Master Thesis

Heterogeneous Error-Correcting Output Codes

F. Gediz Aksit

Master Thesis DKE 14-02

Thesis submitted in partial fulfillment
of the requirements for the degree of Master of Science
of Artificial Intelligence at the Department of Knowledge
Engineering of the Maastricht University

Thesis Committee:
Dr. Evgueni Smirnov
Dr. Mark Winands
Fırat İsmailoğlu

Maastricht University
Faculty of Humanities and Sciences
Department of Knowledge Engineering
Master Artificial Intelligence

25 February 2014

This page is intentionally left blank.

Abstract

Error-Correcting Output Codes (ECOC) is a machine learning algorithm that converts multi-class classification problems into multiple binary classification problems through a codeword matrix. One assumption behind ECOC is solving these binary classification problems with a classifier of the same type forming a *homogenous* classifier composition. With this work we propose a *Heterogeneous ECOC* approach that relaxes this property of ECOC. This approach allows solving the same binary classification problems with any one of the classifiers in the pool of binary classifiers forming a *heterogeneous* classifier composition. While this change may appear intuitive, it also brings complications with it.

For instance, this scheme change prompts an optimization problem where the algorithm needs to find the classifier composition with the highest performance. In connection, another complication is distinguishing classifier compositions from each other. To determine the best metric to judge the performance of classifier compositions, we used knowledge from the binary class values of predicted training instances such as the pairwise distance among these instance columns and/or rows as well as accuracies of predicted binary class values of validation instances. Using this knowledge we ran our experiments to compare the four different units of fitness: column distance, row distance, column × row distance, validation accuracy. For this purpose we initially used exhaustive search but then favored genetic algorithm search instead.

We used traditional (homogenous) ECOC as a benchmark to compare with the Heterogeneous ECOC approach. Through 10-fold cross validation, our experiments shown that while the three distance based approaches had mediocre performance – often being inferior to traditional ECOC, validation accuracy based unit of fitness outperformed both rivaling units of fitness and traditional ECOC.

Keywords: Classification, Data Mining, Error-Correcting Output Codes, Genetic Algorithm, Machine Learning

Acknowledgements

Firstly I'd like to thank my father for diverting his attention away from his own PhD thesis and tolerating my less than ideal writing as well as for his moral and financial support. Secondly I'd like to thank my thesis supervisor Evgueni Smirnov and Fırat İsmailoğlu for their endurance and support. This thesis could not have been possible had it not been for their generous allocation of time and patience with me.

[1] http://creativecommons.org/licenses/by-sa/3.0/

Table of Contents

Table of Figures

Table of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Networks
CART	Classification And Regression Tree
CHAID	CHi-squared Automatic Interaction Detector
CMC	Contraceptive Method Choice
DNA	DeoxyriboNucleic Acid
DNN	Deep Neural Networks
ECOC	Error-Correcting Output Codes
GA	Genetic Algorithm
ID3	Iterative Dichotomiser 3
MARS	Multivariate Adaptive Regression Splines
NB	Naïve Bayes
OCR	Optical character recognition
PDP	Parallel Distributed Processing
RNN	Recurrent Neural Network
SVM	Support Vector Machine
SVR	Support Vector Regression

1 Introduction

With this work we investigate relaxing one of the assumptions of the *machine learning* algorithm Error-Correcting Output Codes (ECOC) (Diettrich & Bakiri, 1995) for multi-class classification problems where the algorithm assumes a homogeneous composition of its binary classifiers. We investigate the validity of this approach and also address the optimization problem this approach introduces where we need to find the heterogeneous composition of binary classifiers with the best performance.

We have broken our research into five chapters. In this chapter we briefly introduce our research topic and an overview of our approach. In chapter two we introduce a pool of well-known classifiers we use for our binary classification problems and we also discuss well-known methods to convert multi-class problems into binary problems. In chapter three we introduce the Heterogeneous ECOC approach and also discuss our approaches towards solving the aforementioned optimization problem. In chapter four we present our experiments where we discuss and compare the performance of the individual units of fitness as well as how it compares with Homogenous ECOC. In chapter five we discuss our findings and also discuss further research possibilities.

1.1 What is Machine Learning?

Since the first operation of ENIAC (Electronic Numerical Integrator And Computer), the first fully electronic computer in 1946, nearly seven decades ago, computing has greatly benefited in breakthroughs in engineering as well as other physical sciences. For instance with breakthroughs such as transistors, integrated circuits, microprocessors, parallel computing, supercomputers, etc. computing capabilities has seen a very drastic increase in

capability. With possible future breakthroughs such as DNA or quantum computing further exponential increase in computing capabilities in the near future is anticipated (Cory, Fahmy, & Havel, 1997).

Computer science has also made and is still making its own breakthroughs in a variety of fields utilizing this ever more powerful tool. Branch of *Artificial Intelligence (AI)*, *Machine Learning* is one of these fields which is the scientific the study that investigates systems capable of learning from data rather than being programmed with explicit instructions. In the past six decades the brief history of Machine Learning has seen a number of noteworthy milestones.

One such milestone was the first learning program in 1952 by Arthur Lee Samuel that became better at playing the game checkers by learning winning strategies in a *supervised learning mode* against human players as well as against itself. *Perceptron* (Rosenblatt, 1957) was another milestone that built upon the prior research on *Neural Networks* connecting a mesh of nodes where simple decisions are made in each resulting in a complex decision when put together. 1967 saw the first programs capable of *pattern recognition* using algorithms such as nearest neighbor. *Explanation Based Learning (EBL)* capable of analyzing training data and discarding irrelevant information was introduced by Gerald Dejong in 1981.

In 1990s Machine Learning saw a wide variety of applications in a variety of areas such as *data mining*, *text learning*, and *language learning*, naming a few. 2000s and onward is seeing an explosive growth in applications of Machine Learning which no longer is a curious research topic among computer scientists with its vast practical real-world uses in research, engineering, commerce, forensics, and practically anything imaginable involving data. One of the practical uses of Machine Learning is classification tasks through *supervised learning* where a dataset with labels is used for training.

1.2 Classification Problem

In this work we investigate *binary classification* and *multi-class classification* methods. Binary classification problems involve classifying elements of a given set into two groups. Multi-class classification problems involve classifying elements of a given set into groups of three or higher.

One of the challenges of multi-class classification is that most powerful classifiers – even those usable or extendable for multi-class classification – are typically optimized for binary classification (Valiant, 1984). This strongly suggests that it would be beneficial to develop methods to convert multi-class classification problems into binary classification problems (Natarajan, 1991). There are a number of methods that is commonly used to convert multi-class classification problems into multiple binary classification problems such as *One-Vs.-All* (Rifkin & Klautau, 2004), *One-Vs.-One* (Allwein, Schapire, & Singer, 2001)*, and Error-Correction Output Codes* (*ECOC)* (Diettrich & Bakiri, 1995).

Our focus on this work will be on ECOC which offers a robust "classification by consensus" approach significantly reducing the effects of noise in the data - unlike approaches such as One-Vs.-All or One-Vs.-One. ECOC uses *codwords* to achieve this where each of these codewords refer to a class from the multi-class classification problem. When combined, rows of these codewords form a binary matrix where each column forms a binary classification problem. Ideally, the design of the binary matrix should have high *column separation* and *row separation* which measured in terms of column and row Hamming distance. This reduces the effects of overfitting as well as correlation of errors. This high Hamming distance should also be non-complimentary to each other as some binary classifiers treat a class and its compliment symmetrically. With exhaustive settings, for a *k*-class multi-

class classification problem ECOC creates $b = 2^{(k-1)} - 1$ binary classification problems. One of the assumptions behind ECOC is that all of the mentioned b binary classification problems must be solved by the same *binary classifier* in a homogeneous manner which we will henceforth refer to as *Homogeneous ECOC*.

With this work we research if it is sound to relax this homogenous assumption where we would introduce diversity among the binary classifiers selected from a *pool of classifiers* solving the binary classification problems in a heterogeneous manner which we will henceforth refer to as *Heterogeneous ECOC*. With this diversity however Heterogeneous ECOC approach creates an optimization problem to find a composition of binary classifiers with the highest performance. In an attempt to find a metric to distinguish binary classifier compositions among each other, we compared the performance of four different approaches: column distance (instance column separation), row distance (instance row separation), column × row distance (instance column × row separation), and validation accuracy. We also compared the performance of Heterogeneous ECOC with the performance of Homogeneous ECOC which we treat as a benchmark. We verify the statistical relevance of our results through *10-fold cross validation*. We also perform a second 10-fold cross validation for the candidates from each fold to determine the composition with the highest validation accuracy.

We discuss well-known classification problems and methods in the next chapter serving as a precursor to the Heterogeneous ECOC approach. We first briefly introduce our pool of binary classifiers we use with our approach and then discuss well-known comparable multi-class to binary class conversion methods (One Versus All, One Versus One, ECOC).

2 Classification Problems and Methods

In Machine Learning supervised and unsupervised learning are the most important tasks. Unlike in unsupervised learning, in supervised learning we are provided with labels of classes. If these labels are real numbers, then the task for the learning algorithm is a regression problem; however if the labels are discrete values then the task becomes a classification problem. In classification we are going to discuss two types of tasks – namely binary classification which has two classes and multi-class classification which has more than two classes.

Many tasks in the real world can be thought of as complex data sets creating multi-class classification problems. This creates an immense number of applications for Artificial Intelligence (AI) such as biometric recognition, medical diagnostics, weather prediction, OCR, natural language processing, etc. where machine learning algorithms with a training set can be trained to make predictions on future data (test set) using statistical principles.

While some machine learning algorithms such as C4.5 and SVM can be extended to handle multi-class classification directly (Kong & Dietterich, 1995), it is often beneficial and easier to devise the problem as a binary (two-class) classification problem (Allwein, Schapire, & Singer, 2001). After all, many machine learning algorithms including the most advanced ones are designed and optimized for binary class classification tasks. Therefore there is a certain need (Kong & Dietterich, 1995) for the reduction of multi-class classification problems into a series of binary classification tasks. Even though this brings some computational cost, it significantly improves the accuracy of predictions (Diettrich & Bakiri, 1995). Furthermore, binary classification has a greater variety of techniques in comparison to multi-class classification. (Flach, 2012) Hence, we can exploit and employ these techniques

in their original forms. One of these techniques is Error-Correcting Output Codes (ECOC) which we investigate improving with this work.

2.1 Classifier Pool

Once we reduce the multi-class classification problem down to multiple binary classification problems, various machine learning techniques become useable. Our approach (discussed in the next chapter) utilizes a pool of classifiers to solve these multiple binary classification problems.

A larger pool of classifiers is beneficial as it offers a greater number of possible solutions for the multiple binary classification problems. It is however important to note that the search space to find the optimal combination of these algorithms for the individual binary classification problems also increases with a double exponential manner which has an impact on performance (See Figure 11). While we used the binary classifiers ANNs, SVMs, and Logistic Regression and binary/multi-class classifiers Naïve Bayes and Decision Trees, any combination of any two or more classifiers for supervised learning can be used in their place. We briefly discuss these classifiers below.

2.1.1 Artificial Neural Networks

*Artificial Neural Network*s (ANN) (McCulloch & Pitts, 1943) is a binary classifier that uses inter-connected nodes (input nodes, hidden nodes and output nodes) called *neurons* which mimic biological neural networks. The data travels from the input neuron through one or more inter-connected hidden neurons ending up in an output neuron (See Figure 1). With methods such as defining a *cost function*, ANN can be trained to find a more optimal solution

to problems it is able to solve. It can be further refined if the cost function is modeled on the observations on the data.

We used a very simple feed forward ANN implementation with a hidden layer size of 20 and each run only had 25 iterations/epochs in an attempt to save time. This is much lower than the default of 1000 iterations. We believe this is the main reason as to why ANN has significantly underperformed. This however had an unintentional consequence that has provided most interesting results (Chapter 5.2).

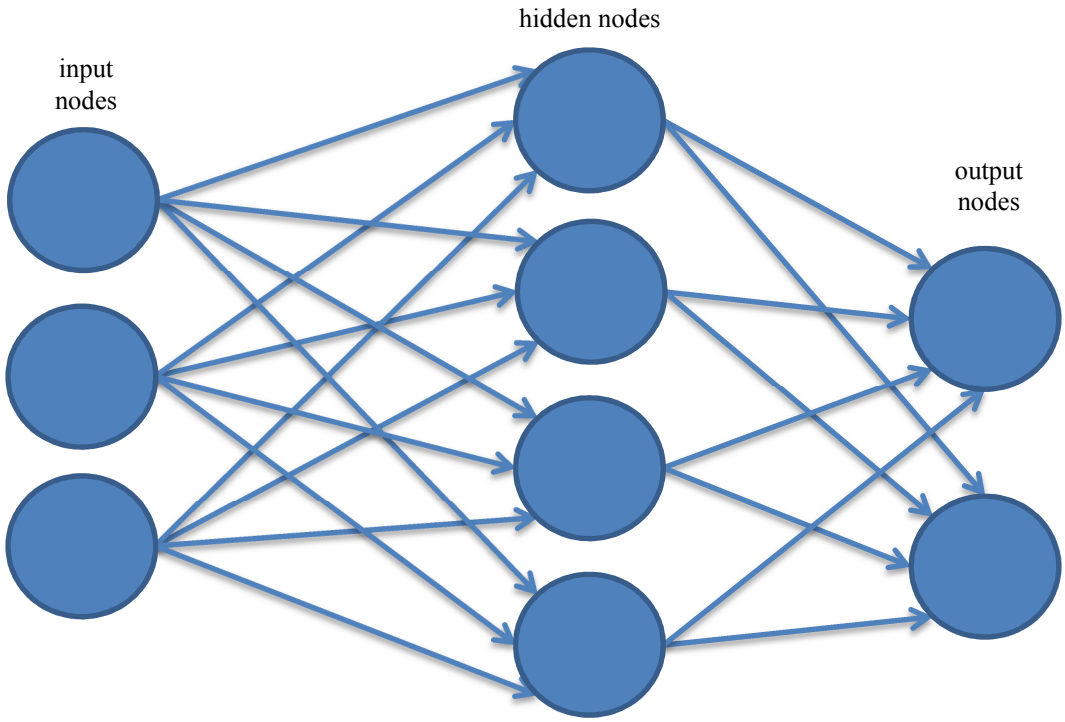

Figure 1: Inter-connected nodes (input, hidden, output) of ANN

In 1943, three years before the first operation of ENIAC, Warren McCulloch and Walter Pitts introduced a new computational model which has come to be known as ANN. ANN has seen improvements since such as Perceptron (Rosenblatt, 1957), Parallel Distributed Processing (PDP) (Rumelhart & MacClelland, 1988), Recurrent Neural Network (RNN) (Williams & Zipser, 1989), Deep Neural Networks (DNN) (Larochelle, Bengio, Louradour, & Lamblin, 2009) to name a few. While the popularity of ANN as a tool for

machine learning diminished with the availability of much simpler methods in 1990s, it regained some of the popularity it lost with more recent improvements.

2.1.2 Support Vector Machines

Support Vector Machines (SVM) (Cortes & Vapnik, 1995) is a non-probabilistic binary classifier. SVM is a linear generalization of Generalized Portrait Algorithm (Vapnik & Lerner, 1963) double check. SVM separates two groups' elements in a data by building hyperplane(s) in high or infinite dimensional space. This can offer a good separation of data elements (Vapnik, 1998) if the hyperplane provides a large distance to the nearest training element of any class.

Several extensions to SVM exist such as Support Vector Regression (SVR) (Drucker, Burges, Kaufman, Smola, & Vapnik, 1997), Multiclass Support Vector Machines (Weston & Watkins, 1998), Transductive Support Vector Machines (Gammerman, Vovk, & Vapnik, 1998) and Structured Support Vector Machines (Finley & Joachims, 2008).

2.1.3 Logistic Regression

Logistic Regression or *Logit Regression* (Bishop, 2006) is a binary classifier that uses probability to predict the values of dependent variables using the relationship between dependent and independent variables. Unlike linear regression that estimates numeric values, logistic regression estimates probabilities. Logistic regression calculates a threshold by weighting and treating the features. Logistic Regression then determines the class probability by using the distance from this threshold of each new instance in a sigmoid function where the said probability increases with the distance from the threshold.

Typically in logistic regression implementations where an exact binary classification is required, classifier is given a second threshold where a particular instance is classified as belonging to a class if its probability is higher than this threshold and if not it is classified as belonging to the other class. In our implementation we use the threshold of .5 and any instance with less probability is classified as belonging to the other class.

Several extensions to Logistic Regression also exist such as, Mixed Logit (Revelt & Train, 1998) Multinomial Logistic Regression (Multinomial Logit) (Hedeker, 2003), Ordered Logistic Regression (Ordered Logit) (Hilbe, 2009).

2.1.4 Naïve Bayes

Naïve Bayes (Ng & Jordan, 2002) also known as an *independent feature model* is a probabilistic binary and multi-class classifier that makes strong assumptions that the presence or absence of any feature does not have any correlation with the presence or absence of any other feature. In a binary class and multi-class setting for each instance Naïve Bayes returns one probability value for each class based on the features and prefers the more probable candidate as the class.

Despite its simple seemingly counterintuitive and even unrealistic assumption, Naïve Bayes is remarkably successful with results comparable (Rish, 2001) to more sophisticated methods. A significant benefit over other more complex classifiers is that Naïve Bayes is not very complex computationally however as a result a key disadvantage of Naïve Bayes is that its use is limited to more simplified models.

2.1.5 Decision Trees

Decision Trees existed prior to their first use in computing but were hand drawn. Computing saw the implementation of decision trees in the form of ID3 (Quinlan J. R., 1986), C4.5 (Quinlan J. R., 1993), C5.0 (Kuhn & Johnson, 2013). Despite their simple structure, decision trees are a robust method commonly used in machine learning (Rokach, 2010) as a binary or as a multi-class classifier.

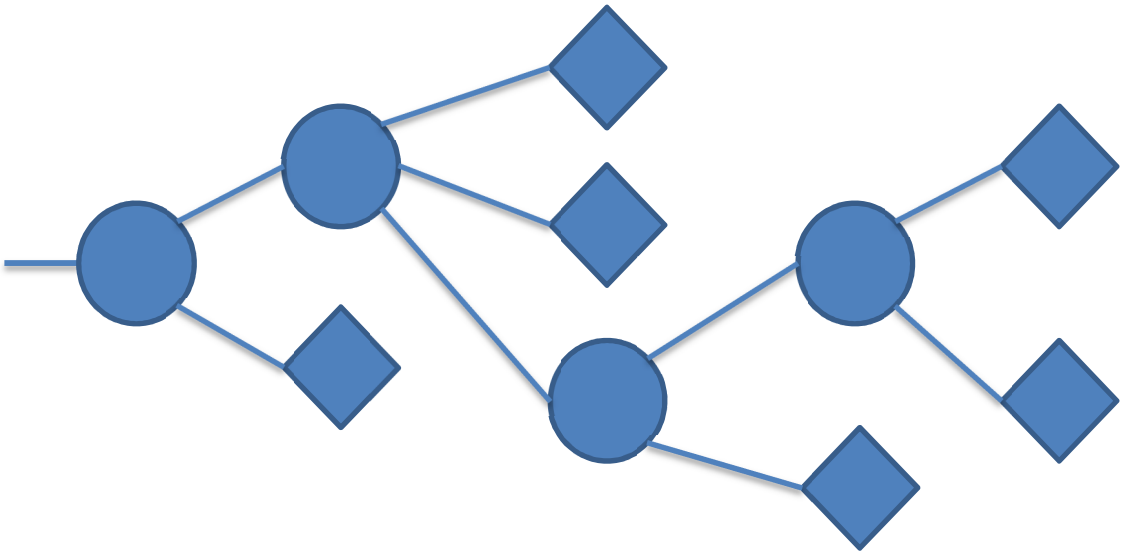

Figure 2: A sample decision tree

Decision trees have a design that resembles the structure of trees. The *tree* starts at a *root* node which branches to either more branches which serve as *internal* or *test* nodes or to leaves which serve as *terminus* or *decision* nodes. Figure 2 has sample decision tree diagram where circles represent internal nodes and diamond shapes represent terminus nodes.

Notable decision tree implementations include Iterative Dichotomiser 3 (ID3), C4.5, C5.0, Classification and Regression Tree (CART), CHi-squared Automatic Interaction Detector (CHAID), Multivariate Adaptive Regression Splines (MARS). Decision Tree algorithms typically build the tree in a top-down recursive manner where during construction or during search iterations correspond to the individual nodes – be internal or terminus.

2.2 Multi-Class to Binary-Class Conversion Methods

Multiple techniques exist to convert multi-class classification problems into multiple binary classification problems such as One Versus All (Rifkin & Klautau, 2004), One Versus One (Allwein, Schapire, & Singer, 2001), and our main focus: Error-Correcting Output Codes (ECOC) (Diettrich & Bakiri, 1995).

2.2.1 One Versus All

One Versus All (Rifkin & Klautau, 2004) also known as *One Versus Rest* progressively separates each of the *k*-classes into *k*-binary classifiers such that during training all instances referring to every class k_i' is treated as a negative example for class k_i. Every new instance will be classified based on the class with highest confidence. While simplistic, One Versus All preforms on par with more complicated methods.

One significant assumption of One Versus All is that the prediction scores determined by the binary classifiers are comparable which leads the One Versus All classifier to assign the query instance with the class that corresponds to the largest prediction score. However, this is not often the case, in particular when the class distribution is far from being uniform. The other downside of this type of classifier is its disregard of class inter-relationships, such as hierarchical relationships (Melvin, Ie, Weston, Noble, & Leslie, 2007).

2.2.2 One Versus One

One Versus One (Allwein, Schapire, & Singer, 2001) also known as *pairwise classification* or *round robin classification* progressively separates each of the *k*-classes into *k*-binary classifiers such that during training every class k_i is only paired once with each class k_j where class k_i is treated as a positive example and class k_j as a negative example while

ignoring every class k'_{ij}. Similar to One-Vs.-All method every new instance will be classified based on the class with highest confidence. One-Vs.-One is an increasingly more popular technique for efficiently converting multi-class problems into binary problems performing significantly better (Fürnkranz, 2002) than One-Vs-All.

2.2.3 Error-Correcting Output Codes (ECOC)

Error-Correcting Output Codes (ECOC) (Diettrich & Bakiri, 1995) offers a robust "classification by consensus" approach significantly reducing the effects of noise in the data - unlike approaches such as One-Vs.-All or One-Vs.-One. (Berger, 1999) ECOC achieves this by creating rows of *codewords* to correspond to each class forming a binary matrix in such a manner that each codeword is well separated in Hamming distance making them distinct from each other. Furthermore each column of bit position function f_i should not correlate with the neighboring columns and this can also be achieved through high Hamming distances among the columns. In exhaustive settings, for a k-class multi-class classification problem $b = 2^{(k-1)} - 1$ columns are created which corresponds to the maximum number of non-complimentary combinations of the bit position functions forming b columns of ECOC codewords. The exponential nature of the number of binary classification problems makes ECOC computationally very expensive for higher class problems (See Figure 3).

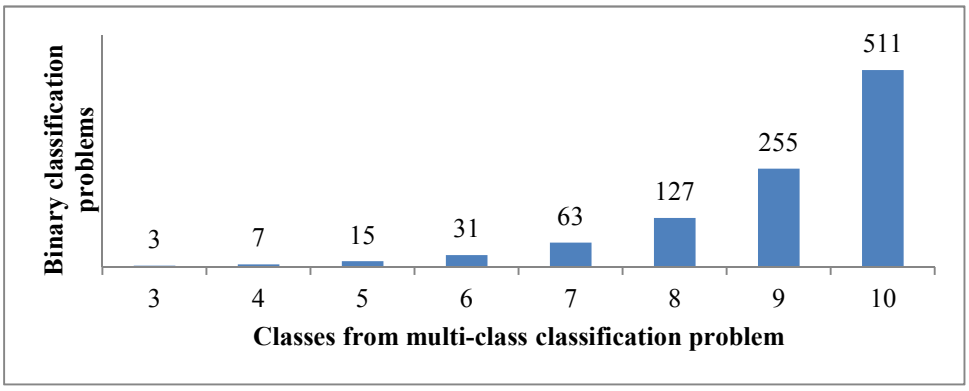

Figure 3: Number of binary classification problems per number of multi-class classes

The purpose for this rather complicated setup is to avoid similar or even correlated errors as each of these columns is actually a binary classification problem. After all, ECOC will only succeed in correcting errors if the error itself isn't consistent on the majority of the bit positions of the codeword matrix. Furthermore the errors will highly correlate if the columns have complimentary rows. This is because not only would the same errors be repeated on the complementary rows but also most learning algorithms will treat a class and its compliment symmetrically which can result in worse than random performance as the algorithm will construct the same search environment for both binary values interchangeably. (Kowalczyk & Chapelle, 2005) This is why column separation is used to try to minimize similarity and complementariness among the columns. (Diettrich & Bakiri, 1995)

Multi-class	Codeword (each row)							
	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
1	0	1	0	0	1	0	1	1
2	0	0	1	0	1	1	0	1
3	0	0	0	1	1	1	1	0

Figure 4: All possible columns for 3 class codewords

However it is very difficult to meet both of these goals as there only are $2^3 = 8$ possible combinations for a 3 class problem (See Figure 4). As perhaps immediately observable, the right half and the left half of the columns are a compliment of each other so removing four columns we are left with only four columns of which one is either a vector of all zeros or ones making it useless for discrimination among rows leaving us with only 3 columns to work with (See Figure 5). (Diettrich & Bakiri, 1995)

Multi-class	Codeword (each row)		
	f_1	f_2	f_3
1	1	1	1
2	0	0	1
3	0	1	0

Figure 5: ECOC code matrix for a 3-class multi-classification problem

Unlike One-Vs.-All or One-Vs.-One which directly predicts the class of new instances, ECOC predicts a codeword for every new instance application of then binary classifiers. If an exact match to the codeword is not found, the codeword with the closest Hamming distance will be chosen. This is the key component of ECOC that facilitates the error correction. Once a codeword is decided, it will then be *decoded* to a multi-class value from the original multi-class classification problem.

For instance, consider the ECOC codeword matrix generated for a 4-class classification problem (See Figure 6). The codeword 1 0 0 0 1 1 1 would have the closest Hamming distance to 0 0 0 0 1 1 1 which would be decoded to class 2. This would hence correct the error by the first binary classifier.

Multi-class	Codeword (each row)						
	f_1	f_2	f_3	f_4	f_5	f_6	f_7
1	1	1	1	1	1	1	1
2	0	0	0	0	1	1	1
3	0	0	1	1	0	0	1
4	0	1	0	1	0	1	0

Figure 6: ECOC codeword matrix for a 4-class multi-classification problem

ECOC however does not take advantage of the knowledge from the individual instances from the actual data into account when constructing the codeword matrix with its predetermined make-up. Furthermore, even if the ideal conditions discussed above regarding column and row separation are reached, ECOC requires that the all of the binary classification problems are to be solved by the same binary classifier in a *homogenous* manner which may result in correlated errors. This is particularly problematic because ECOC can only recover from errors if and only if the errors themselves are not correlated and the homogeneous selection of classifiers can lead to correlated errors – particularly errors due to overfitting the training data.

2.3 Conclusion

Intuitively, relaxing homogenous classifier composition assumption of ECOC may appear sound; however such a modification introduces new complications. In the next chapter we investigate these complications and the feasibility of a Heterogeneous ECOC approach where individual binary classification problems can be solved by different binary classifiers selected from a pool of classifiers. Heterogeneous ECOC approach however creates an optimization problem where the algorithm now needs to decide on which classifier composition to use to solve the individual binary classification problems. This however generates another problem where we need to be able to distinguish the performance of classifier compositions for which we investigate four different approaches: column distance (instance column separation), row distance (instance row separation), column × row distance (instance column × row separation), and validation accuracy. We investigate the feasibility of exhaustive search and genetic algorithm search with using these four approaches. In order to avoid the problems discussed for Homogeneous ECOC, we discuss our use of 10-fold cross validation to verify the classifier composition found by the search approach in a statistically relevant manner. However this statistical approach we use particularly to avoid overfitting the dataset creates yet another problem where each fold can return one or more different compositions for which we discuss distinguishing by testing each candidate from each fold through a second 10-fold cross validation to find the optimal composition.

3 Heterogeneous ECOC

As discussed thus far, one of the assumptions behind the approach for ECOC is that all of the *b* binary classification problems (columns of the codeword matrix) are assumed to be solved homogeneously by classifiers of the same type (Kong & Dietterich, 1995).

It may appear logical to relax this homogeneous assumption however such a modification to ECOC generates new problems. In this work we investigate the feasibility and effects of the *Heterogeneous ECOC* approach where instead of having a one-fits-all classifier, we explore using a pool of *c* classifiers to solve the *b* binary classification problems heterogeneously such that every column of the codeword matrix can have any one of the *c* classifiers from the said pool of classifiers. As it may be evident, this heterogeneous treatment of ECOC creates an optimization problem where the algorithm needs to decide on a classifier composition more fit in solving the individual binary classification problems converted from the multi-class classification problem. This suggests the algorithm needs to be able to distinguish the *fitness* of classifier compositions from each other.

3.1 Units of Fitness

In the Heterogeneous ECOC approach, while we perform machine learning through the same codeword matrix from Homogeneous ECOC, the codeword matrix does not utilize knowledge from the data. Hence, the codeword matrix even with its column and/or row separation calculations does not make it possible for us to distinguish classifier compositions from each other. This prompted us to investigate four *units of fitness* that offered the ability to distinguish the classifier compositions. We have compensated for this shortcoming by utilizing knowledge from the training and validation instances.

3.1.1 Column Distance

Homogeneous ECOC relies on column separation of the codeword matrix to have more distinct binary classification problems for the classifiers. The rationale behind this is to minimize correlation of the solutions to the individual binary classification problems as if two binary classification problems have similar or identical solutions this will also mean any errors will also be correlated. ECOC can only correct errors if the errors themselves aren't correlated.

We utilized a similar column separation to distinguish Heterogeneous ECOC classifier compositions where we calculate the Hamming distance between the columns formed by the predicted solutions to the binary classification problems. We will henceforth refer to this as *column distance*. The key difference of our approach from the column separation of the fixed codeword matrix of Homogeneous ECOC is that we utilize the knowledge from the training set.

3.1.2 Row Distance

Homogeneous ECOC also relies on row separation of codeword matrix to have more distinct codewords. The purpose of having well separated codewords may be more straightforward as the more distinct each codeword gets, the more distinct each class of the multi-class classification becomes. ECOC will classify new instances by transforming the predicted codeword to the class codeword from the codeword matrix with the smallest Hamming distance. However if the Hamming distances among the classes are small, it is possible for even the smallest errors to be non-recoverable.

We utilize a similar row separation to distinguish Heterogeneous ECOC classifier compositions where we calculate the Hamming distance between each row of class c_i and all

rows belonging to every other class formed by the predicted solutions to the binary classification problems. We will henceforth refer to this as *row distance*. The key difference of our approach is that similar to column distance we utilize knowledge from the training set unlike the fixed codeword matrix of Homogeneous ECOC.

3.1.3 Column × Row Distance

Homogenous ECOC tries to maximize both column separation and row separation at the same time however it is difficult to satisfy both of these properties unless there are at least five classes (Diettrich & Bakiri, 1995).

Similarly, to distinguish Heterogeneous ECOC classifier compositions we attempted to maximize both column distance and row distance discussed in the previous two sections. We observed that the value of column distance and row distance do not necessarily scale well with each other so we compensated this by multiplying the two values in an attempt to have a unit of fitness that maximizes both. As with what is discussed in the previous two sections we utilize knowledge from the training set instead of the fixed codeword matrix of Homogenous ECOC.

3.1.4 Validation Accuracy

Homogenous ECOC relies on accuracy to decide which classifier to use homogenously where the classifier with the highest accuracy is selected. This approach however has a chance of overfitting the training data and statistical methods such as *k*-fold cross validation is used to circumvent this problem.

We decided to also use accuracy as our unit of fitness to distinguish Heterogeneous ECOC classifier compositions. However we recognize the same overfitting problem that

would arise from the use of training accuracy as the unit of fitness and we hence used validation accuracy instead. Using the training set, we first train using a certain classifier composition. Then, we use the same classifier composition to predict classes of the validation/test set. We then compare this prediction with the actual class values to see what percent of the test set classes we managed to predict correctly. While it may not appear to be straightforward, with this approach we also utilize the knowledge from the training set but also from the test set. One setback of this approach is that the processing of all that is mentioned takes significantly more time than other three approaches.

3.1.5 Knowledge from Predicted Training Instances

Thus far we have discussed our treatment of predicted training and validation instances to acquire knowledge to distinguish the Heterogeneous ECOC classifier compositions from each other where we have introduced four units of fitness to this end in the prior four sections. We consider first three units of fitness *distance based* and the last unit of fitness (validation accuracy) *non-distance based*.

How distance based methods acquire knowledge from training instances may not be intuitive. For the purpose of training we use the same Homogeneous ECOC codeword matrix. We have created a sample 3-class predicted training instances with a pool of three classifiers to demonstrate the heterogeneous treatment of ECOC which in the case of our 3-class classification problem would be a three-by-three matrix (Figure 5). We would like to note that these class values were randomly generated to serve as a hypothetical sample and has no actual correlation with any data.

We first train each *c* classifier on each *b* binary classification problem and then record the binary classification predicted by each classifier for each instance for each binary

classification problem (Figure 9). The classifiers will have varying levels of success in predicting the binary classifications. Our task at this stage is to decide on which of the three column/classifier to pick for each of the three binary classification problems b_1, b_2, b_3. With $b = 3$ and $c = 3$, the search space will have $b^c = 3^3 = 27$ possible combinations of these nine columns. Complication of Heterogeneous ECOC due to search space size will be discussed at a later section (Chapter 3.2).

Composition	Column distance	Row distance	Column × Row distance	Accuracy
ANN, ANN, ANN	30	410	12,300	35%
NB, NB, NB	26	502	13,052	65%
SVM, SVM, SVM	30	460	13,800	60%

Figure 7: Heterogeneous classifier composition values

While we will discuss how each unit of fitness acquires knowledge and briefly compare them, our experimental findings will be discussed at a later chapter (Chapter 4.2). Homogeneous compositions of classifiers from our sample have different accuracies (Figure 7) and ideally Naïve Bayes would be preferred by homogeneous ECOC. Also in our sample column distance values range between 26 and 36, row distance values range between 328 and 504, column × row distance values range between 11,152 and 16,128 and accuracy values range between 35% to 70%.

Composition	Column distance	Row distance	Column × Row distance	Accuracy
SVM, SVM, NB	**36**	414	14,904	60%
NB, SVM, NB	32	**504**	16,128	65%
NB, SVM, NB	32	504	**16,128**	65%
NB, SVM, SVM	34	432	14,688	**70%**

Figure 8: Classifier compositions with highest unit of fitness value (in bold)

Column distance values are calculated by pairwise distance among the predicted training instance binary class columns of the selected composition. For our sample, the ideal composition for column distance as unit of fitness would be the composition SVM, SVM, NB with the highest column distance value of 36 with an accuracy of 60% (Figure 8). However

this value is trailed by column distance value of 34 which corresponds to five different classifier compositions with accuracies ranging from 40% to 70% and so on.

Row distance values are calculated by pairwise distance among predicted training instance binary class rows belonging to each class and every other class of the selected composition. For our sample, the ideal composition for row distance as the unit of fitness would be NB, SVM, NB with the highest row distance value of 504 with an accuracy of 65% (Figure 8). However this value is trailed by column distance values of 502 with 65% accuracy, 492 with 40% accuracy, 466 with 65% accuracy, 464 with 60% accuracy and so on.

Column × Row distance values are calculated by the multiplication of the previously explained column distance value and row distance value. The motivation behind this unit of fitness was an attempt to maximize both column and row distance values. For our sample, the ideal composition for column × row distance as the unit of fitness would be NB, SVM, NB with the highest column × row distance value of 16,128 with an accuracy of 65% (Figure 8). However this value is trailed by column × row distance values of 15,744 with 40% accuracy, 14,904 with 60% accuracy, 14,688 with 70% accuracy, 14,400 with 45% accuracy and so on.

Accuracy values are calculated based on the accuracy of the predicted training instance binary class values when these are decoded back to multi-class values and compared to the actual class values where ECOC's error recovery plays an important part. For our sample, the ideal composition for accuracy as the unit of fitness would be NB, SVM, SVM with the highest accuracy of 70%. This value is trailed by accuracy value of 65% which corresponds to four different classifier compositions.

For our sample, increase in distance based units of fitness value did not always increase accuracy as even maximization of any of the said value did not correspond to the

optimal classifier composition. Hence at a glance a notable weakness of distance based units of fitness can be observed. We will discuss and compare the performance of these units of fitness in greater through experimental data in a later chapter (Chapter 4.2).

		b_1				b_2				b_3			
Multi-class	Instances	f_1	ANN	NB	SVM	f_2	ANN	NB	SVM	f_3	ANN	NB	SVM
k_1	i_1	1	0	1	1	1	1	1	1	1	0	0	1
	i_2		1	1	0		0	0	1		1	1	0
	i_3		0	1	1		1	1	0		1	1	1
	i_4		1	0	1		1	0	1		0	0	1
	i_5		0	1	1		0	1	0		0	1	1
k_2	i_6	0	1	0	0	0	0	0	0	1	0	1	1
	i_7		0	0	0		1	0	0		1	1	0
	i_8		0	1	1		0	0	0		1	1	1
	i_9		1	0	0		1	0	1		0	0	1
	i_{10}		0	0	1		0	0	0		0	1	1
	i_{11}		1	0	0		1	0	0		1	1	1
	i_{12}		1	1	0		1	1	0		0	1	0
	i_{13}		1	0	0		0	1	0		1	0	1
k_3	i_{14}	0	0	0	0	1	0	1	0	0	0	0	0
	i_{15}		0	0	0		1	1	1		0	0	0
	i_{16}		1	0	0		1	1	1		1	1	0
	i_{17}		0	0	1		0	0	1		1	0	0
	i_{18}		1	1	0		1	1	1		0	1	0
	i_{19}		1	0	1		0	1	1		1	0	0
	i_{20}		0	0	0		1	1	1		0	0	1

Figure 9: Sample of knowledge based on predicted training instances

3.2 Search Approaches Used

As discussed thus far, the core problem of our new treatment of ECOC is an optimization problem that needs to choose a binary classifier type for each of the binary classification problem. The search approach needs to find a classifier composition of length *b*

with *c* possible values each corresponding to *b* binary classifiers that are selected from a pool of *c* binary classifiers to solve *b* binary classification problems such that the validation accuracy *a* is maximized. The performance of each search approach will be discussed in the next chapter.

3.2.1 ID Vector

In order to simplify the programming but also express the optimization problem in a convenient median for search algorithms, we assign an ID to each of our pool of *c* binary classifiers in no particular order (See Figure 10). This assignment remains consistent in the entirety of the code's execution. Using these IDs we represent the optimization problem of *b* binary classification problems as a string of *b* discrete values of *c* possible values which we will henceforth refer as an *ID Vector*.

ID	Binary classifier
1	Artificial Neural Networks (ANN)
2	Naïve Bayes
3	Decision Tree
4	Support Vector Machines (SVM)
5	Logistic Regression

Figure 10: IDs of binary classifiers

3.2.2 Exhaustive Search

Initially we used an exhaustive search approach to find the highest column distance based on the publication by (Diettrich & Bakiri, 1995) where we have attempted to search for all combinations using a pool of only three classifiers (Decision Trees, SVM, Logistic Regression) on datasets with progressively higher number of classes. For this we created a matrix that arranged all instance combinations of the IDs of *b* binary problems with *c* classifiers. We observed that this approach quickly became impractical as it was taking far too much time and system resources just to create the said matrix which we would later

search for the optimal ID vector composition offering the optimal solution. This was not entirely surprising as the search space size of a *k*-class problem with *c* classifiers in the classifier pool is calculable through a double exponential function $\left(2^{(k-1)}-1\right)^{c}=b^{c}$ which basically is the number of binary classification problems raised to the power of *c* (See Figure 11). We as a result did not attempt this approach on other unit of fitness and instead only used a Genetic Algorithm.

Multi-class	Binary Classification Problems	Pool of two classifiers	Pool of three classifiers	Pool of four classifiers	Pool of five classifiers
3	3	9	27	81	243
4	7	49	343	2,401	16,807
5	15	225	3,375	50,625	759,375
6	31	961	29,791	923,521	28,629,151
7	63	3,969	250,047	15,752,961	992,436,543
8	127	16,129	2,048,383	260,144,641	33,038,369,407
9	255	65,025	16,581,375	4,228,250,625	1,078,203,909,375
10	511	261,121	133,432,831	68,184,176,641	34,842,114,263,551

Figure 11: Search space size for pools of classifiers

3.2.3 Genetic Algorithm

We decided to use genetic algorithms to process the search space to find classifier ID combinations that score highest on the four different unit of fitness we have previously defined. We ran the Genetic Algorithm for all four of the unit of fitness to compare and determine their usefulness.

A *Genetic Algorithm* is a local search that resembles evolution by taking a *population* of competing *genotypes* or *individuals* which encode possible solutions – typically in an array/string of bits or discrete values. Individual elements of this encoding are typically called *genes* or *chromosomes*. The genetic algorithm combines members of the population based on a *fitness function* to produce genotypes more fit. Least fit genotypes are dropped from the population and are replaced by genotypes derived from more fit genotypes through mutation, crossover and selection operations.

A *mutation* operation alters one or more genes of a genotype i_i such that it introduces greater diversity also serving as a recovery measure from local optima convergence should the genetic algorithm get trapped in one. It is important to note that too frequent mutations will lead to near-random results. There are a number of mutation types suitable depending on the type of the genotype array such as bit string mutation, flip bit mutation, boundary mutation, Gaussian mutation, uniform mutation and non-uniform mutation methods. In bit string mutation method a single random bit is inverted. In flip bit mutation method all bits are inverted. In boundary mutation method the entire genotype is replaced randomly either with an upper or lower bound. In Gaussian mutation method a Gaussian distributed random value is added to a random gene within defined bounds. In uniform mutation method a chosen gene is replaced a random value within defined bounds. In non-uniform mutation method similar to uniform mutation method a chosen gene is replaced by a random value within defined bounds but the probability of a mutation decreases as generation count increases.

A *crossover* operation mixes the encoded solution of two parent genotypes g_i and g_j such that new child genotype is or genotypes are formed. Many techniques for this process exists such as one-point crossover, two-point crossover, cut and splice crossover, uniform crossover and half uniform crossover, three parent crossover methods. In one-point crossover method genotypes g_i and g_j of *l* length are both split at point *m* such that two new children are created where the first child has the genotype of $g_{i_{1\to m}}$ and $g_{j_{(m+1)\to l}}$ combined and the second child has the genotype of $g_{j_{1\to m}}$ and $g_{i_{(m+1)\to l}}$ essentially swapping the genes of the two parents at point *m*. In two-point crossover method genotypes g_i and g_j of *l* length are both split at point *m* and *n* such that two new children are created where the first child has the genotype of $g_{i_{1\to m}}$ and $g_{j_{(m+1)\to n}}$ and $g_{i_{(n+1)\to l}}$ combined and the second child has the

genotype of $g_{j_{1\to m}}$ and $g_{i_{(m+1)\to n}}$ and $g_{j_{(n+1)\to l}}$ essentially swapping only a segment of the genes of the two parents. In cut and splice method genotype g_i of l length is split at point m and genotype g_j of l length is split at point n where $m \neq n$ such that two new children are created where the first child has the genotype of $g_{i_{1\to m}}$ and $g_{j_{(n+1)\to l}}$ combined and the second child has the genotype of $g_{j_{1\to n}}$ and $g_{i_{(m+1)\to l}}$ essentially swapping segments of the genes of the two parents at different points forming gene lengths different from both parents. In both uniform and half uniform crossover method unlike prior methods genotypes g_i and g_j of l length are allowed to exchange genes rather than segment(s) of genes. In uniform crossover method genes are swapped with a fixed ratio and probability such that the same number of from g_i and g_j are swapped maintaining length l. Unlike the uniform crossover method, in half uniform crossover method exactly half of the non-matching genes are swapped instead. In three parent crossover method genotypes g_i, g_j and g_k of length l are randomly chosen and the child is derived from the three parents such that for each gene of the three parents the most common occurring gene is used to construct the child gene of length l.

A *selection* operation determines which genes are to survive for the later generations using the fitness function. Depending on the implementation the parent genotypes can be dropped entirely at the end of each generation with only a certain number of most fit children being allowed to create offspring. In elitist models however the genotypes with the highest fitness values of the previous generation is retained preventing the loss of the best found solution which the children may not necessarily outperform. There are a number of approaches in treating the fitness values from the fitness function such as: sorting, normalization, accumulation and thresholding. In sorting the fitness values are sorted without any sort of modification. In normalization each fitness value is divided by the sum of all

fitness values providing normalized fitness values. In accumulation a normalized fitness value of the fitness values of the current genotype and the fitness values from the parent genotypes from past generation(s) of the genotype. In thresholding only genotypes that have a fitness value above a certain threshold are retained.

```
bestVector  : best ID vector composition(s)
type        : type of unit of fitness being searched
generation  : current generation of ID vectors
hist        : history of previously tested ID vectors
classifier  : pool of classifiers
genRem      : number of generations remaining
trainSet    : training set
testSet     : test set
enum {Col, Row, ColRow, Acc}
Funciton bestVector = GeneticAlg(type, generation, hist, classifier, genRem,
trainSet, testSet, numClass)
ForEach i:generaiton
  If (type == Acc)
    Score(i)= ValAcc(generation(i) ,classifier, trainSet, testSet);
  Else
    Solutions=Learn(generation(i), classifier, trainSet);
    If (type == Col)
      Score(i) = ColDist(Solutions);
    Else if (type == Row)
      Score(i) = RowDist(Solutions);
    Else if (type == ColRow)
      Score(i) = ColDist(Solutions) * RowDist(Solutions);
    EndIf
  EndIf
End ForEach
generation = EliteSort(generation,Score);
If (genRem > 0)
  hist = UpdateHist(generation, hist);
  end = length(generation);
  generation(end) = Crossover(generation(1), generation(2), hist);
  generation(end-1) = Crossover(generation(1), generation(3), hist);
  For (j = 2; j < floor(end/2); j++)
    generation(end-j) = Crossover(generation(j), generation(j+1), hist);
  EndFor
  bestVector = GeneticAlg(type, generation, hist, classifier, (genRem-1),
trainSet, testSet, numClass);
Else
  BestIDs=MaxScoreIDs(Score);
  ForEach k:BestIDs
    bestVector(k) = generation(BestIDs(k));
  End ForEach
EndIf
EndFunciton
```

Figure 12: Pseudocode of the Genetic Algorithm implementation

In our implementation we use uniform crossover method and elitist sorting selection operations (Figure 12). We determine our genotype population size based on the number of

classes from the multi-class classification problem. Our Initial attempt determined the size of our population as $p = 100 \times 3^{(k-3)} + 1$ for 21 generations (See Figure 13). We attempted such a dynamic size for the population to circumvent known limitations of genetic algorithms such as the convergence on local optima (Horn & Goldberg, 1994). However it was quickly apparent that this was not feasible due to the time constraints. Hence we restricted our population to 59 genotypes for 3-class problems and 101 genotypes for higher class classification problems. We realize this is an insufficient amount (See Figure 14) especially for higher class classification problems where the portion of the search space investigated rapidly becomes insignificant but our goal is to demonstrate whether our optimization attempt yields better results not to find the optimal solution for each dataset. Mind that even with such a restricted population and only a single 10-fold run with 21 generations, the 8-class dataset ecoli had taken about 41 hours to process.

Multi-class	Binary Problems	Search Space Size	Initial Population	Total Population	Percent of Search Space
3	3	243	101	1,101	453.0864197%
4	7	16,807	301	3,301	19.64062593%
5	15	759,375	901	9,901	1.303835391%
6	31	28,629,151	2,701	29,701	0.103743908%
7	63	992,436,543	8,101	89,101	0.008978005%
8	127	33,038,369,407	24,301	267,301	0.000809062%

Figure 13: Formulated genetic search size for a pool of five classifiers and 21 generations

Multi-class	Binary Problems	Search Space Size	Initial Population	Total Population	Percent of Search Space
3	3	243	59	639	262.9629629%
4	7	16,807	101	1,101	6.550841911%
5	15	759,375	101	1,101	0.144987654%
6	31	28,629,151	101	1,101	0.003845731%
7	63	992,436,543	101	1,101	0.000110939%
8	127	33,038,369,407	101	1,101	0.000003332%

Figure 14: Restricted genetic search size for a pool of five classifiers and 21 generations

We populate our *population matrix* $p \times b$ with the c homogeneous genotypes and then append $p - c$ uniformly distributed pseudorandom integers (using MATLAB function randi) corresponding to the algorithm ID range. The inclusion of the homogeneous genotypes allows the genetic algorithm to also consider Homogeneous ID vector compositions essentially insuring that the heterogeneous result will never underperform homogeneous results in terms of the unit of fitness considered. The entire initial population matrix is ranked through the fitness function and sorted.

Our crossover function is a slightly modified version of the formal definition of the method such that it only produces one child instead of two. This is done primarily to save time as training task takes a considerable amount of time. Each crossover operation has genotypes g_i and g_j as input and g_c as output. All genotypes are of l length where $l = b = 2^{(k-1)} - 1$, the number of binary class classification problems. Crossover function determines a random value n where $l/3 < n < l$ and then picks n many genes from g_i and then picks the remaining $l - n$ genes from g_j forming g_c. Due to the random nature of the crossover operation we implemented a history of the genotype population that were attempted before to avoid repetitions which not only insures more genes are actually tested but also improves performance by avoiding redundant operations. Before returning g_c as output, it is compared to the current and past population and if g_c is already on it, the crossover operation thus-far between the two parents is repeated until either all combinations between the two parents are attempted or if a suitable child is found.

3.3 k-fold Cross Validation and ID Vector Selection

We observed a profound impact of the training instances on the selected ID vector such that it was imperative for us to overcome this overfitting problem. Furthermore some

datasets – particularly those with more symmetric instance distribution – even registered multiple ID vector solutions with the same value of unit of fitness requiring further refinement of our ID vector selection.

In order to generalize our results and in order to overcome this overfitting problem we use *k-fold Cross Validation* which is a statistical model validation technique that randomly divides the sample of data into *k* equally sized subsamples. Each subsample is then retained for validation and the other *k*-1 samples are collectively used as the training set. This process is repeated for each subsample.

In our implementation we use 10-fold cross validation by consistently applying the exhaustive search or the genetic algorithm search for each fold. Each iteration returns an ID vector or vectors that best performed as far as the unit of fitness is concerned. ID vector(s) from each iteration may or may not agree with each other.

Initially we attempted frequency based approaches such as frequency distribution but this quickly became problematic when the number of possible candidate ID vectors made the usability of frequency based statistical approaches difficult. Instead we implemented a second 10-fold cross validation to exhaustively process all of the ID vectors returned by the previous 10-fold cross validation run. Among them we choose the ID vector(s) with the highest average validation accuracy for all of the iterations.

It is possible for this second 10-fold cross validation to also return multiple ID vectors with identical validation accuracies. If so we intuitively choose the ID vector with the greatest variety to minimize the potential problems Homogeneous ECOC suffers as previously discussed. This also maximizes the diversity of the binary classifiers the IDs represent.

3.4 Conclusion

We have thus far discussed the complications arisen due to the heterogeneous treatment of ECOC and our proposed solutions to said complications. In the next chapter we discuss our experiments to evaluate the validity of the Heterogeneous ECOC approach. We first discuss the properties of thirteen datasets we use for our experiments where we briefly explain each dataset. We then examine the performance of exhaustive search with column distance as the unit of fitness on the nine three and four class datasets. We do not run exhaustive search on other units of fitness or more datasets due to performance and time considerations. Afterwards we examine the performance of the genetic algorithm with all four units of fitness (column distance, row distance, column × row distance, validation accuracy). We discuss our findings by comparing the four unit of fitness first of the Heterogeneous ECOC approach first with each other and afterwards with Homogeneous ECOC serving as our benchmark.

4 Experimental results

As discussed in the previous chapter, our main task is an optimization problem where we choose an ID vector of length *b* with each element having *c* possible values corresponding to the selection of *c* binary classifiers for *b* binary classification problems using a variety of datasets. We use 10-fold cross-validation to statistically verify our results as well as to select the aforementioned ID vector.

We also use 10-fold cross-validation on homogeneous ECOC to calculate statistically relevant validation accuracy for each of the *c* classifiers as our benchmark and pick the homogeneous classifier distribution with the highest validation accuracy. We then compare this validation accuracy of Homogeneous ECOC with the validation accuracy of the classifier composition of the Heterogeneous ECOC approach depicted by the ID vector.

4.1 Datasets Processed

We have chosen thirteen different datasets with varying degree of complexity to test our approach (See Figure 15). We acquired all of our datasets from University of California, Irvine (UCI) Machine Learning Repository and credited the authors of the datasets in Figure 40 (Appendix A – Sources of All Datasets).

We preferred datasets without missing attribute values as some binary classifiers are unable to compensate for this. As a pre-processing step we moved the class column as the last column, we removed any statistically irrelevant columns (such as ID columns) and we shuffled the columns to have more uniform distribution of all classes for the cross validation process.

Dataset name	Number of classes	Number of features	Number of instances	Dataset classification problem
balance	3	4	625	Classify tipping of balance scale using weight and distance
cmc	3	9	1,473	Classify contraceptive method choice using demographic and socio-economic characteristics[2]
iris	3	4	150	Classify the type of iris flower using sepal and petal characteristics
thyroid	3	5	215	Classify otolaryngology patients using thyroid gland features
vertebral	3	6	310	Classify orthopaedic patients using biomechanical features
wine	3	13	178	Classify the origin of wines using chemical analysis
car	4	6	1,728	Classify acceptability of cars using price and technical characteristics
lymph	4	18	148	Classify lymph patients using lymphography features
vehicle	4	18	846	Classify images of vehicles using features extracted from silhouettes
derm	6	34	358	Classify patients of erythemato-squamous diseases using dermatology features
glass	6	9	214	Classify glass types in crime scenes using chemical composition
zoo	7	16	101	Classify animals in zoos using anatomic features
ecoli	8	7	336	Classify E. coli bacteria using protein localization sites

Figure 15: Statistics of the datasets used

4.1.1 Balance Scale Dataset

This artificially generated multivariate datasets (balance) with 625 instances models psychological experiment results using a balance scale. Each instance with four features has values of the two sides of the scale for mass M_1 and M_2 with values ranging between one and five and for distance a and b with values also ranging between one and five. The three class values represent if the scale is remaining balanced (49 instances), tipping to the right (288 instances), or tipping to the left (288 instances). The dataset basically tries to determine the mechanical advantage formula $M_1 a = M_2 b$.

4.1.2 Contraceptive Method Choice Dataset

This multivariate dataset (cmc) with 1,473 instances is a subset of the 1987 National Indonesian Contraceptive Prevalence Survey which interviewed married women whom were either not pregnant or did not know if they were pregnant during the interview. Each instance

[2] 1987 National Indonesia Contraceptive Prevalence Survey

has nine demographic and socio-economic characteristics as the nine features: the age of the wife, education of the wife and husband each with a categorical value ranging between one and four, number of children, wife's religion (if Islam or not), wife's employment (if working or not), husbands occupation with a categorical value ranging between one and four, standard of living index with a categorical value ranging between one and four, media exposure (good or not good). The three class values represent the current contraceptive method choice which could either be none used (629 instances), long-term methods (333 instances), or short-term methods (511 instances).

4.1.3 Iris Flower Dataset

This multivariate dataset (iris) with 150 instances is the well-known database collected by Edgar Anderson and introduced by Sir Ronald Fisher as an example of discriminant analysis. Each instance has flower's characteristics as the four features with numeric values in centimeters: sepal length, a sepal width, a petal length and a petal width. The three class values are the three species of iris flower Iris Setosa, Iris Versicolour and Iris Virginica – each with 50 instances.

4.1.4 Thyroid Gland Dataset

This multivariate dataset (thyroid) with 215 instances representing diagnosis of otolaryngology patients. Each instance has the five features with numerical values: T3-resin uptake test (percentage), total Serum thyroxin as measured by the isotopic displacement method, total serum triiodothyronine as measured by radioimmuno assay, basal thyroid-stimulating hormone (TSH) as measured by radioimmuno assay, maximal absolute difference of TSH value after injection of 200 micro grams of thyrotropin-releasing hormone as compared to the basal value. The three class values represent the diagnosis of the thyroid

gland of otolaryngology patients which could either be normal (150 instances), hyper functioning (35 instances), hypo functioning (30 instances).

4.1.5 Vertebral Column Dataset

This multivariate dataset (vertebral) with 310 instances represent vertebral column (backbone) of orthopaedic patients. Each instance represents each patient's biomechanical characteristics with numerical values as the six features: pelvic incidence, pelvic tilt, lumbar lordosis angle, sacral slope, pelvic radius, and degree spondylolisthesis. The three class values represent the diagnosis of the patients which could either be disk hernia (60 instances), spondylolisthesis (150 instances), or normal (100 instances).

4.1.6 Wine Origin Dataset

This multivariate dataset (wine) with 178 instances represent wines in the same geographic location in Italy. Each instance holds chemical composition as the thirteen features with numerical values: alcohol , malic acid, ash, alcalinity of ash, magnesium, total phenols, flavonoids, nonflavanoid phenols, proanthocyanins, color intensity, hue, OD280/OD315 of diluted wines, proline. The three class values represent the cultivars which could either be class 1 (59 instances), class 2 (71 instances) or class 3 (48 instances). The actual identity of the cultivars is not known.

4.1.7 Car Evaluation Dataset

This multivariate dataset (car) with 1,728 instances represent car evaluations. Each instance has as the six features: buying price, price of maintenance, number of doors with four possible categorical values each, person capacity, size of luggage boot, estimated safety with three possible categorical values each. The four class values represent desirability of

cars which could either be unacceptable (1,210 instances), acceptable (384 instances), good (69 instances), or very good (65 instances).

4.1.8 Lymphography Dataset

This multivariate dataset (lymph) with 148 instances representing diagnosis of lymph patients. Each instance has the 18 features: block of affere, bl. of lymph. c, bl. of lymph. s, by pass, extravasates, regeneration of, early uptake in, dislocation of, and exclusion of no each with binary values, changes in lym., and special forms with three categorical values, lymphatics, lym.nodes dimin and lym.nodes enlar, defect in node, and changes in node each with four categorical values, changes in stru, and no. of nodes each with eight categorical values. The four class values represent the diagnosis which could either be normal find (2 instances), metastases (81 instances), malign lymph (61 instances), or fibrosis (4 instances).

4.1.9 Vehicle Silhouettes Dataset

This multivariate dataset (vehicle) with 846 instances representing features of vehicle silhouettes. Each instance has the 18 features: compactness, circularity, distance circularity, radius ratio, pr.axis aspect ratio, maximum length aspect ratio, scatter ratio, elongatedness, pr.axis rectangularity, maximum length rectangularity, scaled variance, along major axis, scaled variance along minor axis, scaled radius of gyration, skewness about major axis, skewness about minor axis, kurtosis about minor axis, kurtosis about major axis, hollows ratio. The four class values represent the vehicle types which could either be opel (212 instances), saab (217 instances), bus (218 instances), or van (199 instances).

4.1.10 Dermatology Dataset

This multivariate dataset (derm) with 358 instances represent erythemato-squamous diseases. Each instance has the 34 features: family history with a binary value, erythema, scaling, definite borders, itching, koebner phenomenon, polygonal papules, follicular papules, oral mucosal involvement, knee and elbow involvement, scalp involvement, melanin incontinence, eosinophils in the infiltrate, PNL infiltrate, fibrosis of the papillary dermis, exocytosis, acanthosis, hyperkeratosis, parakeratosis, clubbing of the rete ridges, elongation of the rete ridges, thinning of the suprapapillary epidermis, spongiform pustule, munro microabcess, focal hypergranulosis, disappearance of the granular layer, vacuolisation and damage of basal layer, spongiosis, saw-tooth appearance of retes, follicular horn plug, perifollicular parakeratosis, inflammatory monoluclear infiltrate, band-like infiltrate each with four categorical values, age with a numeric value. The six class values represent the diagnosis which could either be psoriasis (112 instances), seboreic dermatitis (61 instances), lichen planus (72 instances), pityriasis rosea (49 instances), cronic dermatitis (52 instances), pityriasis rubra pilaris (20 instances).

4.1.11 Crime Scene Glass Identification Dataset

This multivariate dataset (glass) with 214 instances represent forensic glass evidence in crime scenes. Each instance has the nine features with numerical values: refractive index, sodium, magnesium, aluminum, silicon, potassium, calcium, barium, iron. The six class values represent the glass types which could be building windows float processed (70 instances), building windows non float processed (76 instances), vehicle windows float processed (17 instances), vehicle windows non float processed (0 instances/not in this dataset), containers (13 instances), tableware (9 instances), headlamps (29 instances). The

study that prompted this dataset was prompted by criminological investigation procedures requiring glass to be identified on the crime scene to be processed as evidence.

4.1.12 Zoo Dataset

This artificial multivariate dataset (zoo) with 101 instances represent animals. Each instance has the sixteen features: hair, feathers, eggs, milk, airborne, aquatic, predator, toothed, backbone, breathes, venomous, fins, tail, domestic, catsize each with binary values, legs with six categorical values. The seven class values represent the seven groups of animals which can ben class 1 (41 instances), class 2 (20 instances), class 3 (5 instances), class 4 (13 instances), class 5 (4 instances), class 6 (8 instances), class 7 (10 instances).

4.1.13 Escherichia Coli Protein Localization Sites Dataset

This multivariate dataset (ecoli) with 336 instances represent protein localization of Escherichia coli (e. coli) bacteria. Each instance has the seven feature: lip (Von Heijne's Signal Peptidase II consensus sequence score), and chg (Presence of charge on N-terminus of predicted lipoproteins) each with binary values, mcg (McGeoch's method for signal sequence recognition), gvh (Von Heijne's method for signal sequence recognition), aac (score of discriminant analysis of the amino acid content of outer membrane and periplasmic proteins), alm1 (score of the ALOM membrane spanning region prediction program), and alm2 (score of ALOM program after excluding putative cleavable signal regions from the sequence) each with numeric values. The eight class values represent cytoplasm (143 instances), inner membrane without signal sequence (77 classes), periplasm (52 instances), inner membrane with uncleavable signal sequence (35 instances), outer membrane (20 instances), outer membrane lipoprotein (5 instances), inner membrane lipoprotein (2 classes), or inner membrane with cleavable signal sequence (2 instances).

4.2 Analysis of Experiments

We attempted solving the optimization problem using different units of fitness in an attempt to find which fitness approach provides the ID vector with the highest validation accuracy. As discussed in chapter 3.1, we use Hamming to calculate distances between class predictions based on training set instances.

4.2.1 Exhaustive Search for Column Distance

We decided to use column distance (Chapter 3.1.1) as our first unit of fitness as discussed in chapter 3.2.2. For performance reasons as well as due to the double exponential search space size, we did not apply this exhaustive approach for other competing units of fitness (See Figure 11). Furthermore, we only performed this exhaustive search on three and four class classification problems which comprised of nine datasets total. We calculated validation accuracy values ranging between 50.33% and 96.76% with an average of 73.21% (See Figure 16). For higher than four-class datasets, we exclusively use the genetic algorithm.

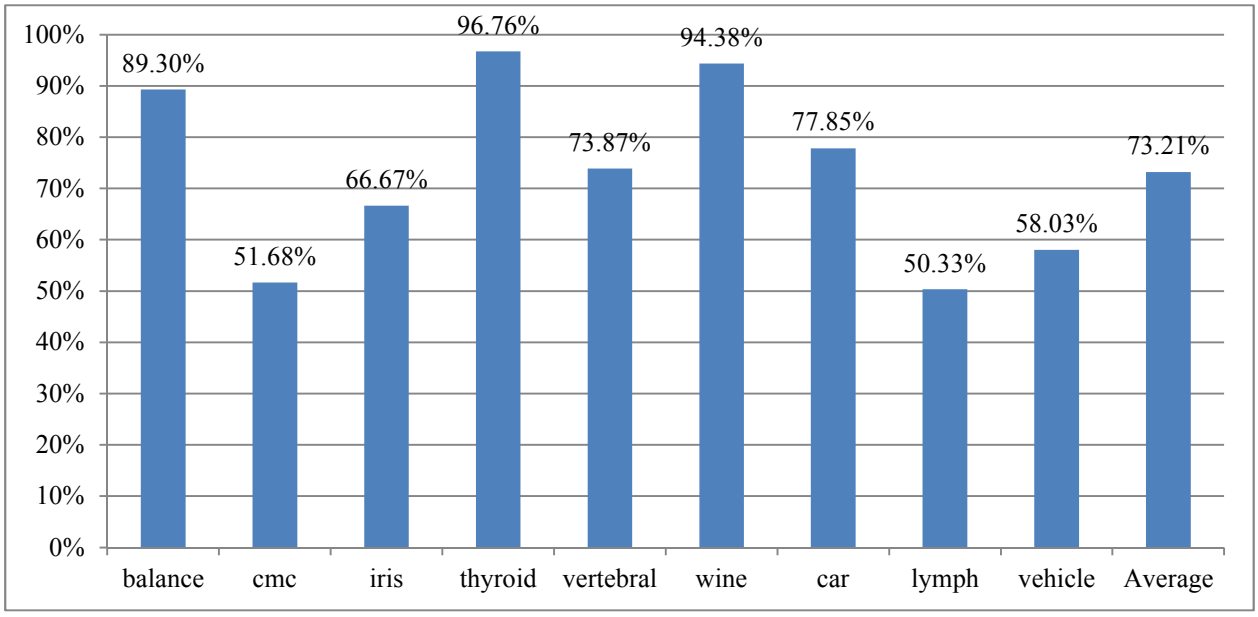

Figure 16: Accuracies for exhaustive search of column distances

4.2.2 Column Distance as GA Fitness Function

We also use column distance (Chapter 3.1.1), the same unit of fitness, as the genetic algorithm fitness function such that the fitness value corresponded to the column distance and the genetic algorithm attempts to maximize this column distance as discussed in chapter 3.2.3. We perform this genetic search for all of the datasets – including the ones solved by an exhaustive search. As it may be observed, we are indeed repeating our column distance based search for the three and four class datasets. We do so in an attempt to observe if our genetic algorithm's overall performance is comparable to exhaustive results. We will discuss such comparisons in chapter 4.3.

We observed an increase in average validation accuracy with this approach on some datasets such as the lymph dataset (See Figure 19). However, this was not always the case with most datasets as visible with car and vehicle datasets (See Figure 18 and Figure 20). On all datasets we even observed fluctuations in validation accuracy for individual iterations of the 10-fold cross validation as the genetic algorithms generations progressed where the column distance consistently increased. We calculated validation accuracy values ranging between 45.38% and 96.76% with an average of 76.57% (See Figure 17).

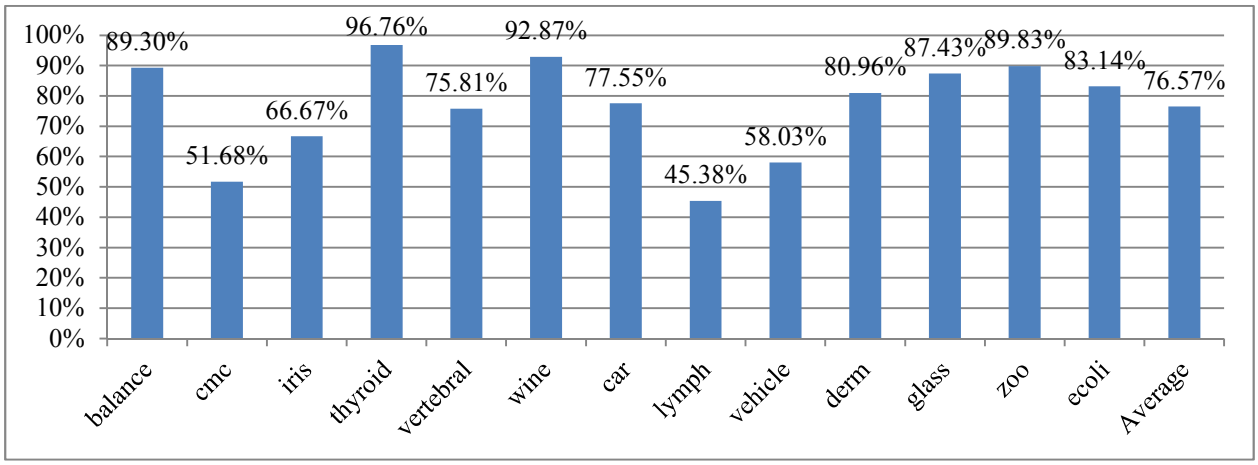

Figure 17: Accuracies for column distance as GA fitness function

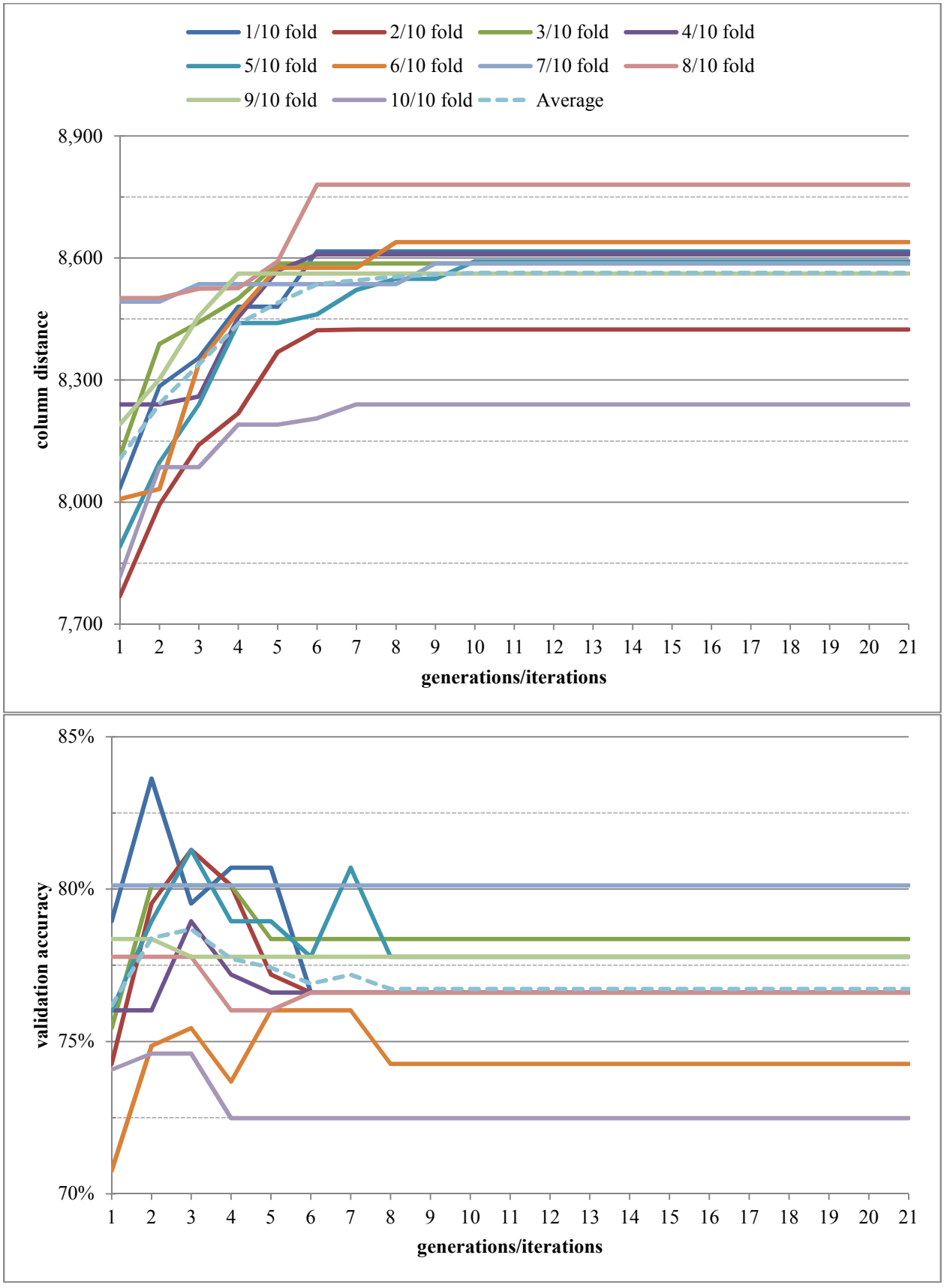

Figure 18: Column distance as fitness function for car dataset

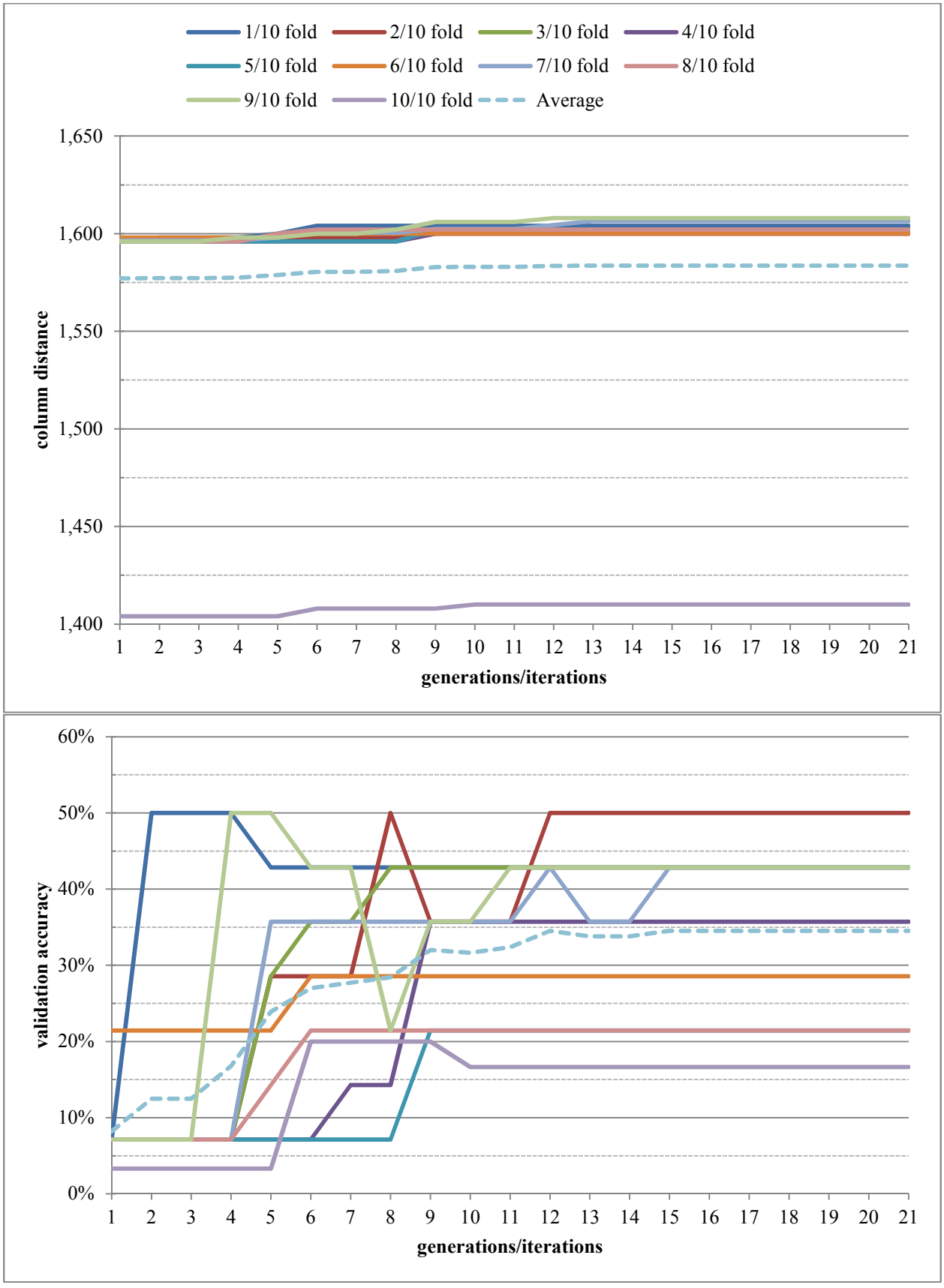

Figure 19: Column distance as fitness function for lymph dataset

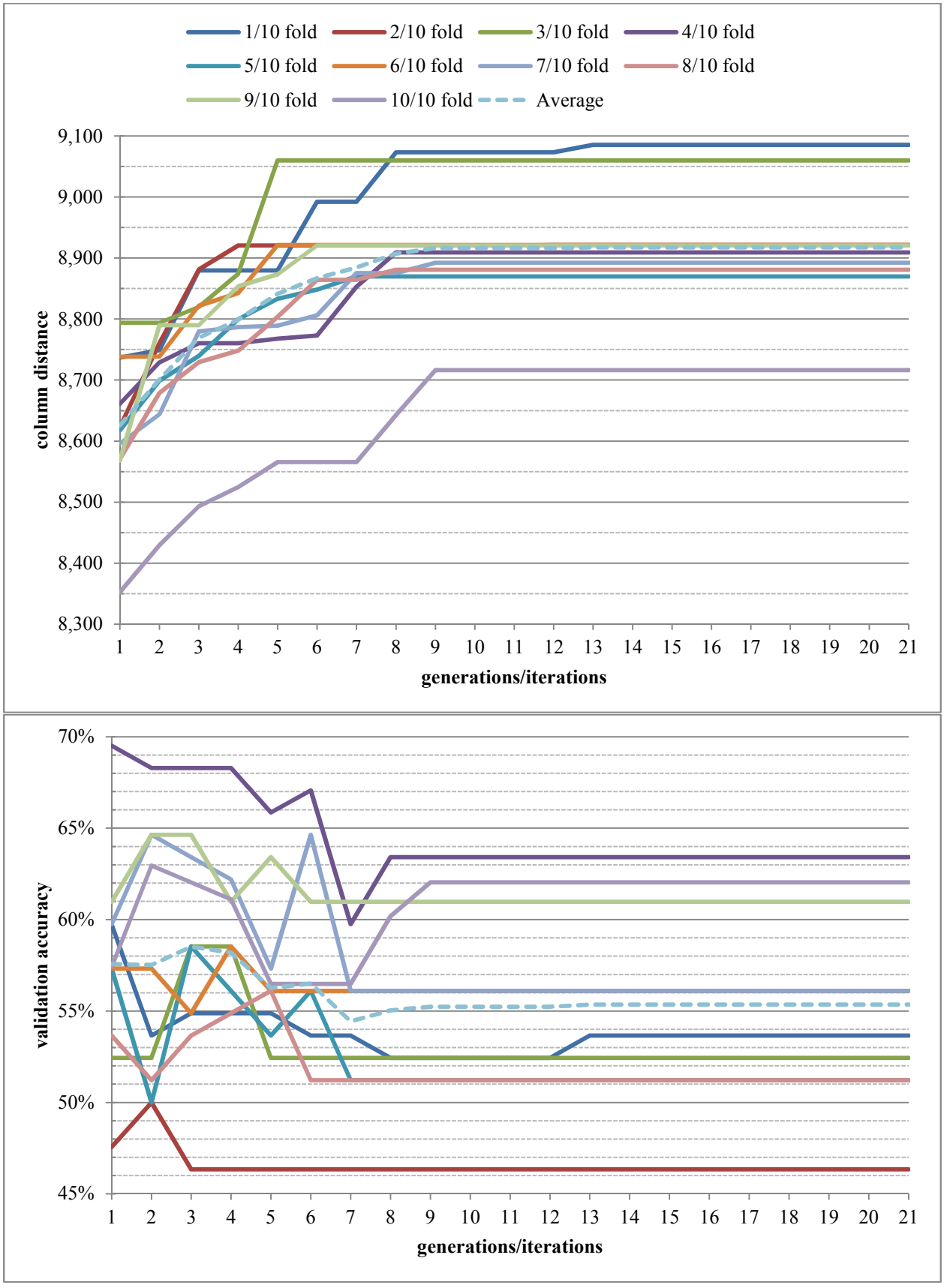

Figure 20: Column distance as fitness function for vehicle dataset

4.2.3 Row Distance as GA Fitness Function

As our second unit of fitness, we use row distance (Chapter 3.1.2) as the genetic algorithm fitness function such that the fitness value corresponded to the row distance and the genetic algorithm attempts to maximize this row distance as discussed in chapter 3.2.3. We perform this genetic search for all of the datasets.

We observed an increase in average validation accuracy with this approach on some datasets such as the vehicle dataset (See Figure 24). However, this was not always the case with most datasets as visible with car and lymph datasets (See Figure 22 and Figure 23). On all datasets we even observed fluctuations in validation accuracy for individual iterations of the 10-fold cross validation as the genetic algorithms generations progressed where the row distance consistently increased. We calculated validation accuracy values ranging between 38.17% and 98.95% with an average of 83.18% (See Figure 21).

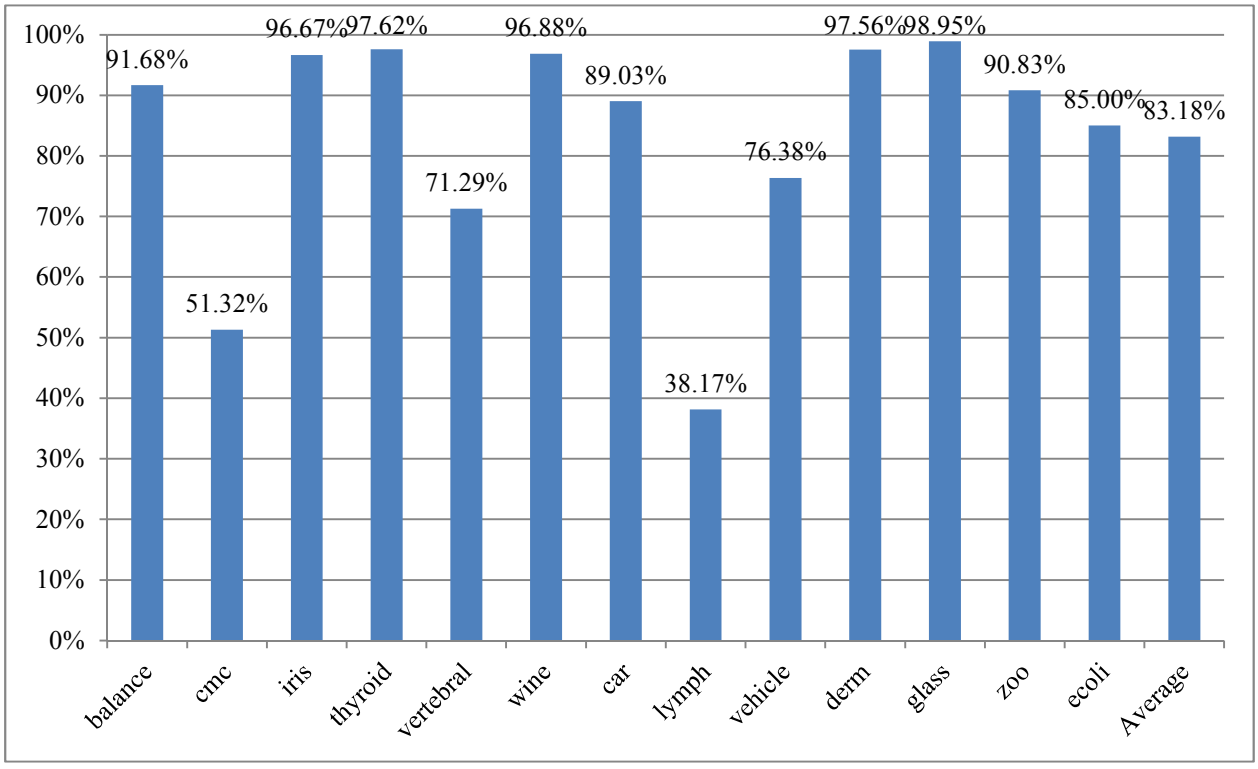

Figure 21: Accuracies for row distance as GA fitness function

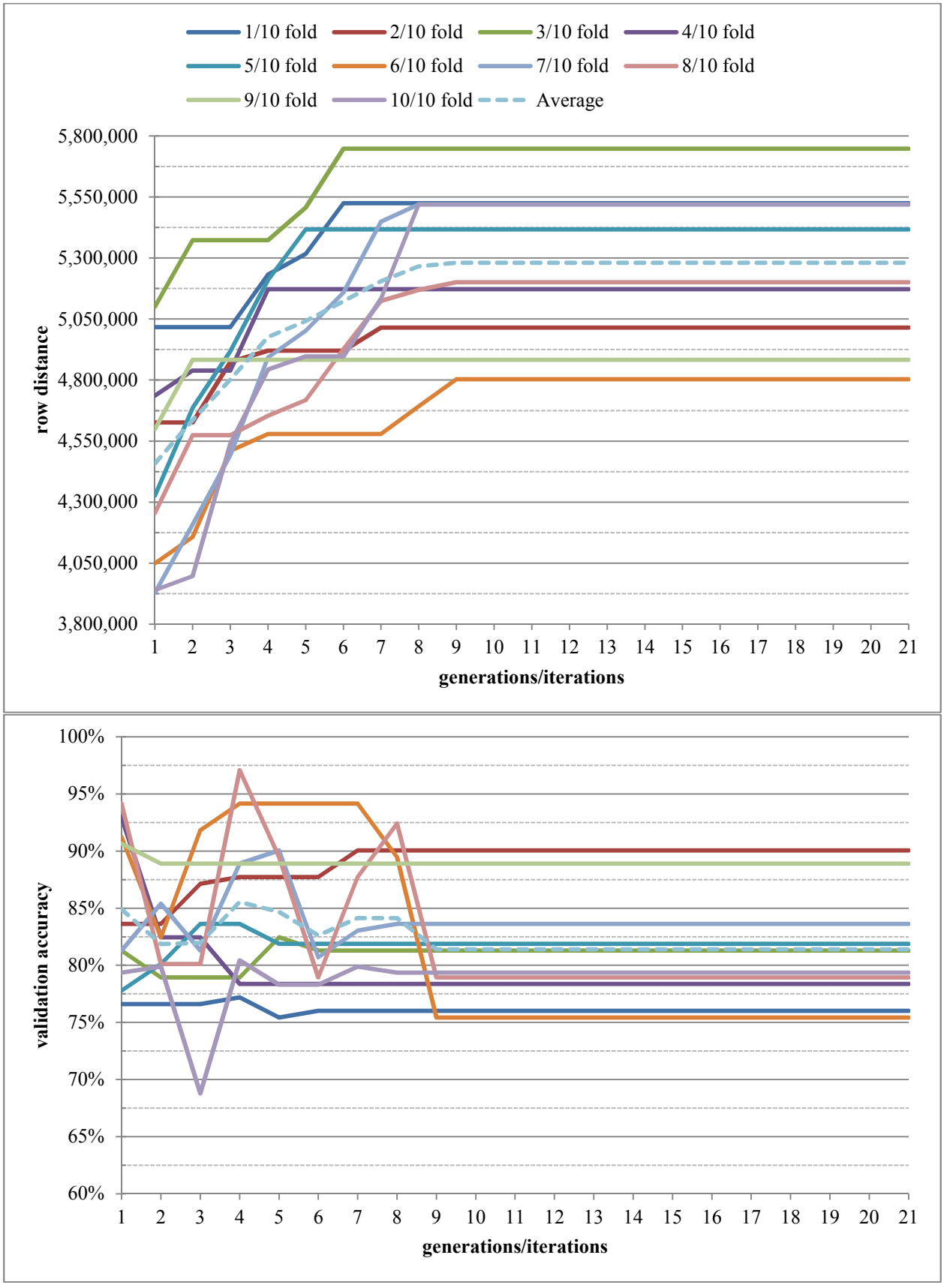

Figure 22: Row distance as fitness function for car dataset

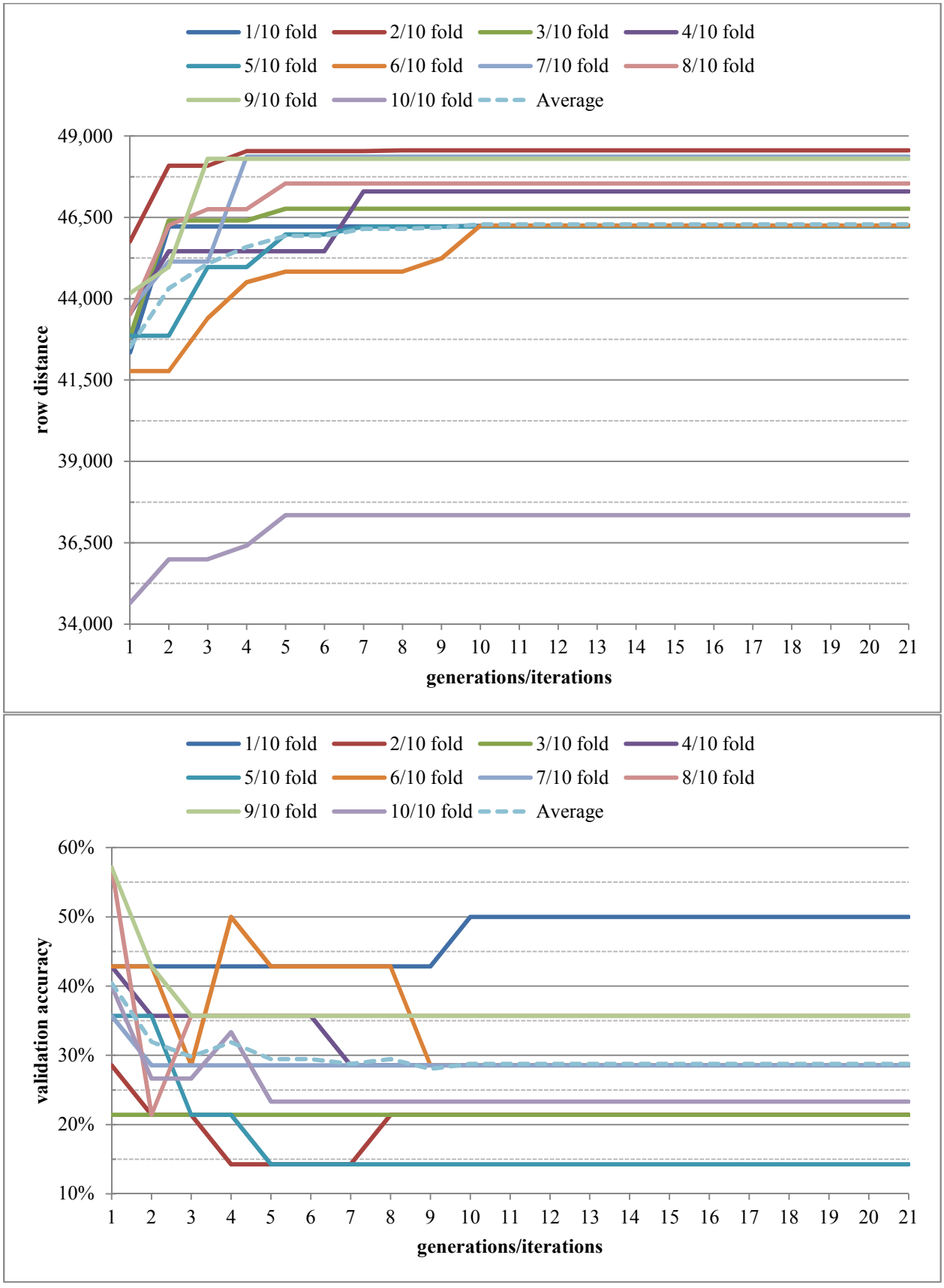

Figure 23: Row distance as fitness function for lymph dataset

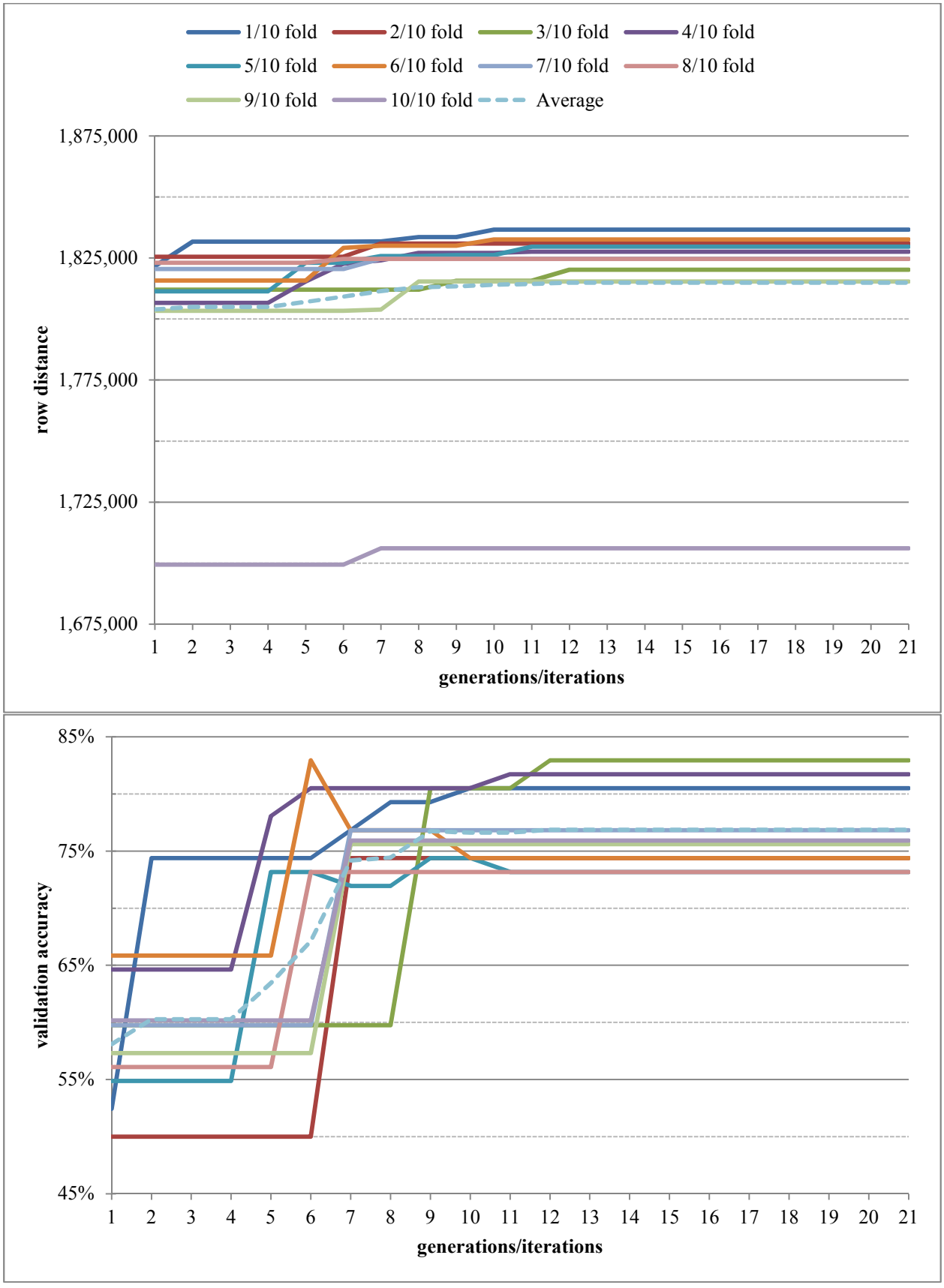

Figure 24: Row distance as fitness function for vehicle dataset

4.2.4 Column × Row Distance as GA Fitness Function

As our third unit of fitness, we use a multiplication of column and row distance or column × row distance (Chapter 3.1.3) as the genetic algorithm fitness function such that the fitness value corresponded to the multiplication of the two distance values and the genetic algorithm attempts to maximize this multiplication value as discussed in chapter 3.2.3. We perform this genetic search for all of the datasets.

We observed an increase in average validation accuracy with this approach on some datasets such as the vehicle dataset (See Figure 28). However, this was not always the case with most datasets as visible with car and lymph datasets (See Figure 26 and Figure 27). We even observed fluctuations in validation accuracy for individual iterations of the 10-fold cross validation as the genetic algorithms generations progressed where the multiplied value of the column and row distance consistently increased. We calculated validation accuracy values ranging between 36.62% and 98.95% with an average of 80.67% (See Figure 25).

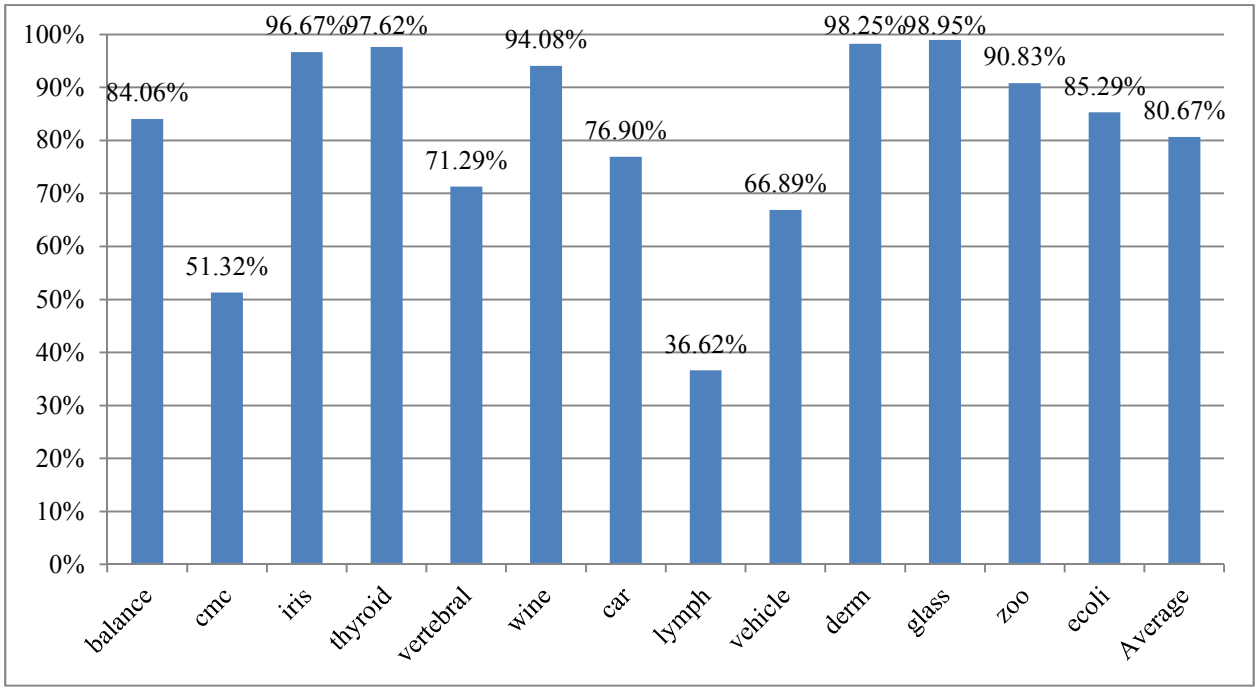

Figure 25: Accuracies for Column × row distance as GA fitness function

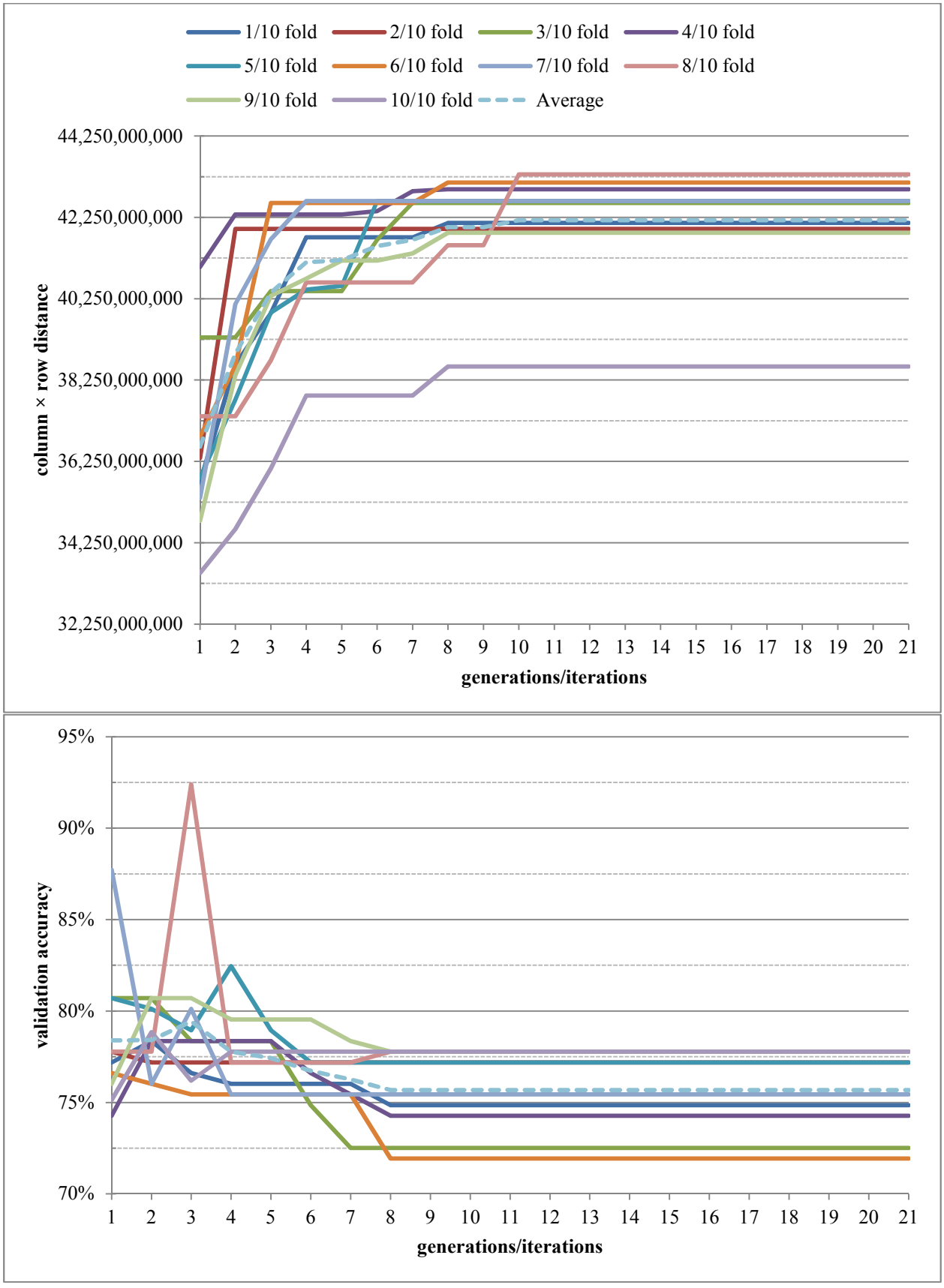

Figure 26: Column × row distance as fitness function for car dataset

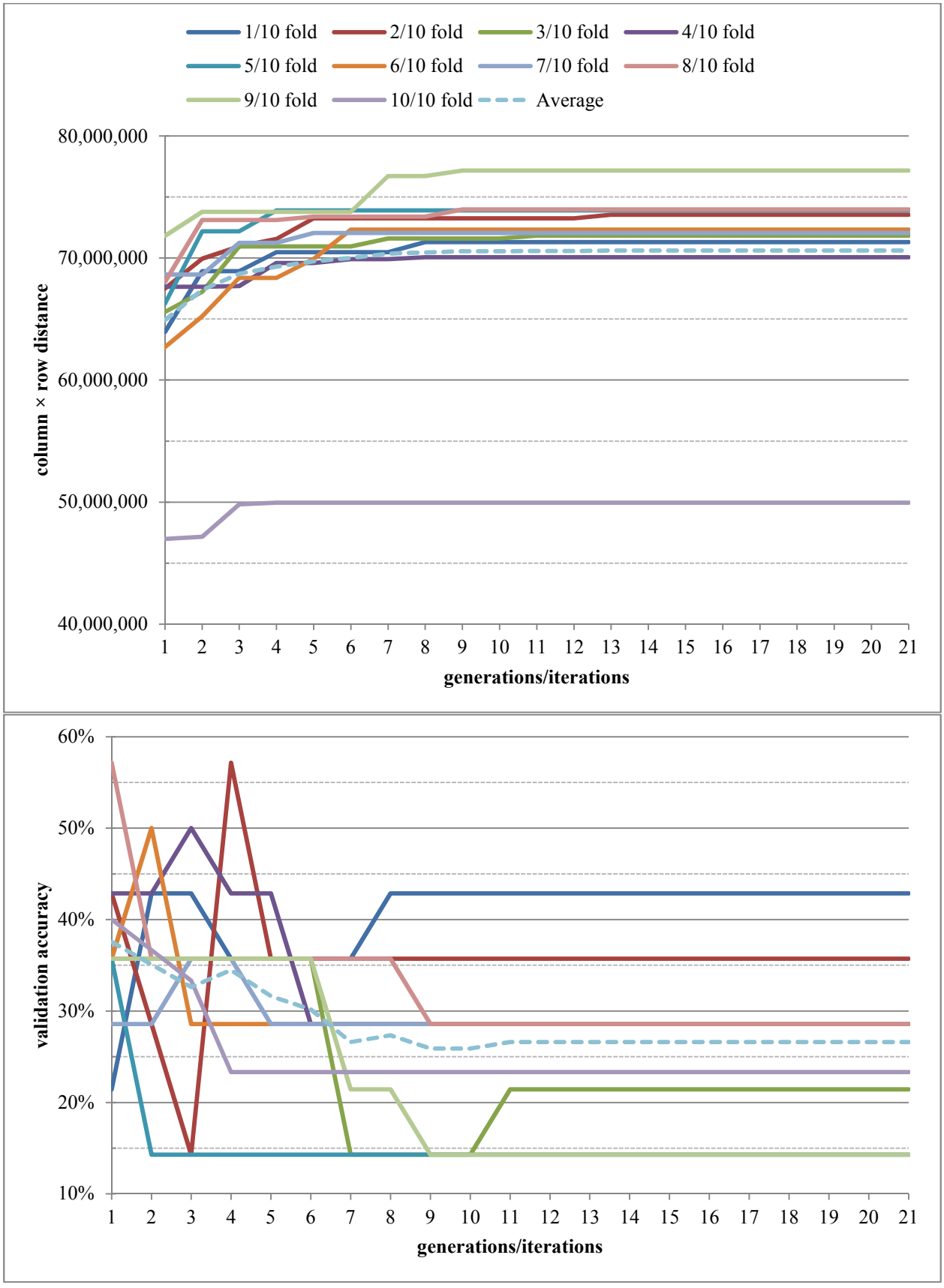

Figure 27: Column × row distance as fitness function for lymph dataset

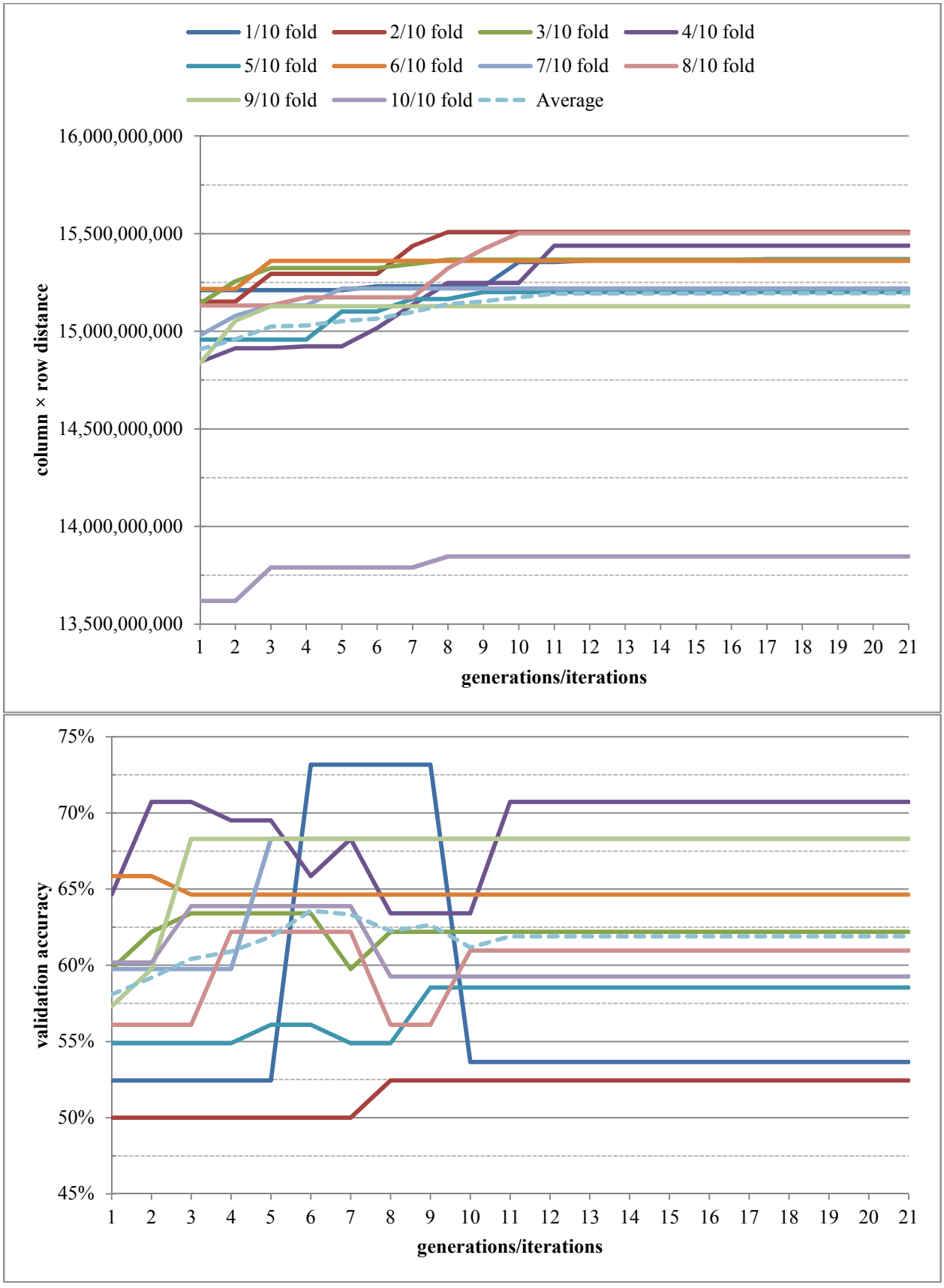

Figure 28: Column × row distance as fitness function for vehicle dataset

4.2.5 Validation Accuracy as GA Fitness Function

As our fourth unit of fitness, we use validation accuracy (Chapter 3.1.4) as the genetic algorithm fitness function such that the fitness value corresponded to the validation accuracy and the genetic algorithm attempts to maximize this multiplication value as discussed in chapter 3.2.3. We perform this genetic search for all of the datasets.

We observed an increase in average validation accuracy with this approach on all datasets except the three class balance dataset which remained retained its initial value. We also observed an increase in column distance on some datasets such as the car dataset (See Figure 30). However this was not the case with most datasets as visible with lymph and vehicle datasets (See Figure 31 and Figure 32). We however observed fluctuations in column distance for individual iterations of the 10-fold cross validation as the genetic algorithms generations progressed where the validation accuracy consistently increased. We calculated validation accuracy values ranging between 52.91% and 98.95% with an average of 86.91% (See Figure 29).

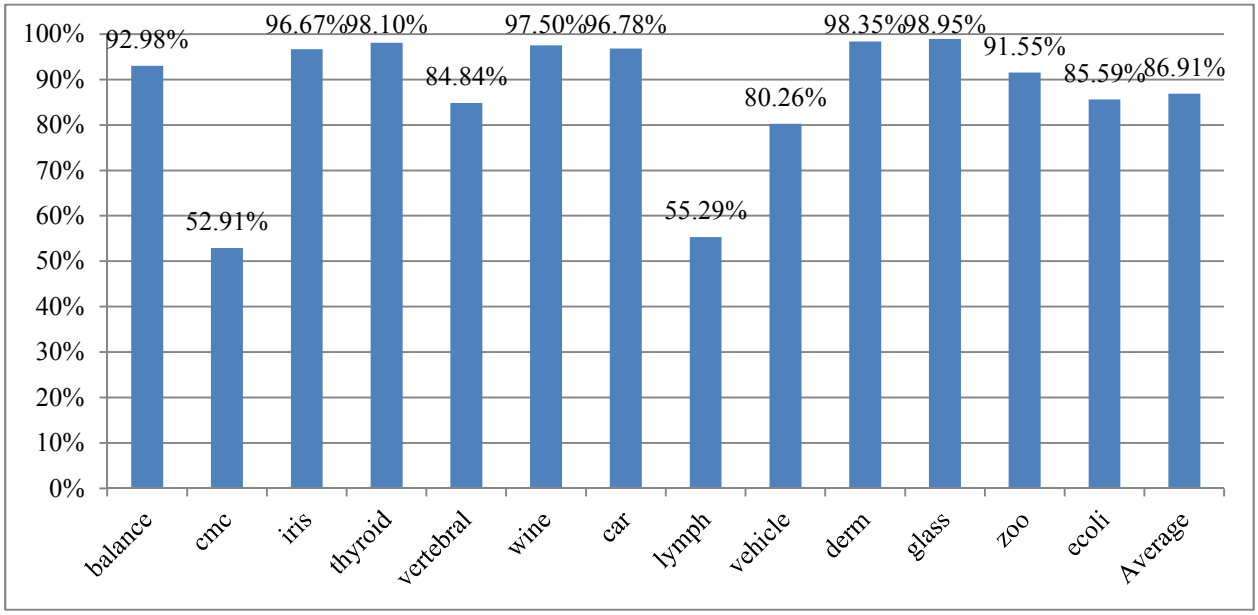

Figure 29: Accuracies for validation accuracy as GA fitness function

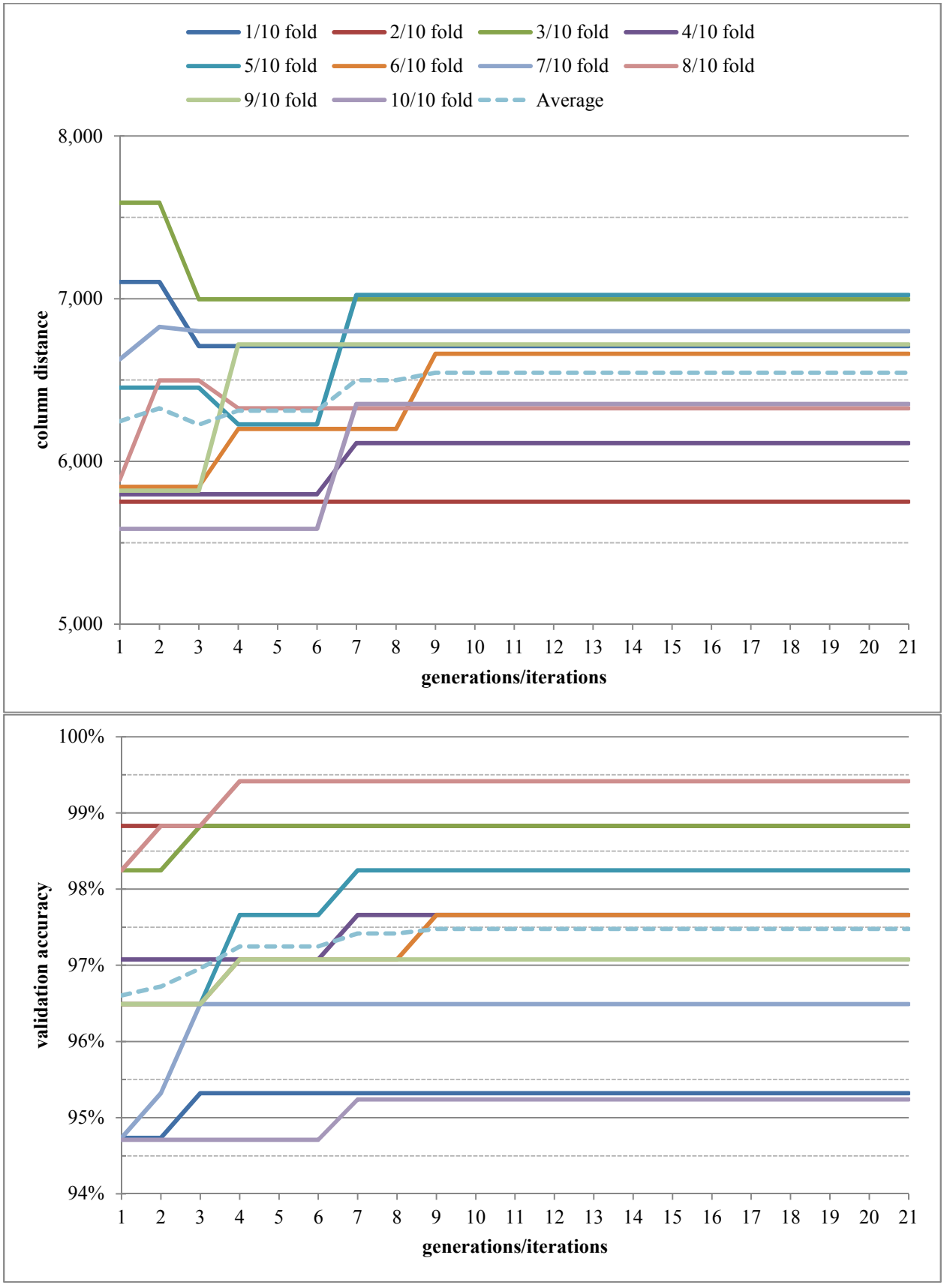

Figure 30: Accuracy as fitness function for car dataset

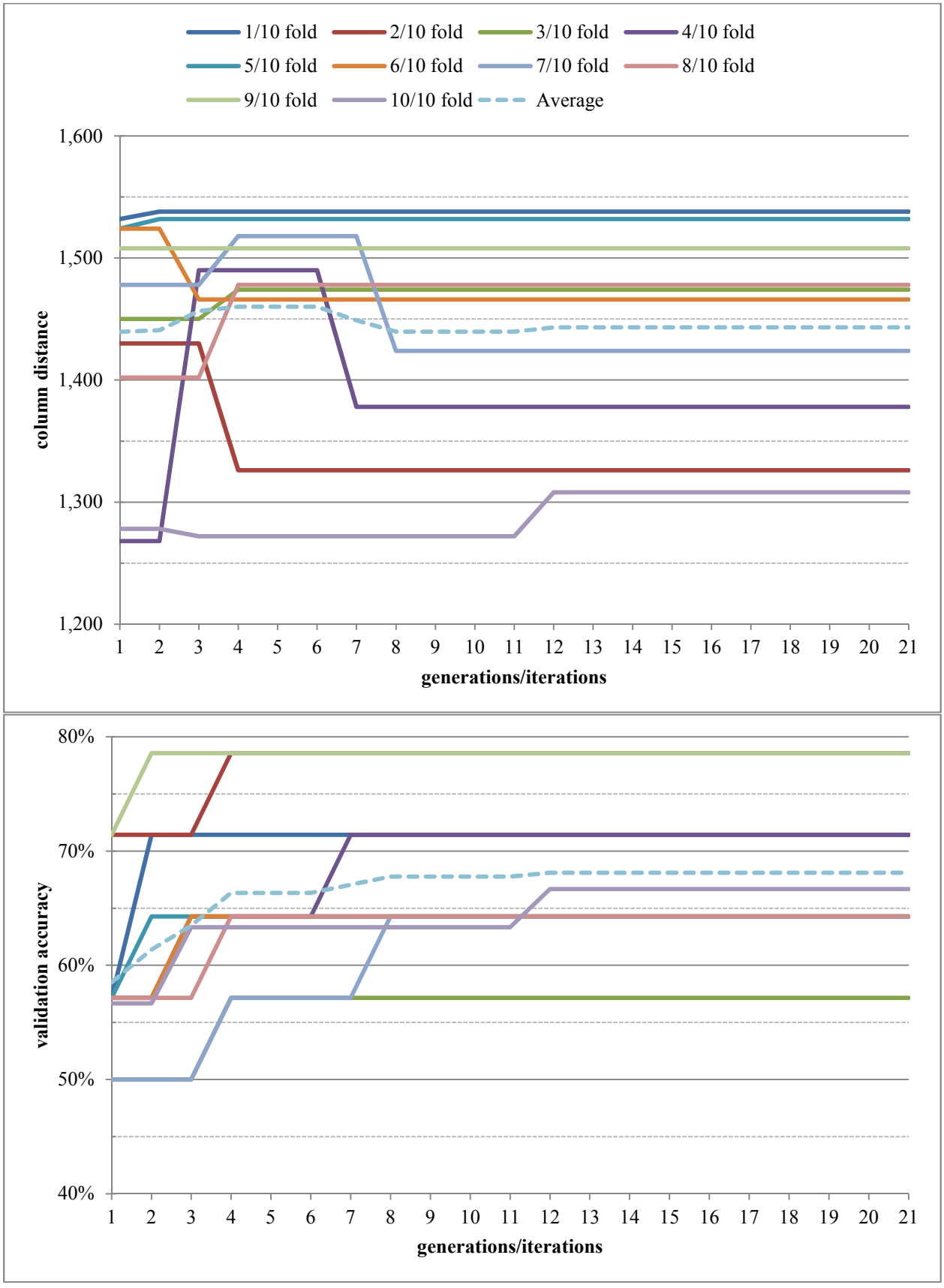

Figure 31: Accuracy as fitness function for lymph dataset

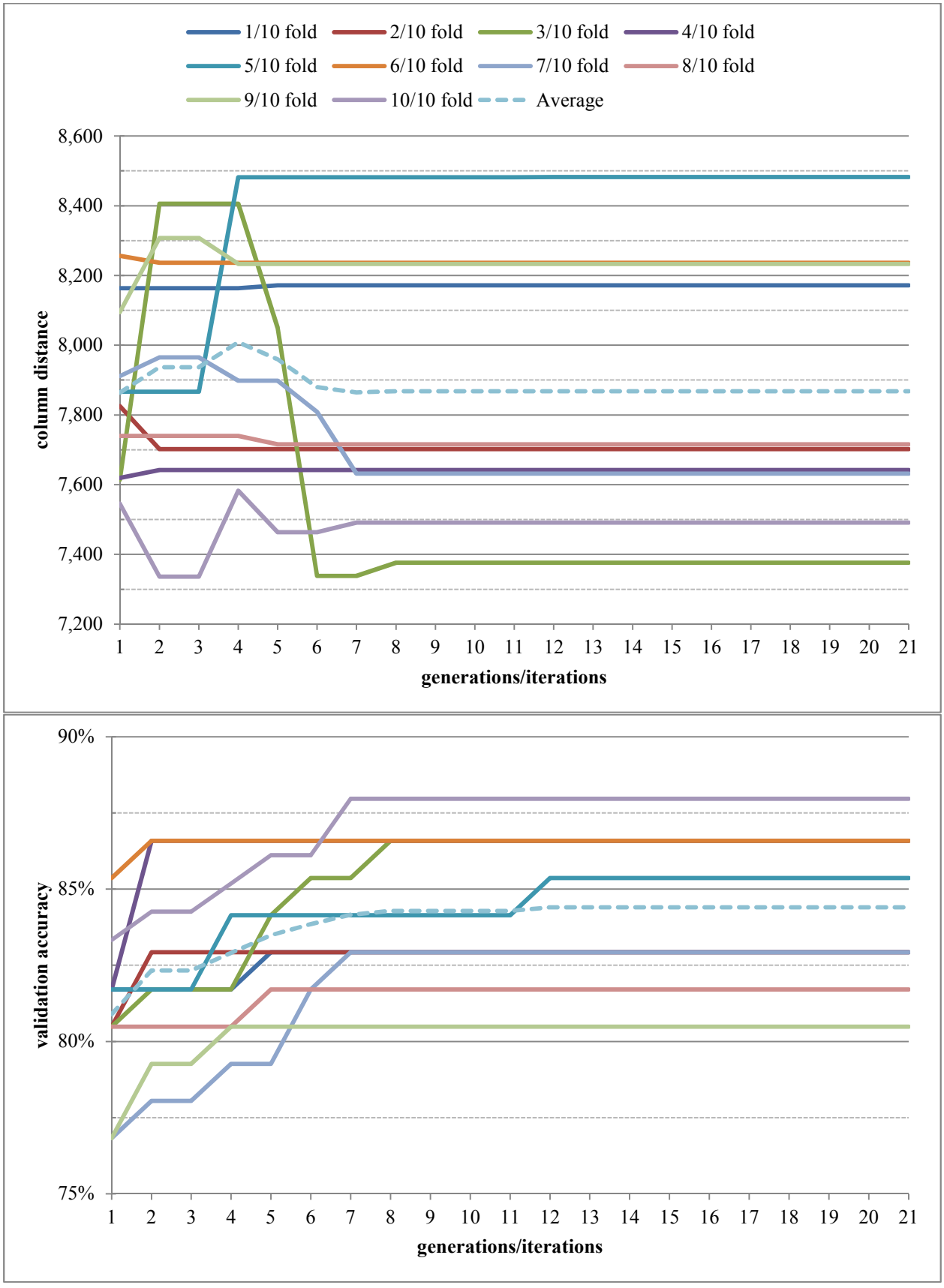

Figure 32: Accuracy as fitness function for vehicle dataset

4.3 Discussion

We first used Homogeneous ECOC (Chapter 2.2.3) on the thirteen datasets (Chapter 4.1) to serve as a benchmark. We used the same *c* classifiers in our classifier pool (Chapter 2.1) one by one for all *b* binary classification problems homogeneously. We used 10-fold cross validation for overfitting avoidance, statistical relevance and to calculate the mean for all iterations of each classifier. Of the *c* classifiers, the classifier with highest validation accuracy is picked as the Homogeneous ECOC composition for each dataset. No single classifier outperformed every other classifier and instead a variety of classifiers were preferred (See Figure 33 and Figure 41). Classifier averages are as follows: ANN 34.46%, Naïve Bayes 76.70%, Decision Trees 80.84%, SVM 75.26%, and Logistic Regression 77.90%. The average of the best performing Homogeneous ECOC classifier compositions is 83.89% (Figure 34 and Figure 44).

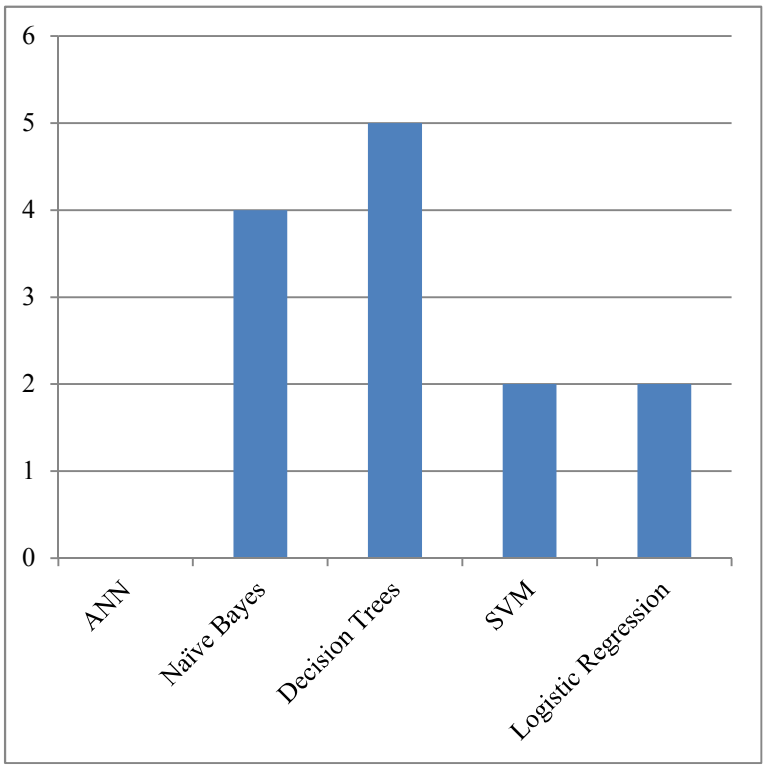

Figure 33: Homogeneous ECOC classifier preferences

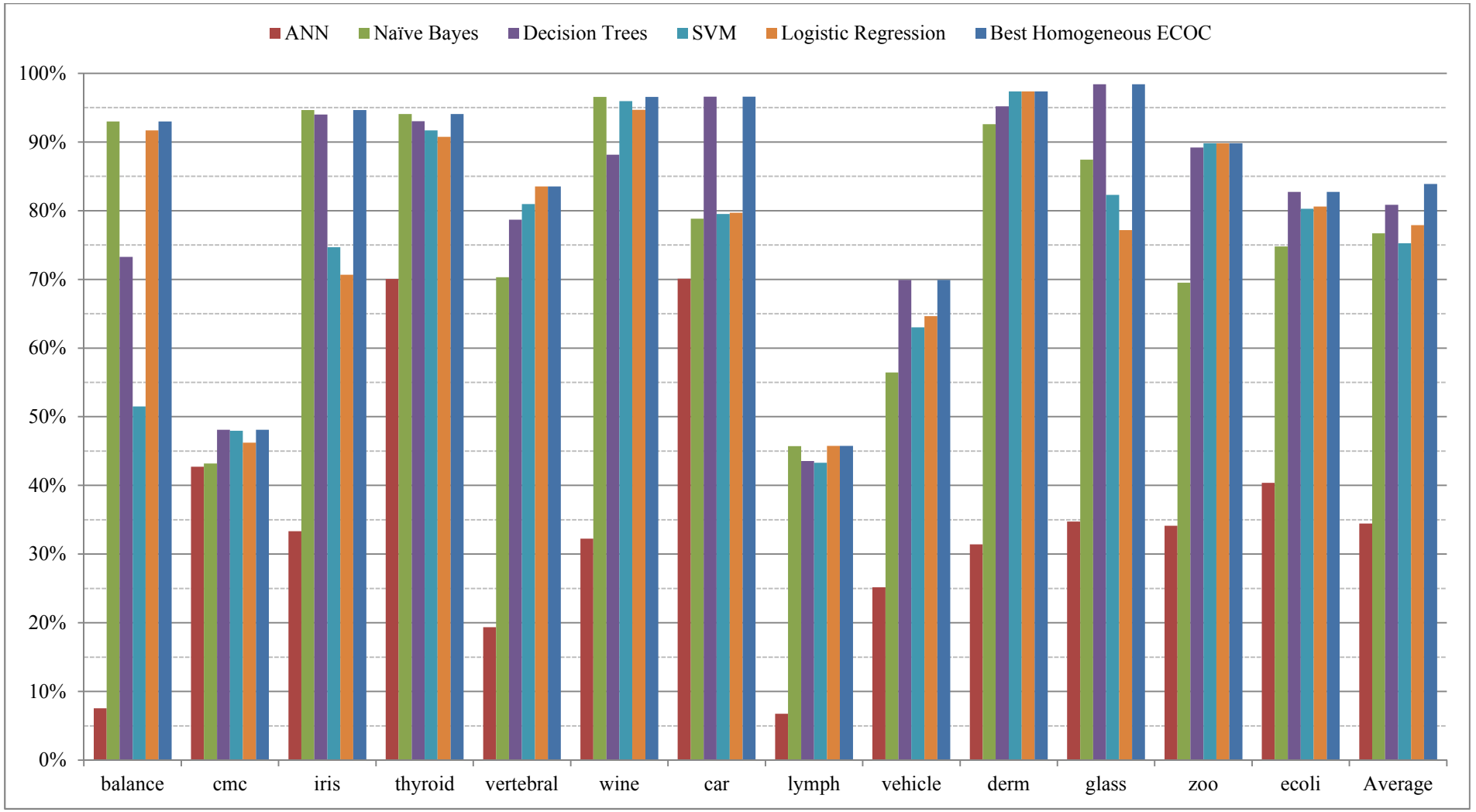

Figure 34: Homogeneous ECOC validation accuracies

Afterwards on the same thirteen datasets as before we applied the Heterogeneous ECOC approach (Chapter 3). With this approach the optimization problem is to find an ID vector of length *b* with each element having *c* possible values. We experimented with four different units of fitness to distinguish ID vectors: column distance, row distance, column × row distance, and validation accuracy (Chapter 4.2). We used 10-fold cross-validation to verify the statistical relevance of these experiments. We used an exhaustive search with column distance as our unit of fitness on nine datasets and calculated 73.21% as the validation accuracy however, the size of the search space made it impractical to use this approach on higher class problems (Chapter 4.2.1). We then used a genetic algorithm using all of the four units of fitness. The four runs revealed and interesting feature of our approach. We noticed that column distance and column × row distance based units of fitness predominantly preferred hording one classifier type rather than a more balanced distribution of classifiers as the other two unit of fitness (Figure 35 and Figure 42). The average validation accuracy based on each of the units of fitness is as follows: column distance 76.57%, row distance 83.18%, column × row distance 80.67%, and validation accuracy 86.91% (See Figure 36 and Figure 45).

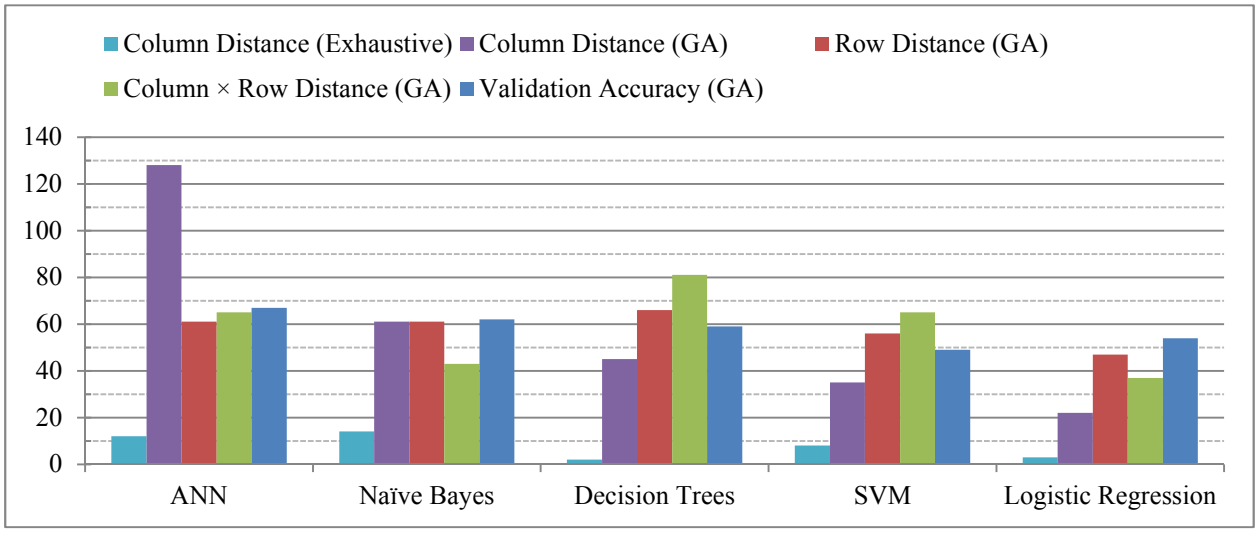

Figure 35: Heterogeneous ECOC classifier preferences

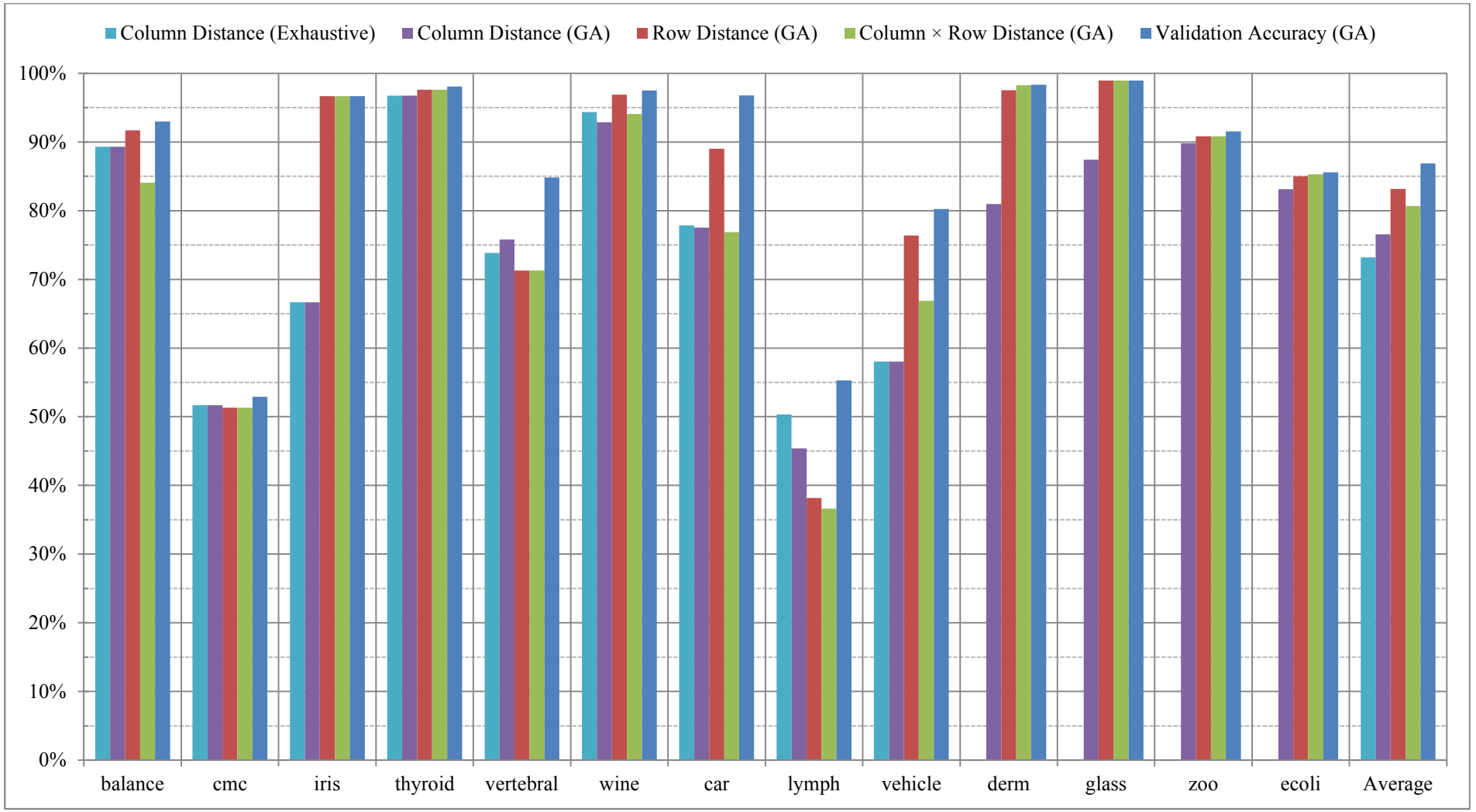

Figure 36: Heterogeneous ECOC validation accuracies

We compare the validation accuracies from Homogeneous ECOC serving as a benchmark with the Heterogeneous ECOC approach validation accuracies for all of the datasets. From the exhaustive search with column distance as the unit if fitness, we calculated an average loss of 7.04% for the first nine datasets from the equivalent Homogeneous ECOC validation accuracy values. From the genetic algorithm search with column distance as the unit if fitness, we calculated an average loss of 7.58% for the first nine datasets and average loss of 7.32% for all datasets from Homogeneous ECOC validation accuracy values, performing the worst. From the genetic algorithm search with column × row distance as the unit of fitness, we calculated an average loss of 3.22% from Homogeneous ECOC validation accuracy values, performing somewhere between the two methods that either use column distance or row distance as the unit of fitness. From the genetic algorithm search with row distance as the unit if fitness, we calculated an average loss of 0.71% from Homogeneous ECOC validation accuracy values, performing the best among distance based units of fitness. From the genetic algorithm with validation accuracy as unit of fitness, we calculated an average gain of 3.01% from Homogeneous ECOC validation accuracy values, performing the best among all units of fitness (Figure 37 and Figure 46).

Among the two search approaches using column distances as the unit of fitness for the first nine datasets, exhaustive search (73.21%) performed slightly better than genetic algorithm search (72.67%) because it was able to find better (higher) column distance because in some datasets it converged to a local optima, a known limitation of the genetic algorithm (Horn & Goldberg, 1994). This difference of 0.54% however was not significant enough for us to prefer exhaustive search to experiment on other units of fitness.

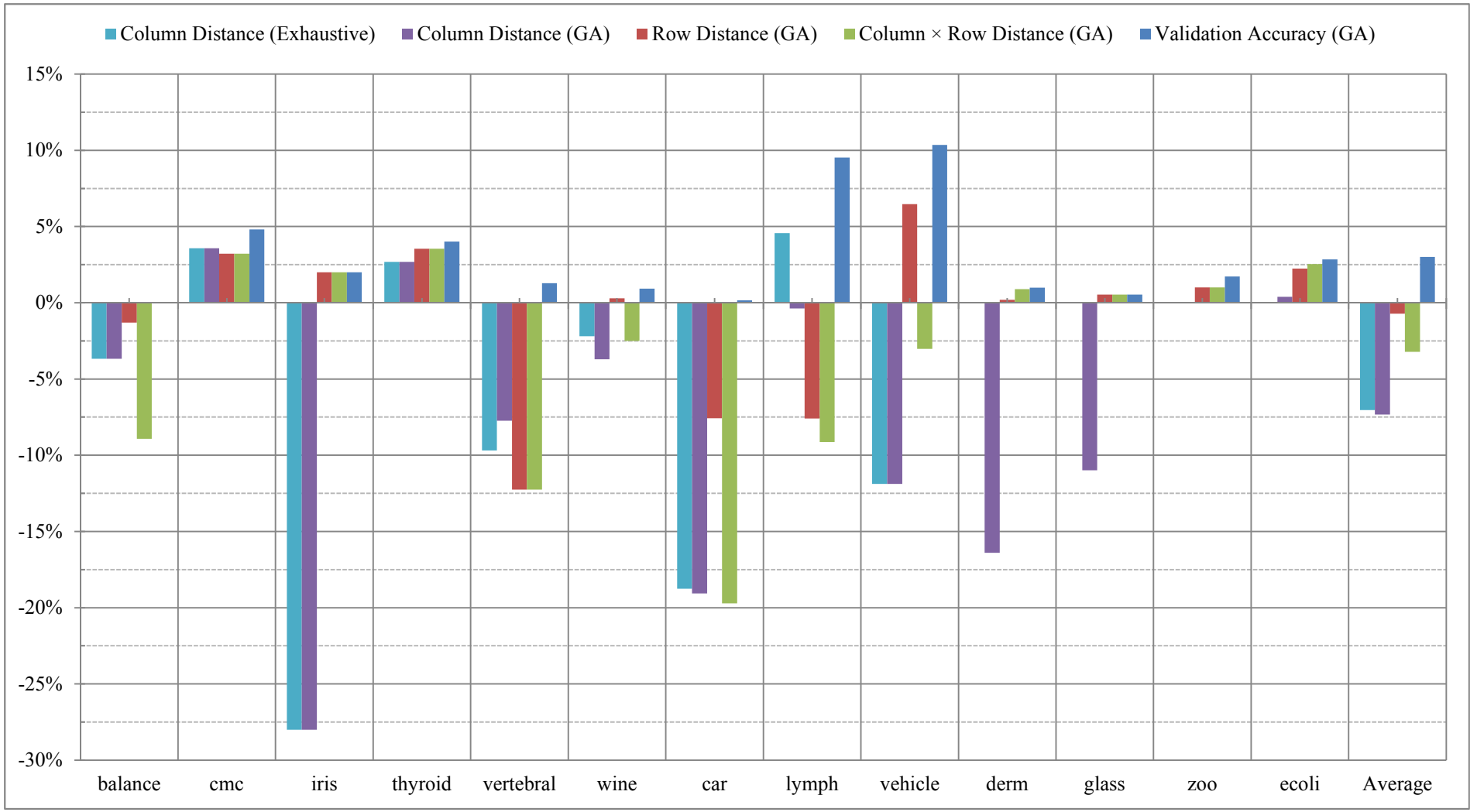

Figure 37: Validation accuracy gain or loss of Heterogeneous over Homogeneous ECOC

One of the most interesting results were with the balance dataset where we observed our approaches with distance as the unit of fitness with heterogeneously distributed ID vectors performing worse than validation accuracy based unit of fitness which had an unforeseen homogenous composition. If we review the balance dataset (Chapter 4.1.1), this actually is not a very extraordinary result as each feature of this 3-class artificial dataset is calculable using the other three features which makes each four feature of each 625 instance have parallel relationships with their class label. Column or row distance relationships would undermine the pre-existing relationships in this dataset and its simple nature would prevent error recovery. While this is a special case, it demonstrates a key weakness of the distance based unit of fitness. This also suggests that homogenous compositions can outperform heterogeneous compositions in such very specific cases. Hence units of fitness methods for Heterogeneous ECOC should not skip such homogeneous compositions during search for the optimal ID vector.

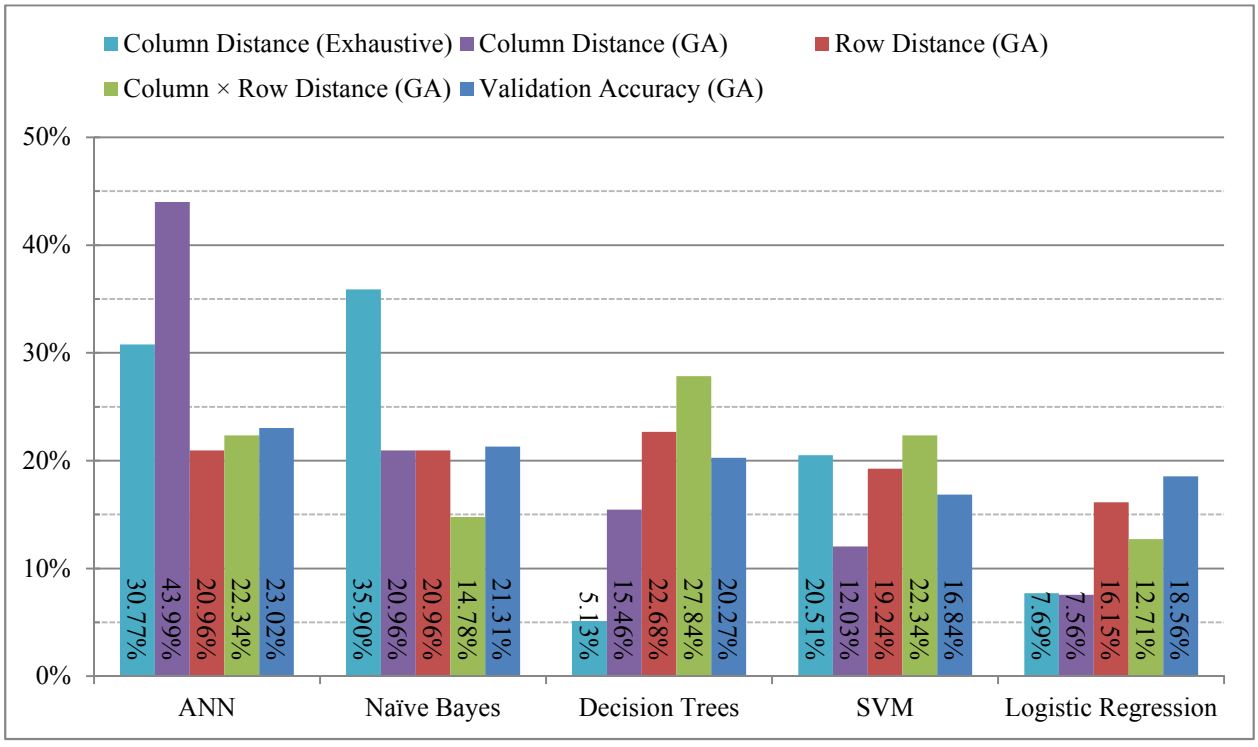

Figure 38: Heterogeneous ECOC classifier preference percentages

4.4 Conclusion

We have compared the performance of the four units of fitness of the Heterogeneous ECOC approach through the use of the genetic algorithm search. All three distance based units of fitness (column distance, row distance, column × row distance) provided inconsistent and unsatisfactory results in terms of validation accuracy. While our distance based units of fitness Heterogeneous ECOC approaches have performed on par or slightly better than the Homogeneous ECOC results on some datasets, they had underperformed on others. On the other hand Heterogeneous ECOC approach with validation accuracy as the unit of fitness performed either the same or better than Homogeneous ECOC's validation accuracies. We also observed that validation accuracy is not necessarily directly proportional to column and/or row distance as in some cases column and/or row distance values have actually decreased as validation accuracy has instead.

We discuss our observations and conclusions from these experiments in greater detail in the next chapter. We also discuss possible future research possibilities such as experimenting with Heterogeneous ECOC in different ECOC settings, improving our genetic algorithm so that it better avoids local optima convergence, experimenting on more domains, particularly larger ones with real-world applications.

5 Conclusion

Thus far we have discussed the beneficial properties of converting multi-class classification problems to multiple binary class classification problems where we have given great emphasis to the approach Error-Correcting Output Codes (ECOC).

In its exhaustive setting ECOC converts a k-class multi-class classification problem to $b = 2^{(k-1)} - 1$ binary classification problems and relies on a predetermined codeword matrix where each row refers to a class and each column refers to the binary classification problem. Ideally these columns and rows should have high column and row separation in terms of Hamming distance to have better error recovery during the decoding phase and also to avoid overfitting the training data. ECOC solves these binary classification problems using a binary classifier of the same type in a homogeneous manner (Chapter 2.2.3).

With this work we investigated the effects of relaxing this property of ECOC so that the binary classification problems can be solved in a heterogeneous manner. While this modification may appear intuitively sound, it does introduce a number of complications. For instance this new approach creates an optimization problem where the algorithm now needs to decide on a classifier composition to solve the individual binary classification problems for which the algorithm needs to have the ability to distinguish classifiers from each other. However, properties of ECOC such as the codeword matrix column and row separation Hamming distance are insufficient to make such a distinction. We discuss how we have addressed these complications with the *Heterogeneous ECOC* approach (Chapter 3) in the next section.

5.1 Heterogeneous ECOC

The Heterogeneous ECOC approach naturally treats the task as an optimization problem where it searches for a classifier composition of *b* binary classifiers combination from a pool of *c* classifiers that has the optimal performance to solve each of the *b* binary classification problems.

We expressed this task in the form of an ID vector of length *b* where each element corresponds to a classifier ID (See Figure 10) corresponding to a classifier that solves one of the binary classification problems. This notation allowed us to search for combinations of classifiers. In order to avoid overfitting the training data and in order to verify the statistical relevance of our results we used 10-fold cross validation (Chapter 3.3) on all of our search approaches. Each fold returned one or more ID vectors. We then run a second 10-fold cross validation to decide on which one of the ID vector has the optimal performance again in a statistically relevant manner.

Before testing our approach we gathered Homogenous ECOC values as a benchmark by testing each distribution from the same pool of *c* classifiers where for each dataset we only use the composition with the highest validation accuracy. We first approached our optimization problem with an exhaustive search using column distance as our unit of fitness (Chapter 3.2.2), however the impracticalities of this approach became apparent fairly quickly as the search space is double exponential. We then experimented with genetic algorithm search (Chapter 3.2.3) using the same unit of fitness to observed the performance of this local search algorithm. After having a comparable result with our initial exhaustive approach, we relied on genetic algorithm search to investigate the four different units of fitness (Chapter 3.1) for the fitness function.

5.2 Findings and Results

We use the performance of aforementioned Homogeneous ECOC in terms of validation accuracy at which we observed that no single homogenous classifier outperforms every other (See Figure 33 and Figure 41). We then compared these performances with the performance of the Heterogeneous ECOC approach. We tested our approach with four units of fitness that utilize knowledge from the training instances: column distance, row distance, column × row distance, and validation accuracy (Chapter 3.1).

While we have used genetic algorithm as our search algorithm, any type of search should facilitate for the same task. Our results show that Heterogeneous ECOC approaches that use distance based units of fitness underperforms even behind Homogeneous ECOC values. On the other hand our validation accuracy based Heterogeneous ECOC approach performed better than Homogeneous ECOC values (Figure 39). We investigated the reason behind this seemingly counter-intuitive finding.

ECOC's focus on maximizing the two properties column separation and row separation is intended to minimize correlated errors and to also minimize complements of columns which also cause correlation of errors. Homogeneous ECOC is designed in a manner to exploit this classification by consensus approach however; it can only recover from errors if and only if the errors themselves have relatively low correlation across codewords (Diettrich & Bakiri, 1995). With the Heterogeneous ECOC approach we attempt to reduce the possibility of correlation of errors as the binary classification problems are solved by diverse group of classifiers. It is important to note however that deciding on a robust unit of fitness is vital for our approach as a poorly selected classifier composition will make correlated errors.

Our findings strongly suggest that the relationship between validation accuracy and column and/or row distance is not straightforward (Kuncheva, 2005) as we observed fluctuations in validation accuracy even when the column and/or row distance consistently increases on some datasets. Likewise a consistent increase in validation accuracy can also result in fluctuations in column and/or row distance. Hence for the purpose of searching for an optimal classifier composition, we consider distance based units of fitness not adequately robust enough as a metric. Furthermore diversity of the Heterogeneous ECOC approach also allows the possibility of ID vector compositions that have very correlated errors which may also have column and/or row distances – even higher than that of the optimal classifier composition to the binary classification problems, particularly if one or more binary classifiers from the pool of classifiers have a very poor and correlated performance. Hence we conclude that the relationship between validation accuracy and column distance or with row distance is weakly correlated if not completely uncorrelated.

We would also like to note that the main intention of our study was to observe the effects of and complications from relaxing the homogeneous property of ECOC's classifier composition rather than to find the optimal classifier compositions for the individual datasets. Hence, in the interest of time constraints we did not run comprehensive searches that seek an optimal convergence. However there is no reason for validation accuracy to decrease through a longer search algorithm run with validation accuracy as the unit of fitness. We would in fact expect a longer search to provide more optimal solutions to the optimization problem. However, we cannot with certainty expect the same for distance based unit of fitness.

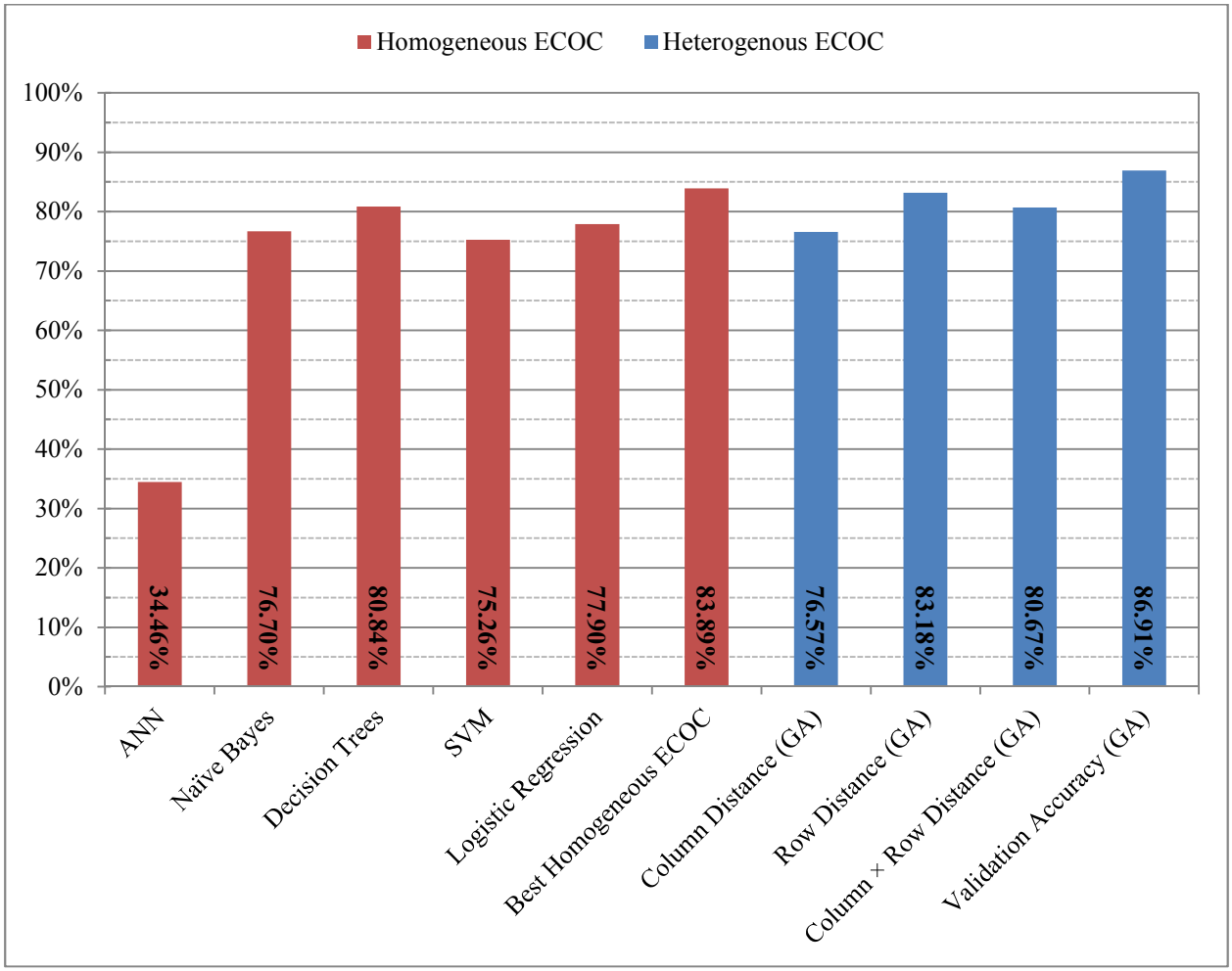

Figure 39: Validation accuracy averages of ECOC approaches

Another interesting feature we observed was the overall classifier preferences of the four approaches with different units of fitness. We observed a correlation between validation accuracy and even distribution in classifier preferences. Column distance unit of fitness predominantly preferred the classifier ANN while column × row distance unit of fitness predominantly preferred decision trees and both had a significant underperformance unlike row distance and validation accuracy units of fitness which had a more uniform distribution of classifier preferences and had far better performance (Figure 35 and Figure 42).

This was an unexpected result as we expected classifier combinations with higher column and/or row distances to have a more diverse composition rather than hoarding any

specific classifier. We suspect this is due to high correlation of errors by these classifiers which happen to also have high distance values. We were also surprised by the uniform distribution of the classifier preference of our validation accuracy based method as we intuitively expected this approach to predominantly prefer the classifier with the best performance (Figure 38 and Figure 43).

5.3 Future Research

For future research we have a number of proposals we will focus on in order to have more practical results but also to circumvent problems stemming from ECOC itself and from the genetic algorithm.

We want our approach to be usable on higher class classification problems, increasing the number of real-world applications. A serious setback for our approach currently is the double exponential growth of the search space (Chapter 3.2.2) because we rely on exhaustive ECOC as the foundation of our approach. We will investigate the performance of our approach with different ECOC approaches such as the ones proposed by (Mayoraz & Moreira, 1997), (Allwein, Schapire, & Singer, 2001), (Windeatt & Ghaderi, 2003), (Escalera, 2009), and (Smirnov, Moed, Nalbantov, & Sprinkhuizen-Kuyper, 2011).

Our current implementation of genetic algorithm search suffers from a number of problems that has an impact on the overall performance of the search. One such problem is local optima convergence, a known limitation of genetic algorithms (Horn & Goldberg, 1994), which we suspect we can further limit the impact of by implementing in mutation operations. We will also investigate other possible improvements to reduce the effects of this problem. We want to also have a better method to detect convergence such that the genetic

algorithm search would be terminated in the event of such a convergence allowing larger number of generations if they are needed.

Lastly, all the datasets we used in this work were fairly small with their number of instances ranging between 101 and 1,728 (Chapter 4.1). While this many instances are sufficient for the purpose of this research, it is not sufficient for real-world usage as the validation error is fairly high for most datasets (Chapter 4.3) which would greatly limit the real world applications of our approach in these domains. One of these domains we will investigate applications of our approach is classification of edits on Wikipedia.

Bibliography

Allwein, E. L., Schapire, R. E., & Singer, Y. (2001). Reducing multiclass to binary: A unifying approach for margin classifiers. *The Journal of Machine Learning Research, 1*, 113-141.

Berger, A. (1999). Error-Correcting Output Coding for Text Classification. *IJCAI-99: Workshop on machine learning for information filtering*.

Bishop, C. M. (2006). *Pattern recognition and machine learning.* New York: Springer.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning, 20*, 273-297.

Cory, D. G., Fahmy, A. F., & Havel, T. F. (1997). Ensemble quantum computing by NMR spectroscopy. *Proceedings of the National Academy of Sciences, 94*, 1634-1639.

Diettrich, T. G., & Bakiri, G. (1995). Solving Multiclass Learning Problems via Error-Correcting Output Codes. *Journal of Artificial Intelligence Research 2*, 263-286.

Drucker, H., Burges, C. J., Kaufman, L., Smola, A., & Vapnik, V. (1997). Support vector regression machines. *Advances in neural information processing systems, 9*, 155-161.

Escalera, S. (2009). Coding and Decoding Design of ECOCs for Multi-class Pattern and Object Recognition. *Doctoral dissertation, UAB*.

Finley, T., & Joachims, T. (2008). Training structural SVMs when exact inference is intractable. *Proceedings of the 25th international conference on Machine learning*, 304-311.

Flach, P. (2012). *Machine Learning: The Art and Science of Algorithms that Make Sense of Data.* Cambridge University Press.

Fürnkranz, J. (2002). Pairwise classification as an ensemble technique. *Machine Learning: ECML*, 97-110.

Gammerman, A., Vovk, V., & Vapnik, V. (1998). Learning by transduction. *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, 148-155.

Hedeker, D. (2003). A mixed-effects multinomial logistic regression model. *Statistics in medicine, 22*, 1433-1446.

Hilbe, J. M. (2009). *Logistic regression models.* Boca Raton: CRC Press.

Horn, J., & Goldberg, D. E. (1994). Genetic Algorithm Difficulty and the Modality of Fitness Landscapes. *Foundations of Genetic Algorithms 3*.

Kong, E. B., & Dietterich, T. G. (1995). Error-Correcting Output Coding Corrects Bias and Variance. *ICML*, 313-321.

Kowalczyk, A., & Chapelle, O. (2005). An analysis of the anti-learning phenomenon for the class symmetric polyhedron. *Algorithmic Learning Theory*.

Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling.* New York: Springer.

Kuncheva, L. I. (2005). Using diversity measures for generating error-correcting output codes in classifier ensembles. *Pattern Recognition Letters, 26*, 83-90.

Larochelle, H., Bengio, Y., Louradour, J., & Lamblin, P. (2009). Exploring strategies for training deep neural networks. *The Journal of Machine Learning Research, 10*, 1-40.

Mayoraz, E., & Moreira, M. (1997). On the decomposition of polychotomies into dichotomies. *ICML (Vol. 97)*, 219-226.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics, 5*, 115-133.

Melvin, I., Ie, E., Weston, J., Noble, W. S., & Leslie, C. S. (2007). Multi-class Protein Classification Using Adaptive Codes. *Journal of Machine Learning Research, 8*, 1557-1581.

Natarajan, B. K. (1991). *Machine Learning: A Theoretical Approach.* San Mateo, CA: Morgan Kaufmann.

Ng, A. Y., & Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems, 2*, 841-848.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 81-106.

Quinlan, J. R. (1993). *C4. 5: programs for machine learning.* Morgan Kaufmann.

Revelt, D., & Train, K. (1998). Mixed logit with repeated choices: households' choices of appliance efficiency level. *Review of economics and statistics, 80*, 647-657.

Rifkin, R., & Klautau, A. (2004). In defense of one-vs-all classification. *The Journal of Machine Learning Research, 5*, 101-141.

Rish, I. (2001). An empirical study of the naive Bayes classifier. *IBM Research Report*.

Rokach, L. (2010). *Pattern classification using ensemble methods.* World Scientific Publishing.

Rosenblatt, F. (1957). The perceptron, a perceiving and recognizing automaton Project Para. *Cornell Aeronautical Laboratory*.

Rumelhart, D. E., & MacClelland, J. L. (1988). *Parallel distributed processing.* IEEE.

Smirnov, E. N., Moed, M., Nalbantov, G., & Sprinkhuizen-Kuyper, I. (2011). Minimally-Sized Balanced Decomposition Schemes for Multi-class Classification. *Ensembles in Machine Learning Application*, 39-58.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM, 27*, 1134-1142.

Vapnik, V. N. (1998). Statistical learning theory.

Vapnik, V., & Lerner, A. (1963). Pattern recognition using generalized portrait method. *Automation and Remote Control, 24*, 774–780.

Weston, J., & Watkins, C. (1998). Multi-class support vector machines. *Technical Report CSD-TR-98-04*.

Williams, R. J., & Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural computation, 1*, 270-280.

Windeatt, T., & Ghaderi, R. (2003). Coding and decoding strategies for multi-class learning problems. *Information Fusion, 4*, 11-21.

Appendix A – Sources of All Datasets

Name	Sources of Datasets
balance	• Generated to model psychological experiments reported by Siegler, R. S. (1976) • Three Aspects of Cognitive Development. Cognitive Psychology, 8, 481-520
car	• Marko Bohanec, Jožef Stefan Institute, Ljubljana, Slovenia
cmc	• Subset of the 1987 National Indonesia Contraceptive Prevalence Survey by Tjen-Sien Lim
derm	• Nilsel Ilter, M.D., Ph.D., Gazi University, School of Medicine, Ankara, Turkey • H. Altay Guvenir, PhD., Bilkent University, Department of Computer Engineering and Information Science, Ankara, Turkey
ecoli	• Kenta Nakai, Institue of Molecular and Cellular Biology, Osaka University, Osaka, Japan
glass	• B. German, Central Research Establishment, Home Office Forensic Science Service, Aldermaston, United Kingdom
iris	• Database collected by Edgar Anderson and introduced by Sir Ronald Fisher
lymph	• This lymphography domain was obtained from the University Medical Centre, Institute of Oncology, Yugoslavia. Thanks go to M. Zwitter and M. Soklic for providing the data.
thyroid	• Danny Coomans, Dept. of Maths. and Stats., James Cook University, Townsville, Australia
vehicle	• Drs.Pete Mowforth and Barry Shepherd, Turing Institute, Glasgow, United Kingdom
vertebral	• Guilherme de Alencar Barreto & Ajalmar Rêgo da Rocha Neto, Department of Teleinformatics Engineering, Federal University of Ceará, Ceará, Brazil. • Henrique Antonio Fonseca da Mota Filho, Hospital Monte Klinikum, Fortaleza, Ceará, Brazil.
wine	• Forina, M. et al, PARVUS - An Extendible Package for Data Exploration, Classification and Correlation, Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno, Genoa, Italy.
zoo	• Richard S. Forsyth, School of Psychology, University of Nottingham, United Kingdom

Figure 40: Attribution of the sources of the datasets used

Appendix B – Classifier Preferences of ECOC Approaches

Classifiers	balance	cmc	iris	thyroid	vertebral	wine	car	lymph	vehicle	derm	glass	zoo	ecoli	Total
Artificial Neural Networks														0
Naïve Bayes	×		×	×		×								4
Decision Trees		×					×		×		×		×	5
Support Vector Machines										×		×		2
Logistic Regression					×			×						2

Figure 41: Classifier preference of Homogeneous ECOC

Classifier Preference of Units of Fitness	ANN	Naïve Bayes	Decision Trees	SVM	Logistic Regression	All
Validation Accuracy (GA)	67	62	61	51	50	291
Row Distance (GA)	61	61	66	56	47	291
Column × Row Distance (GA)	65	43	81	65	37	291
Column Distance (GA)	128	61	45	35	22	291
Column Distance (Exhaustive)	12	14	2	8	3	39
Total	333	241	253	213	163	1203

Figure 42: Classifier preferences of Heterogeneous ECOC

Classifier Preference of Units of Fitness	ANN	Naïve Bayes	Decision Trees	SVM	Logistic Regression
Validation Accuracy (GA)	23.02%	21.31%	20.27%	16.84%	18.56%
Row Distance (GA)	20.96%	20.96%	22.68%	19.24%	16.15%
Column × Row Distance (GA)	22.34%	14.78%	27.84%	22.34%	12.71%
Column Distance (GA)	43.99%	20.96%	15.46%	12.03%	07.56%
Column Distance (Exhaustive)	30.77%	35.90%	05.13%	20.51%	07.69%
Total	27.68%	20.03%	21.03%	17.71%	13.55%

Figure 43: Classifier preference percentages of heterogeneous ECOC

Appendix C – Summary of Results of All Datasets

Classifier	balance	cmc	iris	thyroid	vertebral	wine	car	lymph	vehicle	derm	glass	zoo	ecoli	Average
Artificial Neural Networks	7.53%	42.72%	33.33%	70.05%	19.35%	32.24%	70.09%	6.76%	25.18%	31.43%	34.75%	34.12%	40.39%	34.46%
Naïve Bayes	92.98%	43.19%	94.67%	94.08%	70.32%	96.58%	78.84%	45.71%	56.44%	92.58%	87.43%	69.52%	74.80%	76.70%
Decision Trees	73.29%	48.10%	94.00%	93.04%	78.71%	88.16%	96.61%	43.57%	69.91%	95.20%	98.42%	89.21%	82.75%	80.84%
Support Vector Machines	51.50%	47.95%	74.67%	91.70%	80.97%	95.96%	79.52%	43.29%	63.01%	97.36%	82.31%	89.83%	80.29%	75.26%
Logistic Regression	91.68%	46.21%	70.67%	90.75%	83.55%	94.71%	79.72%	45.76%	64.66%	97.36%	77.19%	89.83%	80.59%	77.90%
Best Homogenous ECOC	92.98%	48.10%	94.67%	94.08%	83.55%	96.58%	96.61%	45.76%	69.91%	97.36%	98.42%	89.83%	82.75%	83.89%

Figure 44: Validation accuracies of Homogeneous ECOC

Unit of Fitness	balance	cmc	iris	thyroid	Vertebral	wine	car	lymph	vehicle	derm	glass	zoo	ecoli	Average
Column Distance (Exhaustive)	89.30%	51.68%	66.67%	96.76%	73.87%	94.38%	77.85%	50.33%	58.03%	-	-	-	-	73.21%[3]
Column Distance (GA)	89.30%	51.68%	66.67%	96.76%	75.81%	92.87%	77.55%	45.38%	58.03%	80.96%	87.43%	89.83%	83.14%	76.57%
Row Distance (GA)	91.68%	51.32%	96.67%	97.62%	71.29%	96.88%	89.03%	38.17%	76.38%	97.56%	98.95%	90.83%	85.00%	83.18%
Column × Row Distance (GA)	84.06%	51.32%	96.67%	97.62%	71.29%	94.08%	76.90%	36.62%	66.89%	98.25%	98.95%	90.83%	85.29%	80.67%
Validation Accuracy (GA)	92.98%	52.91%	96.97%	98.10%	84.84%	97.50%	96.78%	55.29%	80.26%	98.35%	98.95%	91.55%	85.59%	86.91%

Figure 45: Validation accuracies of Heterogeneous ECOC

Unit of Fitness	balance	cmc	iris	thyroid	vertebral	wine	car	lymph	vehicle	derm	glass	zoo	ecoli	Average
Column Distance (Exhaustive)	-3.68%	3.58%	-28.00%	2.68%	-9.68%	-2.20%	-18.76%	4.57%	-11.88%	-	-	-	-	-7.04%[3]
Column Distance (GA)	-3.68%	3.58%	-28.00%	2.68%	-7.74%	-3.71%	-19.06%	-0.38%	-11.88%	-16.40%	-10.99%	0.00%	0.39%	-7.32%
Row Distance (GA)	-1.30%	3.22%	2.00%	3.54%	-12.26%	0.30%	-7.58%	-7.59%	6.47%	0.20%	0.53%	1.00%	2.25%	-0.71%
Column × Row Distance (GA)	-8.92%	3.22%	2.00%	3.54%	-12.26%	-2.50%	-19.71%	-9.14%	-3.02%	0.89%	0.53%	1.00%	2.54%	-3.22%
Validation Accuracy (GA)	0.00%	4.81%	2.00%	4.02%	1.29%	0.92%	0.17%	9.53%	10.35%	0.99%	0.53%	1.72%	2.84%	3.01%

Figure 46: Validation accuracy gain or loss of Heterogeneous over Homogeneous ECOC

[3] Only the average of the first nine datasets.