
ized by a product of unidimensional and independent normal densities.

\[
f_t(x, \mu, \sigma) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{1}{2} \left(\frac{x_i - \mu_i}{\sigma_i}\right)^2}
\]

\[
\mu^t = (\mu_1^t, \mu_2^t, \ldots, \mu_n^t)
\]

\[
\sigma^t = (\sigma_1^t, \sigma_2^t, \ldots, \sigma_n^t)
\]  

(B.2.3)
where $\mu^t_i$ and $\sigma^t_i$ are estimated from the selected subpopulation. In order to find the density function to best fits the selected subpopulation, maximum likelihood estimates are adopted.

### B.3 Factorized Distribution Algorithm (FDA)

Factorized Distribution Algorithm (FDA), as was introduced in [81], is an evolutionary algorithm which combines mutation and recombination by using a distribution instead. First the distribution is estimated from a set of selected points. It is then used to generate new points for the next generation. The FDA has been extensively investigated for the optimization of Additively Decomposed Functions (ADFs). It was shown theoretically and numerically that the scaling of the FDA depends on the ADF structure and the specific assignment of function values. Difficult functions on a chain or a tree structure are optimized in about $O(n\sqrt{n})$ function evaluations. More standard genetic algorithms are not able to optimize these functions.

First an ADF is defined precisely as follows.

Definition: An additively decomposed function (ADF) is defined by

$$f(x) = \sum_{s_i \in S} f_i(\prod_{s_i} x), S = \{s_1, \ldots, s_l\}, s_i \subseteq \tilde{X}$$  \hspace{1cm} (B.3.1)

where

$\tilde{X} = x_1, \ldots, x_n$; $B = 0, 1$; $X = B^\dagger \tilde{X}$

$X_S \subseteq X$ with $s \subseteq \tilde{X}$

$f_i(\prod_{s_i} x) =$ the projection of $x \in X$ onto the subspace $X_s$.

FDA can be used with an exact or an approximate factorization. It uses finite samples of points. Convergence of FDA to the optimum will depend on the size of the samples. FDA can be run with any popular selection method. The truncation selection is usually applied. The FDA algorithm introduced in Ref. [81] is listed as follows.

1) Set $t \leftarrow 0$. Generate $(1 - r) \ast N \gg 0$ points randomly and $r \ast N$ points according to a local approximation of the conditional marginal distributions.

2) Selection of promising points.

3) Compute the conditional probabilities $p^{*}(\prod b_i x, \prod c_i x, t)$.

4) If the termination criteria are met, FINISH.

5) Add the best point of the previous generation to the generated points (elitist).
6) Set $t := t + 1$. Go to STEP 1.

B.4 Conclusions

EDAs are a novel class of evolutionary optimization algorithms that were developed as a natural alternative to genetic algorithms in the last decade. The principal advantages of EDAs over genetic algorithms are the absence of multiple parameters to be tuned (e.g. crossover and mutation probabilities) and the expressiveness and transparency of the probabilistic model that guides the search process. In addition, EDAs have been proven to be better suited to some applications than GAs, while achieving competitive and robust results in the majority of tackled problems.
Appendix C

Affinity Propagation Clustering

This appendix introduces a brief introduction of the Affinity Propagation (AP) clustering. Affinity propagation [45] is a recently introduced clustering method which takes as input a set of measures of similarity between pairs of data points and outputs a set of clusters of the points with their corresponding exemplars.

C.1 Introduction

Clustering data by identifying a subset of representative examples is important for detecting patterns in data and in processing sensory signals. Clustering or discovering meaningful partitions of data based on a measure of similarity is a critical step in scientific data analysis and a fundamental problem in computer science. A common approach within the machine learning community involves unsupervised learning of parameters that describe clusters (e.g. the location and scale/shape of the cluster) and partitioning the data by associating every point or region with one or more clusters. In many situations, data is better and more easily characterized by a measure of pairwise similarities rather than defaulting to negative squared Euclidean distance, and in this case, clusters can instead be represented by an “exemplar” data point rather than domain-specific parameters [39, 53, 58].

The AP clustering uses belief propagation methods to achieve outstanding results for exemplar-based clustering. Identifying exemplars is advantageous because user-specified similarities offer a large amount of flexibility and allow the clustering algorithm to be decoupled from the details of how similarities between data points are computed. Unlike many algorithms that operate in vector space, there is no need for similarity to be based on squared Euclidean distance, or for the data space to be metric or continuous, or even ordinal. Additionally, there is potential for significant improvement on existing algorithms, both in terms of solution time and solution quality.
Affinity propagation takes as input a set of pairwise similarities between data points and finds clusters on the basis of maximizing the total similarity between data points and their exemplars. Similarity can be simply defined as negative squared Euclidean distance for compatibility with other algorithms, or it can incorporate richer domain-specific models (e.g., translation-invariant distances for comparing images). Affinity propagations computational and memory requirements scale linearly with the number of similarities input; for non-sparse problems where all possible similarities are computed, these requirements scale quadratically with the number of data points. Affinity propagation is demonstrated on several applications from areas such as computer vision and bioinformatics, and it typically finds better clustering solutions than other methods in less time.

C.2 Affinity Propagation Algorithm

The task of exemplar-based clustering is to identify a subset of the \( N \) data points as exemplars and assign every other data point to one of those exemplars. The only inputs are a set of real-valued pairwise similarities between data points, \( s(i, k) \), and the number of exemplars to find (\( K \)) or a real-valued exemplar cost to balance against similarities. A simple and fast algorithm for finding clustering solutions is the k-medoids algorithm [73], which begins by randomly selecting a set of \( K \) data points as initial exemplars and then refines these in alternating steps. The algorithm monotonically maximizes the sum of similarities between data points and exemplars but considers only a fixed set of exemplars, and thus is quite sensitive to the initial selection of exemplars. For this reason, k-medoids clustering needs to be run with many different random initializations—it works well only when the number of clusters is small and chances are good that at least one restart lies close to a good clustering solution.

In contrast to k-medoids, affinity propagation simultaneously considers all data points as potential exemplars. By viewing each data point in a network, it recursively transmits real-valued messages along edges of the network until a good set of exemplars and corresponding clusters emerge; see Fig.C.1 for an illustration of these dynamics. Affinity propagation sends two types of message between data points: responsibilities are sent from data points to candidate exemplars and reflect the evidence of how well-suited the message-receiving point is to serve as an exemplar for the sending point. Availabilitys are sent from candidate exemplars to data points and reflect the evidence for how appropriate it would be for the message-sending point to be the exemplar for the message-receiving point (see Fig.C.1). All data points can be considered to be either cluster members or candidate exemplars, depending on whether they are sending or receiving availability.
or responsibility messages.

The algorithm takes as input a matrix of similarity measures between each pair of points \( s(i, k) \). Instead of requiring that the number of clusters be predetermined, affinity propagation takes as input a real number \( s(k, k) \) for each data point \( k \). These values, which are called preferences, are a measure of how likely each point is to be chosen as exemplar.

The algorithm works by exchanging messages between the points until a stop condition is satisfied. There are two types of messages to be exchanged between data points. The responsibility \( r(i, k) \), sent from data point \( i \) to candidate exemplar point \( k \), reflects the accumulated evidence for how well-suited point \( k \) is to serve as the exemplar for point \( i \), taking into account other potential exemplars for point \( i \). The availability \( a(i, k) \), sent from candidate exemplar point \( k \) to point \( i \), reflects the accumulated evidence for how appropriate it would be for point \( i \) to choose point \( k \) as its exemplar, taking into account the support from other points that point \( k \) should be an exemplar.

In the availability update shown in Eq. C.2.2 only the positive portions of incoming responsibilities are added, because it is only necessary for a good exemplar to explain some data points (positive responsibilities), regardless of how poorly it explains points with negative responsibilities.

This message reflects accumulated evidence that point \( k \) is an exemplar, based on the positive responsibilities sent to candidate exemplar \( k \) from other points. For a point \( i \), the value of \( k \) that maximizes \( a(i, k) + r(i, k) \) either identifies point \( i \) as an exemplar if \( k = i \), or identifies the data point that is the exemplar for point \( i \).

1) **INPUT:** a set of pairwise similarities, \( \{s(i, k)\}_{(i,k) \in \{1,\ldots,N\}^2, i \neq k} \) where \( s(i, k) \in R \) indicates
how well-suited data point \( k \) is as an exemplar for data point \( i \).

e.g., \( s(i, k) = -\|x_i - x_k\|^2, i \neq k \) (squared Euclidean distance)

For each data point \( k \), a real number \( s(k, k) \) indicating the \textit{a priori} preference (negative cost of adding a cluster) that it be chosen as an exemplar.

e.g., \( s(k, k) = p, \forall k \in \{1, \ldots, N\} \) (global preference)

2) \textbf{INITIALIZATION}: set availabilities to zero, \( \forall i, k : a(i, k) = 0 \).

3) \textbf{REPEAT}: responsibility and availability updates until convergence.

\[
\begin{align*}
    r(i, k) & \leftarrow s(i, k) - \max_{k' \neq k} \{a(i, k') + s(i, k')\} \quad \text{(C.2.1)} \\
    a(i, k) & \leftarrow \min \left\{ 0, r(k, k) + \sum_{i' \neq \{i, k\}} \max \{0, r(i', k)\} \right\} \quad \text{(C.2.2)} \\
    a(k, k) & \leftarrow \sum_{i' \neq \{i, k\}} \max \{0, r(i', k)\} \quad \text{(C.2.3)}
\end{align*}
\]

4) \textbf{OUTPUT}: assignments \( \hat{c} = \hat{c}_1, \ldots, \hat{c}_N \), where \( \hat{c}_i = \arg\max_k [a(i, k) + r(i, k)] \) and \( \hat{c}_i \) indexes the cluster’s exemplar to which point \( i \) is assigned. Specifically, if point \( i \) is in a cluster with point \( k \) serving as the exemplar, then \( \hat{c}_i = k \) and \( \hat{c}_k = k \). Note: one run of k-medoids may be needed to resolve contradictory solutions.

\section*{C.3 Conclusions}

Affinity propagation achieves outstanding results by employing “loopy belief propagation” techniques that have previously been used to approach Shannon’s limit in error-correcting decoding and solve random satisfiability problems with an order-of-magnitude increase in size [80]. The objective function it maximizes is the net similarity, \( S \), which is the sum of the similarities of non-exemplar data points to their exemplars plus the sum of exemplar preferences (negative costs of adding exemplars).

The Affinity propagation method has been praised because of its ability to efficiently and quickly handle very large problems.
Publication List


Bibliography


