CSC 3002
Introduction to Computer Science: Programming Paradigms

Project Report

Project Title:
Peptide Chain Classifier
Based on
Constitutent Amino Acid Analysis

Group Member:

Lirong Wang	116010218
Binbin Huang	116010084
Dinglin Xia	116010240
Minjue Zhang	116010293
Haoyu Wu	116010236
Nuoyao Yang	115020295

May 19, 2019

Contents

Chapter 1

What We Have Done

1.1 Program Introduction

We build a software to binary classify positive and negative peptides in order to predict whether a peptide has certain function like resisting virus, bacteria, fungi, or so on. The main process is similar with the previous proposals, however, we changed the main models we use for classification: from decision tree and logistic regression to support vector machine, k-nearest-neighbors and Naive Bayes.

First we extract features from raw peptides to get Amino Acid Composition, Composition of k-spaced amino acid pairs and AAC of N and C terminals. Next, we use outlier analysis, redundancy and Principal Component Analysis to reduce dimensions of raw features and provide the new data to the classification models. Then, different classification can be applied and use cross validation to obtain parameters including accuracy, error rate, sensitivity and specificity in order to compare precisions of different models. At last, users can choose the model with the highest accuracy to predict unknown peptides.

1.2 Member Duty

- **Haoyu Wu**: Overall Structure Design, UI design, Implementation of SVM algorithm, Data visualization and its integration with UI, Integrating Machine Learning Functions with UI, peptide prediction for new imported data
- **Lirong Wang**:Import peptides, feature extraction (AAC, CKSAAP, N5C5 AAC) and feature visualization (Quick sort $|pos\_cksaap - neg\_cksaap|$), Overview of all-model performance.
- **Mingjue Zhang**: Data Clustering, Principal Component Analysis
- **Dinglin Xia**:Model Evaluation, Cross Validation, Algorithm implementation (Nave Bayes and KNN), best parameter selection algorithm
- **Binbin Huang**: Implementation of Decision Tree Algorithm
- **Nuoyao Yang**: Logistic Regression

1.3 Difference With Proposal

1.3.1 What We Add

- **3 Classification Algorithms**: Naïve Bayes, KNN and SVM
- **Best Parameter Selection**: Including selecting the best threshold for Logistic Regression and SVM, best K for KNN method.

1.3.2 What We Delete

- **Outlier Detecting**

Chapter 2

Whole Process: How Do We Achieve Protein Classification

2.1 Overall Program Structure

- Each step is supplied with visualization.
- Each step is encapsulated as an object and then integrated together. We also use a standard encapsulation procedure to encapsulate each visualization process.
- In the early days, we provide a test case for each model to test and construct. In the end, we integrate everything together.

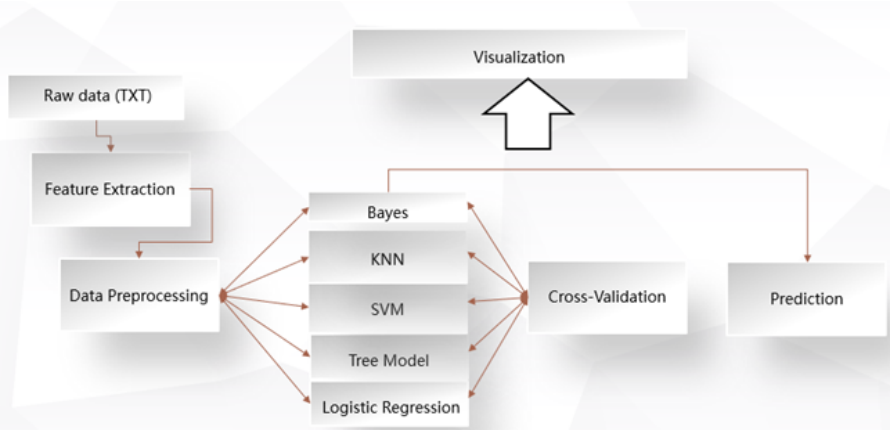
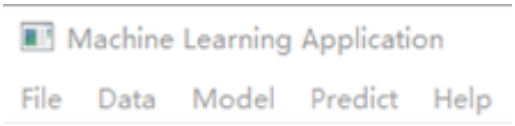


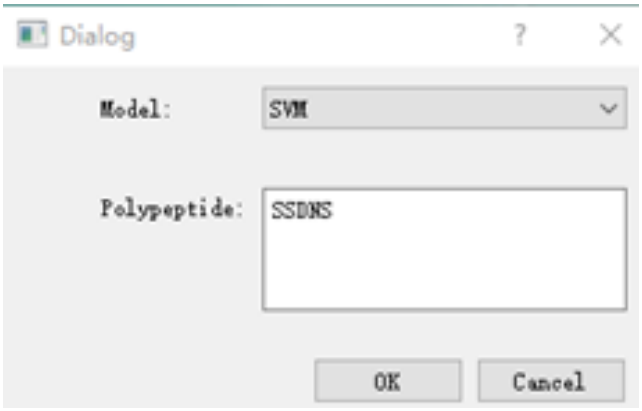
Figure 2.1: The Overall Structure

2.2 Console Design

- Although the mechanism inside is complex, we design the interface to be user-friendly and simple to use, like below:



. File: read the raw data through the file
. Data: Feature Extraction and Data Preprocessing
. Model: Tree Model, SVM, Logistic Regression, Naive Bayes, KNN
. Predict: Model Evaluation will show different parameters to compare precession of different models for users to choose. The prediction part uses Emit, Signal, Slot mechanism to exchange data between the main window and the sub window.



. Help: If you still have questions, we have written a user manual for you. You can click help to see it.

2.3 Feature Extraction

After getting data in proper format, the program will store all class labels separately and calculate Amino Acid Component by:

```
for (int i = 0; i < size; i++) {
    for (int j = 0; j < 20; j++) {
    if (peptide[i] == NATURAL_AA[j]) {
        aac[j]++;
        break;
        }
    }
}
for (int k = 0; k < 20; k++) {
aac[k] = aac[k]/size;
};
```

To get CKSAAP, first we need to put all the amino acid pairs into a new vector, and then compute average component of each kind of aa pair. To get N5C5 AAC, things got a little bit complicated since we need to deal with the peptides which are shorter than ten. So first, we

need to add X into the peptide string by while loop until the length of string reaches ten, and later calculate in the same way as calculating AAC.

To visualize features, both AAC and N5C5 AAC will be shown in barchart to compare the features of positive data and negative data. Sample graph is shown in figure 1.

Figure 2.2: Barchart to compare the features of data

While for the visualization of CKSAAP, since the dimension is high (800 if we want to shown the comparison of two data sets), we need to combine both features of positive and negative data into one new feature ($|pos\_cksaap - neg\_cksaap|$) and to sort to find aa pairs with highest results. Sample graph is shown below,

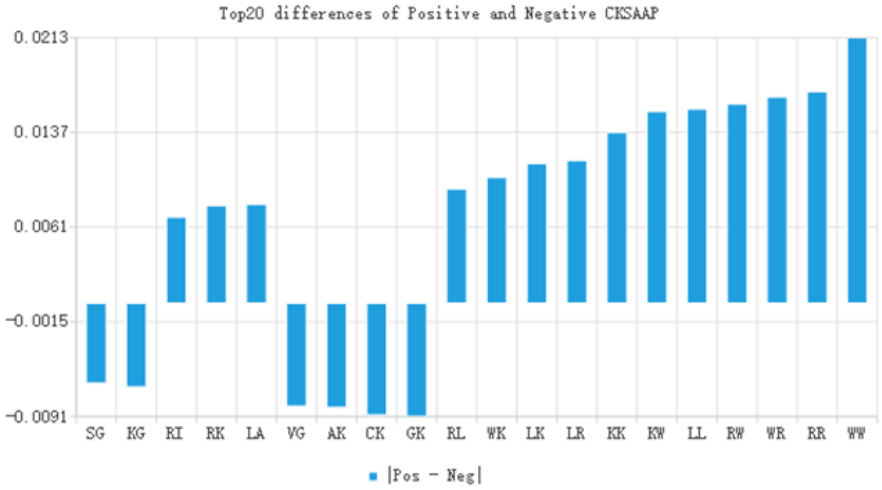


Figure 2.3: Sample graph of new features

Details will be shown in the third part of the report.

2.4 Data Preprocessing

Since the raw data we get is of high dimension, it takes much time to run the result. Also, many value in the raw data are zero, it is reasonable to think that some variables may not be useful and some may have correlation, which will cost the running time and occupy much space. In order to clean the data to have better results, we do outliers analysis, redundancy elimination and PCA in order.

Although do PCA first may reduce the running time in the other parts, it will also weaken the efficiency in performing other parts. Therefore, we identify outliers and redundancy on the raw data, before reducing the dimension.

Outlier analysis is based on clustering. This process includes two parts: divide the data into two clusters(since we have two results) and find out the local outliers in each cluster. We make a template class for cluster and set parameters we need and functions in it,

```
template <typename T>
class KMEANS{
public:
    KMEANS<T>(int k);

    vector< vector<T> > data;
    int k;
    int sample_size;
    int components_num;
    struct Cluster{
        vector<double> centroid;
        vector<int> samples;
    };

    void k_means(vector< vector<T> > data, int k, int max_intera);

    double EcludDist(vector<T> & v1, vector<T> & v2);
};
```

Figure 2.4: Template class for cluster and set parameters

Here, we make Cluster a struct because one cluster mainly includes two parts, one is a centroid point, the other is the samples in it. Every element in the vector centroid is the feature of the centroid point. For vector samples, we do not put every feature of all the sample points in it, instead, we put the ID of the point that it belongs to in the data. The function k_means is the main step in the process, which includes the two parts. In the first part, we choose k-means method, and the iteration is as follows,

- Choose k centroid points randomly;
- Decide which cluster the sample points belong to according to Ecludian distance: belong to the nearest centroid point;
- Renew the centroid points by the points that are in the same cluster: their mean;
- Iterate the (2) and (3) until the centroid points do not change(do it for max_itera times);

In the second part, we use the criteria that if the distance between centroid and observation is larger than threshold, it is an outlier. Therefore, the main object is to find out the threshold.

After finding outliers, we need to delete them in the data, we use the erase function in the vector class.

In the elimination of redundancy, we have two functions since we use Pearson Correlation Coefficients to test the whether two variables are correlated.

```
void cal_pearson(vector< vector<T> > & pearsonCorr, const vector< vector<T> > &cov);
void t_test(vector< vector<double> > & r,vector <vector<double> > & data );
```

Figure 2.5: Correlation Coefficient Calculation

After cleaning the raw data, we do PCA to reduce the dimension in order to do the visualization and improve the efficiency. First, we set a class to contain all the parameters and function we need in PCA.

```
template <typename T>
class PCA {
public:
    PCA() = default;
    int load_data(vector< vector<T> >& data, vector<T>& labels);
    vector< vector<T> > output(vector< vector<T> >& data);

private:
    int eigen(vector< vector<T> >& covar, bool sort_ = true, int max_iter = 100, double threshold = 0.05);
    void cal_covmatrix(vector< vector<T> >& cov, bool scale = false);

    vector< vector<T> > data;
    vector<T> labels;
    int sample_size = 0;
    int feature_num = 0;
    vector< T > eigen_values;
    vector< vector<T> > eigen_vectors;

};
```

Figure 2.6: A class containing all the parameters and function we need in PCA

The output function is what we get after PCA, in which we can also choose the number of components we need in this process. We get output using matrix multiplication: output = data * eigen_vectors.

In the whole preprocess process, we should realize that we are dealing with one data set, but in outliers identify and PCA, we construct two classes, which means that we need to set two object respectively. However, we only one object. Therefore, we use inheritance. We construct a new class KMEANS_PCA that inherited from both KMEANS and PCA.

```
class KMEANS_PCA: public PCA, public KMEANS<double>{
public:
    KMEANS_PCA(int k)
        : PCA(), KMEANS<double>(k) {

    }
};
```

After the above three steps, we get the data that can be used in further analysis. Here, we need to realize that, the above process can be all used only in training. When testing a animo chain, we use PCA only since cluster and redundancy can not be done based only one observation.

2.4.1 PCA Visualization

We use 3D scatter plots to show the first three features (explained most of the variance) of the preprocessed data. Unfortunately, we cannot show 3D particles in 2D pictures very well. So, we provide three perspectives for users to analysis. From this plot, we can easily find outliers. Also, we can zoom in to have a better view at the 3D graphs. Camera Presented Front Low:

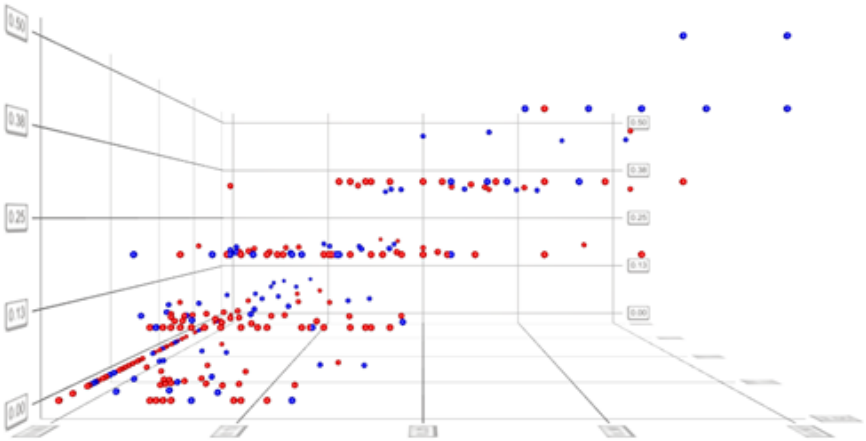


Figure 2.7: Visualization Of PCA

2.5 Algorithm

2.5.1 Decision Tree (ID3) Algorithm

In the past section, we finished the part of the data pre-processing and now we need to train those data and get a model and use the model to get a prediction based on the latter input. The first model of our project is the decision tree model. In this model, we need three basic components: features, labels, and results. Since the labels of our input are of continuous types that means they are numerical values, we need to do some tricky methods based on ID3 algorithm. We divide this part into 4 sections:

- Discretization
- Build a tree model
- Pruning
- Prediction

In our project, we use several functions in the interface:
CDTree(): constructor function.
buildTree():is the interface to build the tree, there are two implementations, one is the construction of the validation set, the other is the construction without the validation set
predict: is the predict function. The input and output data of this class is stored through the Eigen structure First we need to convert the format of the input.

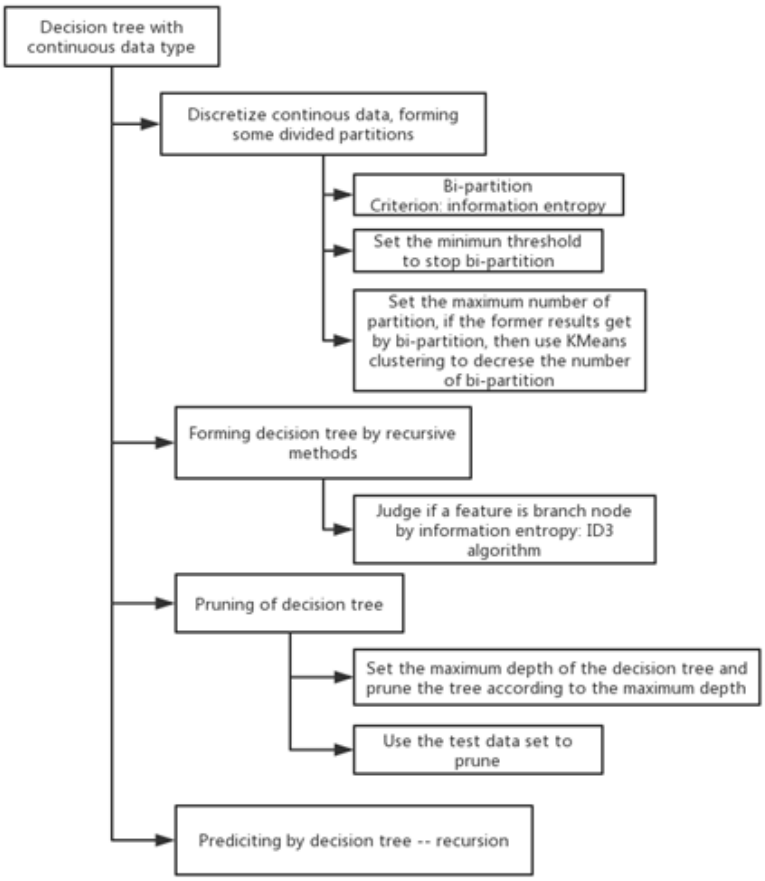


Figure 2.8: The Layout of Whole ID3 Algorithm Process

Visulization

This step will visualize the decision tree build by algorithm, which will be more clearly for user to know the classification process and results by the model.

2.5.2 Logistic Regression

The realization of the model is based on 6 modules, and the core module of the model is optimization. In this module, the functions for calculating the cost function and the the gradient cost function are given. We also employ ¡cmath¿ library to logarithm the variables, so that we can transform the optimization problem into a convex form, and the first-order derivative functions, namely, gradient descent function, convergences as the iterations increases. The iteration and convergence check are formulated into the fmin2 function, and one assisting function for updating parameters in each loop is also given. Notice that the we calculate the cost function for each set of observations and their running average value not for the showing to the audience, but to check the convergence. We once coincide with the non-convergence problem; and it will be explained later in the difficulty part.

When it comes to providing the prediction result for the clients, we formulate the module

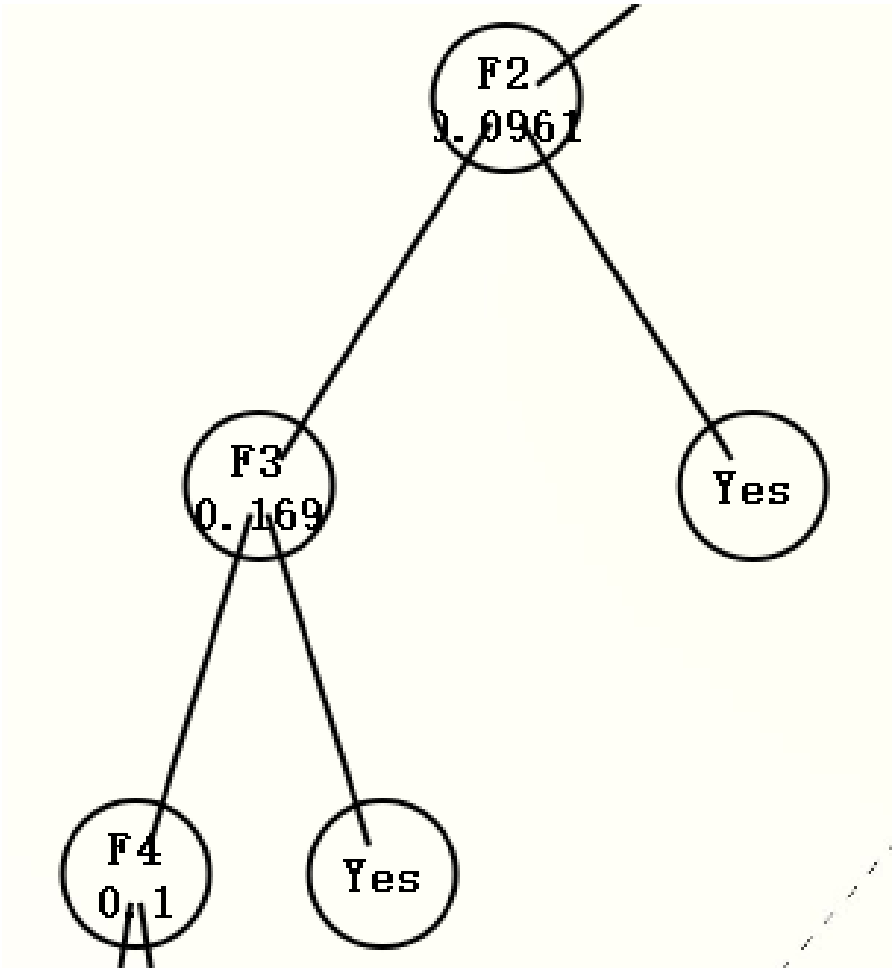


Figure 2.9: Visualization of Tree Model

predict. In this module, we will simply combine the input data from the client with a set of trained parameters gotten from training data, and give the result. The accuracy function is an assisting function that gives the accuracy given a specific value of the threshold; since we havent scaled the variables, we may need a threshold that maximizes the accuracy. The detailed implementation will be given in the cross-validation part.

In the module sigmoid, we construct some fundamental calculation functions. For example, we simulate the sigmoid function and the vector dot multiplication for the following usage.

After developing the model using the vector¡double¿ and vector¡vector¡double¿¿ for the most part, we discovering that the implementation of the algorithm may take some time of approximately 30 seconds. Hence, some other data structures may be tried to enhance the efficiency. DoubleArray is then created and tested, and a segmentation module is used to assist the calculations. However, the result seems to be unsatisfactory; the time complexity does not reduce significantly with the volume of our data.

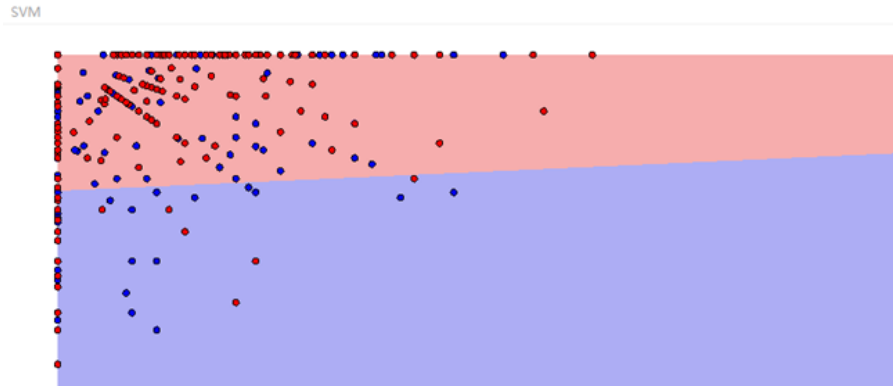
In the model, the parameters will be trained and sorted, so a priority will be given as the visualization. It may help to detect the importance of different features in the peptide chain, and provides some hints for the purpose of future study.

2.5.3 SVM

a. After preprocessing, to better understand the classification process and avoid the overfitting problem, we choose the two features that explain most of the variance after PCA.

b. We use the Sequential Minimal Optimization and the linear kernel to train the SVM model.

c. For each epoch, we can visualize the data like below:



2.5.4 Cross Validation

Cross Validation process are integrated with each classification method so that utilization of training data are maximized. For each training method, training data are equally divided into k parts (k is defaulted as 12). (k-1) of them are selected as training data, 1 of them are selected testing data. After the model trained, testing data will be input trained model to evaluate the model (output training accuracy). Besides, it helped selects the best parameter for KNN methods and Logistic Regression method.

- **KNN**: The algorithm observe the nearest k objects to our target protein and find the highest vote, choose it as the label of target protein. The number k is initialized as 7, however it may not be the best value. Therefore, we let the user input max k, and our program will test the accuracy of the model when k ranges from 2 to k. For example in this

- **Logistic Regression**: The basic thought behind this method is to find a function to represent the possibility that the protein is positive (noted as Y = 1). And the protein will be noted as positive only if:

$$P(Y = 1|X) > threshold$$

  Here threshold will is a parameter. So cross validation will be applied to a series of models with different parameters and their accuracy. Finally, it can help select the best parameter automatically (one with highest accuracy).

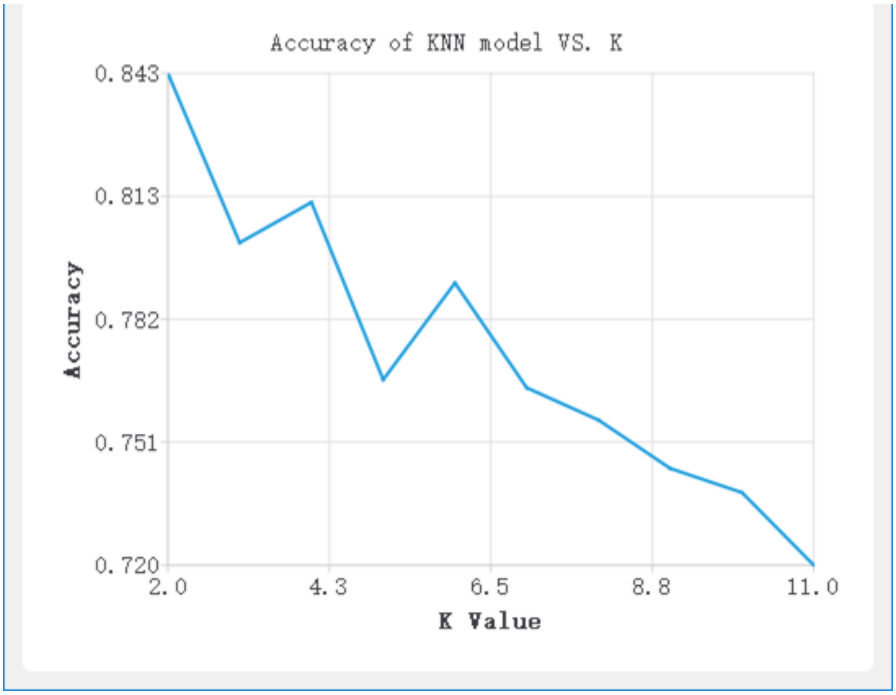


Figure 2.10: Plotting of accuracy of KNN model VS. K

Chapter 3

Our Facing Difficulties and Solutions

3.1 Feature Extraction

In the data extraction part, one of the difficulties is to display high dimensional features (800 in total) into one histogram and able to compare positive and negative datasets at the same time. CKSAAP is a feature which reflect the percentages of each kind of amino acid pairs. In order to display differences between CKSAAP of positive and negative data, we calculate absolute value of pos_cksaap - neg_cksaap.

However, the dimension of data only turn into a half, which means we still have 400 dimensions. In order to further reduce dimensions, we will only display values of 20 (pos_cksaap - neg_cksaap) with highest absolute values. The problem is that we also need to store corresponding names of aa pairs in right order, so we changed the code of quick sort used in the course to change order of indice of aa pair names at the same time. The code is shown below,

3.2 Function Override Strategy in Cross Validation

For logistic regression problem, generally the threshold will be selected as 0.5 (according to Bayes classification rule). However, in practice it doesnt offer good performance (which may due to the nonlinearity of the regression problem). Therefore, at the end of the project, we determined to add another procedure to our project we test a series of parameters and select the best one. However, in training function, the parameter are called as a private variable, and the function has been integrated to many parts of the project. How can we pass different thresholds into these functions? It seems not a good idea to rewrite the function again, so we use the overriding strategy. We copied the function and add another argument, where parameter can be passed to the function when its needed. Otherwise, the old function will be called. This strategy helps us avoid editing old function

3.3 Writing Classification Method as Class/Free Function

Each classification method can be divided into several steps, basically we have 2 choices writing them as class or free function. Finally, we write Decision Tree method, SVM and

```
int partition(vector<double> & vec, vector<int> & indice, int start, int finish) {
    double pivot = vec[start];
    int p_index = indice[start];
    int lh = start + 1;
    int rh = finish;
    while (true) {
        while (lh < rh && vec[rh] >= pivot) rh--;
        while (lh < rh && vec[lh] < pivot) lh++;
        if (lh == rh) break;
        double tmp = vec[lh];
        vec[lh] = vec[rh];
        vec[rh] = tmp;
        int index = indice[lh];
        indice[lh] = indice[rh];
        indice[rh] = index;
    }
    if (vec[lh] >= pivot) return start;
    vec[start] = vec[lh];
    vec[lh] = pivot;
    indice[start] = indice[lh];
    indice[lh] = p_index;
    return lh;
}
```

Figure 3.1: The Code for dimension reduction

KNN method as class and Nave Bayes as free function. The reasons are following:

- For Decision Tree method, SVM and KNN, each steps need to use same data (e.g., the features of data, the labels of data, etc.). When we write them as class, these data can be copied as a private variable of the class, so we avoid call them again and again as argument in each steps.
- 2. For naïve Bayes, since most of its training steps are doing descriptive statistics - some of these steps will be called beyond the classification procedure.

3.4 Self-defined Structure for Decision Tree

The biggest problem that we met so far is how to find the optimal multi-way partition using the gain method. In this project, we set 5 parameters and 1 return value, which mean: continuous vector, matrix of raw data, labels of raw data, continuous attribute of the column of indexth, a threshold.

First, calculate the information entropy of the entire matrix, and then get the information gain of each segmentation point, and the value of the cut point of the maximum information gain and the median value of each of the two values and the information gain of each cut point are obtained. Use the FOR loop to calculate the information gain value for each cut point and then find the cut point for the largest information gain. Set a threshold, if the information gain is less than the threshold, then the amount cannot continue to be split. Otherwise, the segmentation can be continued, and the vector can be divided until it cannot be divided.

In the code of this part of the decision tree, we have used various methods of vector and matrix, and we have realized the normal operation of the code in the transformation of various data structures. This is our advantage.

Chapter 4

Why Do We Deserve Good Grade

4.1 Integration of Whole Machine Learning process

This program offers a very compact view of all steps of a typical machine learning procedure from data preprocessing, model training, model evaluation to data visualization. Though the algorithm involving with it is not very complicated, but they are highly used and mentioned in undergraduate level courses (e.g. ERG2050, CSC4008, BME4050 in CUHKSZ). For Biology major students, this program can help them classify protein with several methods at the same time and offer them a comprehensive view of each models effect (they can also choose a best model according to their own demands).

4.2 Comprehensive View of Several Common-used Algorithm

For statistics/computer science major students, it offers them a deeper insight to these common classification methods for example, Nave Bayes is generally thought as one of best simple algorithms which can compared to the effect of one-hidden-layer neural network, however it doesnt perform well in our protein classification context. That is because of the high dependence between the different amino acids (especially for those who are close to each other).

4.3 High Efficiency of Optimizing Parameters

Due to the efficiency of C++, the program can help run some computational-expensive method within a short time. For example, KNN as a lazy learning, generally takes a lot of time while predicting. For example, for our 492*400 data frame, it takes 3 minutes on python to test the training accuracy. But this task only takes 12s on C++. We can see that for protein analysis context, KNN (or other instanced-based) generally offer the best accuracy. Under this circumstance, the strength of C++ (its efficiency) will be significantly emphasized. In addition, it also offer the possibility for us to select optimized parameter. For example, if we want use cross validation (12 folds) to select best parameter among 5 parameters, it will take about 150 minutes. But this can be finished within 5 minutes by our program.

4.4 Avoid Redundant Steps for Beginners of Machine Learning

Oriented clearly at one certain kind of questions and all the process, what we do to deal with the question is very suitable. For example, in data preprocessing, we do not construct other method that aims to clean data, like using mean value to fill up missing value because we do not have missing values in this situation. If we write functions like it, our program will go through redundant steps that wastes time. Apart from eliminating useless methods, we also make our process suitable in this case. Like in the k-means method that aims to do cluster to find out outliers, we set the k to be 2 because we have two results in the end. Therefore, we do not just use classical methods and turn them into codes, we change them in this situation and make them more suitable to deal with such problems.

Furthermore, the steps of selecting best parameters, though mentioned in some beginning machine learning courses, are seldom used in practice for undergraduate students. We just integrate the optimization part with the training steps and hide it to reduce the complexity for users. Due to the high efficiency of C++, the total running time is still within considerate time.

4.5 A more efficient method to converge to optimal solution

For an application to predict a binary variable, it is quite natural to work out the idea of the logistics regression model. However, there may be some barriers in between, among which the issue of convergence stands out. It is apparent that not all sigmoid functions of a linear model converge. To solve the problem, I went through the optimization materials and finalized with the solution of logarithm. It will transform the sigmoid function into a convex optimization form so that the solely global minimum point can be worked out.

4.6 New Data Type Definition

Second, it takes a little bit significant amount of time to train the parameters with the size of the data inputted in our example, a data frame of 400*400 size and many other iterations. I tried to solve the problem by defining a class call DoubleArray which utilizes the array as the fundamental data structure. Since it seems to be a much more underlying data structure than vector, I hope that it saves some time at a great degree. Unfortunately, it doesnt provide a desirable performance and results into some memory management problems. Hence, I tried to fresh my logic and turn to other directions. After browsing the related references, I find out that a variation in the learning rate (alpha) may give a great help in the training process. After testing for different values of the learning rate with different magnitude, I pick the most feasible one which will not result in the overshooting problem, where overshooting means that global minimum will not be achieved. The time of training decreases for about 30% with this strategy.

4.7 Parallel Task

During the process of the integration and coordination of each parts, I found that we should have assign each member parallel tasks instead of continuous tasks. So we can make good use of time to complete different work of data mining, prediction and console construction and finally link the whole pipe line to give users ideal prediction of unknown peptides.

4.8 Integration

b. When dealing with the visualization part and the modeling part, I find the most hard part is for integration. During the process of the integration and coordination of each part, I found that object-oriented programming is strong, but not enough. To better integrate each parts, I should have designed a uniform standard format for modeling and visualization. Despite the fact that we used many object-oriented programming techniques, we should have used more.

4.9 Using QtCharts

c. The QT charts and the QT is actually not hard to use only if you are very scared of them. Take the 3D visualization and tree visualization as examples, it did not take me much time to finish them.

Chapter 5

Feedback to Our Project and Course

5.1 What we have learned

5.1.1 QtCharts

Its a very useful tool for us to plot line chart, bar chart and scatter plots. While we use this tool, we also encountered a lot of advanced QT type like widget, QtChart, QtMainwindow. We learned some special method of them which is quite useful to our project (like setting the size of window, setting the axis, etc.). This experience also offer us a deeper impression of OO programming in the world of C++, everything can be designed as an object. Its first time for us to see that the plots title, axis, line are designed as object, which is quite different from other language (e.g MATLAB, R).

5.1.2 Collaborative Development

In the project, we learn how to divide a big problem into several independent small parts and then, how to work in a team. To improve our efficiency, we must do our own part independently, not waiting for the front person to finish his part and then start your own section. However, it will increase the workload of binding the small parts together. It is the most valuable gains we get through this project. To achieve it, we should first know what we need in every part and what our next part need, so we can decide what the input is and output should be. Also, we need to know the main purpose and principle of the functions in head file. In the process of putting parts together, we improve our understanding towards head files and know more about how to design a project.

As for the coding skill, we certainly have a better understanding to what we have learned and what we newly know. An impressive case is the use of inheritance, constructing a new class for that need to implement two other independent classes. This provides us with a strategy when we are dealing with one object in several steps in order, we can use inheritance.

Also, because we have many team members, we take highly advantage of applying object-oriented programming techniques. Each member constructs his/her part as an object. Then in the main function, we can easily integrate different parts by instantiation.

Different with plain algorithms, we implemented many visualizations for users to gain a better understanding of the modeling procedure, which is very helpful for our users, those biological students lacking in knowledge of machine learning. Some of the graphs are bar charts, some of them are line charts, some of them are 2D, some of them are 3D and some

of them are even dynamically drawn to show the training procedure. We believe our software bring people convenience of modeling without losing the deep analysis of the modeling, and also, we got well trained to use others packages to achieve our own goals.

5.2 Suggestions to lecture

5.2.1 Basic knowledge of QT (Especially QT Data Type)

While we are working on our project, we find that QT is especially a useful tool for our project it provides a very intuited way to build user interface (UI) and many efficient containers. However, we met a lot of troubles while suiting these tools with basic c++ data type (for example, the operation between char, std::string and Qtstring). I think since QT is the tool we used to do assignment, the lecture can also talk something basic about QT, like some useful QT containers, basic UI design, etc.

5.2.2 Lecture itself

a. Ask more questions to students to see if they have understood what the professor has taught them.

b. Qt is a very strong programming tool, you can let TA to teach them more about Qt and its sample projects.

c. During the project, all kinds of potential problems can be met, and you could encourage teams with similar project title to communicate together and let them display their experience during the semester

Reference

(1) Wenzhou Lu, QT 5: Development and Cases (3rd edition), 2017.

(2) Sartaj Sahni, Data Structures, Algorithm, and Applications in C++ (second edition), 2011.

(3) Qcustomplot: `http://www.qcustomplot.com/index.php/`.

(4) CS107 Programming Paradigms: `https://see.stanford.edu/course/cs107`

(5) CS229 - Machine Learning: `https://see.stanford.edu/Course/CS229`

(6) Jiawei Han, Micheline Kamber, Jian Pei: Data Mining: Concepts and Techniques,In The Morgan Kaufmann Series in Data Management Systems (Third Edition),Morgan Kaufmann, 2012

(7) Dan GusfieldAlgorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology (First Edition)Cambridge University Press, 1997

(8) Casella, G.Berger: Statistical Inference, Cengage Learning, 2001

(9) VTK package: `https://vtk.org/`

(10) David Abrahams, Aleksey Gurtovoy: C++ Template Metaprogramming, Pearson Education, 2005

(11) Sudheer Gupta, Ashok K. Sharma, Vibhuti Shastri, Midhun K. Madhu and Vineet K. Sharma: Prediction of anti-inflammatory proteins/peptides: an insilico approach, Journal of Translational Medicine, 2017

(12) Piyush Agrawal, Sherry Bhalla, Kumardeep Chaudhary, Rajesh Kumar, Meenu Sharma, Gajendra P. S. Raghava: In Silico Approach for Prediction of Antifungal Peptides, Frontiers in Microbiology, 2018

(13) Salman Sadullah Usmani, Sherry Bhalla, Gajendra P. S. Raghava: Prediction of Antitubercular Peptides From Sequence Information Using Ensemble Classifier and Hybrid Features, Frontiers in Microbiology, 2018

(14) Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani: An Introduction to Statistical Learning: With Applications in R, Springer Publishing Company, 2014

(15) QT Sample Projects