Spring 2016

Applications of Computational Geometry and Computer Vision

Joseph Lemley
Central Washington University, lemleyj@cwu.edu

APPLICATIONS OF COMPUTATIONAL GEOMETRY AND COMPUTER VISION

A Thesis

Presented to

The Graduate Faculty

Central Washington University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Computational Science

---

by

Joseph Ely Lemley

June 2016

CENTRAL WASHINGTON UNIVERSITY

Graduate Studies

We hereby approve the thesis of

Joseph Ely Lemley

Candidate for the degree of Master of Science

APPROVED FOR THE GRADUATE FACULTY

_____ _____
Dr. Razvan Andonie, Committee Chair

_____ _____
Dr. Donald Davendra

_____ _____
Dr. Boris Kovalerchuk

_____ _____
Dean of Graduate Studies

ABSTRACT

APPLICATIONS OF COMPUTATIONAL
GEOMETRY AND COMPUTER VISION

by

Joseph Ely Lemley

June 2016

Recent advances in machine learning research promise to bring us closer to the original goals of artificial intelligence. Spurred by recent innovations in low-cost, specialized hardware and incremental refinements in machine learning algorithms, machine learning is revolutionizing entire industries. Perhaps the biggest beneficiary of this progress has been the field of computer vision. Within the domains of computational geometry and computer vision are two problems: Finding large, interesting holes in high dimensional data, and locating and automatically classifying facial features from images.

State of the art methods for facial feature classification are compared and new methods for finding empty hyper-rectangles are introduced. The problem of finding holes is then linked to the problem of extracting features from images and deep learning methods such as convolutional neural networks. The performance of the hole-finding algorithm is measured using multiple standard machine learning benchmarks as well as a 39 dimensional dataset, thus demonstrating the utility of the method for a wide range of data.

ACKNOWLEDGEMENTS

I am grateful to the college of the sciences for providing a $400 grant for expenses related to travel to COMPSAC 2016, and to the Computer Science department for providing $500 for additional conference-related expenses.

My research would not have been possible without the Titan X GPU and high performance computers provided by the Computer Science Department.

I also wish to thank Dr. Filip Jagodizinski for providing me with his HIV-1 dataset, computational geometry and structural biology knowledge and the many helpful suggestions he made in improving my paper, which later became a substantial portion of this thesis.

I thank Alexandru Drimbarean, Gabriel Costache, and Pawel Filipczuk of FotoNation, Ireland who initially suggested Gender classification from face images as a thesis topic and taking the time for biweekly Skype meetings to discuss research questions.

Dr. Donald Davendra and Dr. Szilard Vajda who provided feedback and advice.

Dr. Boris Kovalerchuk, who first hired me in his imaging lab years ago when I was an undergraduate and who later provided me with advice about representing information in high dimensional space and served on my graduate committee.

Dr. Razvan Andonie, for providing advice and endless ideas and for first introducing me to my favorite topic: Computational Intelligence 10 years ago and for suggesting I come back to Central Washington University for my master's degree.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Within the domains of computational geometry and computer vision are two problems: Finding large, interesting holes in high dimensional data, and locating and automatically classifying facial features from images.

Recent advances in machine learning research promise to bring us closer to the original goals of artificial intelligence. Spurred by recent innovations in low cost, specialized hardware and incremental refinements in machine learning algorithms, machine learning is revolutionizing entire industries. Perhaps the biggest beneficiary of this progress has been the field of computer vision. Finding large empty rectangles or boxes in a set of points in 2D and 3D space has been well studied. Efficient algorithms exist to identify the empty regions in these low dimensional spaces. First, I duplicated existing work on this problem, and then evaluated its performance on 1D, 2D, 3D, 4D, and 5D datasets. I also developed an application to visualize such holes in 2D and 3D space. I then designed a new algorithm for finding holes in high dimensional data that runs in polynomial time with respect to the number of dimensions and size of the input. This is significant because all previously published algorithms are exponential in the number of dimensions. Applications for algorithms that find large empty spaces include big data analysis, recommender systems, automated knowledge discovery, and query optimization. This new Monte Carlo algorithm discovers interesting maximal empty hyper-rectangles in cases where dimensionality and input size would otherwise make analysis impractical.

Automatic facial feature classification is an active and rapidly advancing area of research within computer vision. The problem of how to best classify facial features is an

open problem of great interest to industry, government, and academia. The best-known methods for this task involve the use of Convolutional Neural Networks (CNN) and are performed using graphical processing units (GPUs). I will compare my method with my own implementations of facial attribute classifiers using Eigenfaces and Support Vector Machines (SVM) as well as classifiers based on CNN that operate directly on pixels without filtering. I also investigate the use of dual-tree complex wavelet transform for enhanced feature learning and perform a comparative study of existing algorithms for gender classification.

Finally, I link the problem of finding holes to the problem of extracting features from images.

Much of the content in the following chapters has been published in [1] and [2].

Holes

I present the first algorithm for finding holes in high dimensional data that runs in polynomial time with respect to the number of dimensions. Previous published algorithms are exponential. Finding large empty rectangles or boxes in a set of points in 2D and 3D space has been well studied. Efficient algorithms exist to identify the empty regions in these low-dimensional spaces. Unfortunately, such efficiency is lacking in higher dimensions where the problem has been shown to be NP-complete when the dimensions are included in the input. Applications for algorithms that find large empty spaces include big data analytics, recommender systems, automated knowledge discovery, and query optimization. This Monte Carlo-based algorithm discovers interesting maximal empty hyper-rectangles in cases where dimensionality and input size would otherwise make analysis impractical. The run-time is polynomial in the size of the input and the number

of dimensions. I apply the algorithm on a 39-dimensional data set for protein structures and discover interesting properties that could not be inferred otherwise.

An important aspect of data mining research is discovering large empty areas (or holes) in data sets. According to Liu *et al*, "If we view each item (or tuple) in a data set as a point in a *k*-dimensional space, then a hole is a region in the space that contains no data points. In a continuous space, there exist a large number of holes because it is not possible to fill up the continuous space with data points."[3]

Liu goes on to state that such holes exist for the following reasons:

1. The data cases collected are insufficient, resulting in some regions having no data point.

2. Certain value combinations are not possible. For example, assume there is a data set with two continuous attributes X and Y. Both X and Y can take values from 1 to 10, but it is observed that `if X > 5, then Y < 4`. In other words, there exists an empty area, i.e., a rectangular region defined by `5 < X ≤ 10 and 4 ≤ Y ≤ 10`.

Organizations often deploy systems to generate rules about values that are present in their data. However, these rules often do not provide the user with complete information about their data set. For example, a particular organization may rely on a learning system to generate a set of rules from their database. One of the rules may be the following:

```
if Company Size > 600 then Service = Yes
```

Liu continues to point out, "This rule specifies that if the company has more than 600 employees, then the company uses the service provided by the organization. Now assume the company size is partitioned into 10 categories. A close inspection of the

database may reveal that no company, whose size is in the range of 400-800, uses the service. Hence, there is a hole in the data. Realizing the existence of this hole may lead the organization to probe into the possibilities of modifying its service or of doing more promotion to attract the medium size companies." [3]

In the case of a one-dimensional data set, finding the largest hole becomes the problem of finding large gaps and is trivial to solve. When the number of dimensions increases, the problem quickly becomes complex and has been shown to be NP-hard when the number of dimensions is included in the input. In two or more dimensions, locating large holes is very difficult. Consider for example Table 1, which lists 4 entries in a hypothetical 7-dimensional data set. Locating the largest hole in the seven-dimensional space is very difficult.

TABLE 1: Sample 7-Dimensional Data.

ID	**Age**	**GPA**	**Gender**	**Height**	**Weight**	**Income**
1	20	3.6	male	60in.	170lb	$100,000
2	19	4.0	female	75in.	160lb	$20,000
3	21	3.7	female	71in.	250lb	$94,000
4	26	3.4	female	62in.	150lb	$112,000

Notes: Finding the largest 7-dimensional hole is not intuitive.

In general, a data set contains a large number of holes because each item is only a point in a k-dimensional space. Even if each attribute takes discrete or nominal values, it may be still quite difficult to fill the entire space. However not all holes are interesting. Actually, most of them are not. I concur with Liu *et al.* [3] regarding the types of holes that tend to be interesting and uninteresting.

Liu *et al.* notes that the following types of holes are not interesting:

1. Small holes: they exist because there are only a limited number of cases in the data set.

2. Known impossible value combinations: they exist because certain value combinations are not possible and this fact is known previously.

Liu *et al.* also states, Certain types of holes can be of great importance:

1. Holes that represent impossible value combinations that are not known previously.
2. Holes that indicate that the data collected within those areas are insufficient.
3. Holes that are suspected by the user and need confirmation.[3]

Although holes can take the form of any shape, I focus particularly on the problem of axis-aligned rectangular holes in arbitrarily high dimensions. Holes of this type are easier to understand [3]: (especially in high dimensions) and have the useful property that they can be easily converted to if/then rules. Formally, a Maximal Empty Hyper-Rectangle (MEHR) is defined as follows:

> **Given a *k*-dimensional space S containing *n* points, where each dimension is bounded by a minimum and maximum value, a MEHR is a hyper-rectangle that has no data point within its interior and has at least one data point on each of its 2*k* bounding surfaces.**

Liu *et al.* calls these points the bounding points of the hyper-rectangle. Each side *i* of the MEHR is parallel to one axis of **S** and orthogonal to all the others. (see Figure 1).

The largest rectangles can often be assumed to be the most interesting because they have the largest impact on the distribution of points, thus the focus of this thesis is on finding such holes.

These are all application dependent. A simple way of measuring the size of a MEHR is by its volume. I follow the convention used by [4], [5], and [6] of a generalized definition of volume (or hyper-volume) when speaking of the size of rectangular objects

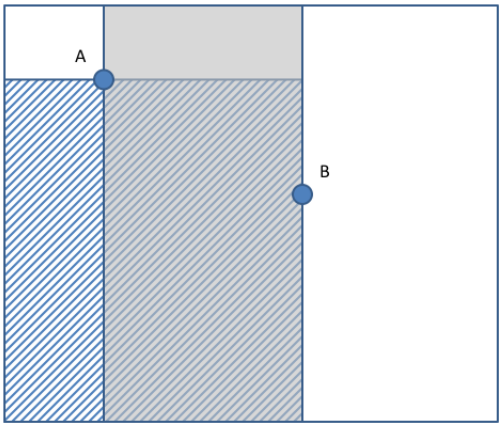


FIGURE 1: Two sample points, A and B, in 2D space. Both of these points are bounding points for the blue thatched and gray rectangles, that overlap. The top and bottom edges of the gray rectangle are bound by the minimum and maximum values of the *y*-direction, while the left and bottom edges of the blue thatched rectangle are bounded by the minimum *y* value and minimum *x* values of the 2D space.

in arbitrary dimensions. For axis aligned hyper-rectangles, this is the product of the distance between the smallest and largest points on each axis which are the boundary points of this shape.

The objective is to find the largest MEHRs in a *k*-dimensional space and rank them according to volume.

The upper bound on the number of MEHRs has been found to be in $O(n^{k-1})$ [3], where n is the number of k-dimensional data points. Finding all MEHRs has been shown to be NP-hard [7]. Previously developed approaches are unsuitable for large data sets primarily because they must locate and rank all MEHRs to find the largest MEHR.

This thesis presents the first efficient polynomial time algorithm for discovering large interesting hyper-rectangles in a *k*-dimensional space. The algorithm runs in polynomial time with respect to *k*. All previously published algorithms designed to solve this problem optimally are exponential. The solution is a Monte Carlo approach, which does not necessarily find the optimal solution. In experiments, this algorithm identifies the

same large holes as existing algorithms for low dimensions (where all existing algorithms perform well) but is also able to quickly identify large holes in high dimensional spaces that would not be feasible with existing approaches that exhaustively identify all of the rectangles and then find the largest from among them. The algorithm is well suited for large data sets with many attributes.

I demonstrate the speed and accuracy of this approach by comparing the time the method takes to find the largest hole with the time taken by the method used in [3].

After experimentally demonstrating that this algorithm produces correct results orders of magnitude faster than the comparison method on publicly available machine learning datasets, the algorithm is then used on a larger 39-dimensional data set. I will then show how to use information about large empty regions to infer interesting relationships among dimensions that would have been difficult to discover using other techniques.

Machine Learning for Face Feature Classification

Recently, several machine learning methods for gender classification from frontal facial images have been proposed. In addition to the diversity of methods, there is also a diversity of benchmarks used to assess them. This gave me the motivation: to select and compare in a concise but reliable way the main state-of-the-art methods used in automatic gender recognition. As expected, there is no overall winner. The winner, based on the accuracy of the classification, depends on the type of benchmarks used.

A major goal of computer vision and artificial intelligence is to build computers that can understand or classify concepts such as gender in the same way humans do.

Automatic classification of gender from frontal face images taken under contrived conditions has been well studied with impressive results. The variety of methods

published in the literature shows that there is not a unique or generic solution to the gender classification problem.

Applications of gender classification include, image search, automatic annotation of images, security systems, face recognition, and real time image acquisition on smart phones and mobile devices.

State-of-the-art gender classification methods generally fall into the following main categories: Convolutional Neural Networks (CNN), Dual Tree Complex Wavelet Transform (DTCWT) + a Support Vector Machine (SVM) classifier, and feature extraction techniques such as Principal Component Analysis (PCA), Histograms of Oriented Gradients (HOG) and others with a classifier (SVM, KNN, etc). The SVM approach is natural, since this is a two class problem. The CNN is related to the well-known deep learning paradigm. Kingsbury et al, notes that "DTCWT provides approximate shift invariance and directionally selective filters (properties lacking in the traditional wavelet transform) while preserving the usual properties of perfect reconstruction and computational efficiency with well-balanced frequency responses" [8].

To assess gender classification techniques, two types of benchmarks may be used: standard posed datasets (with well defined backgrounds, lighting and photographic characteristics) and datasets containing "in the wild" images that exhibit the diversity of subjects, settings, and qualities typical of everyday scenes.

State-of-the-art methods used in automatic gender recognition are compared on two benchmarks: the most popular standard dataset Facial Recognition Technology (FERET) [9] and a more challenging data set of "in the wild" images (Adience) [10].

Disseminated Results

Portions of this research have been disseminated at a number of venues, including peer reviewed international conferences as full papers.

"Big Holes in Big Data: A Monte Carlo Algorithm for Detecting Large Hyper-rectangles in High Dimensional Data." [2], was accepted for inclusion as a full paper for "The 40th IEEE Computer Society International Conference on Computers, Software & Applications" as part of their "Web, Big Data & Analytics (WEDA) series". Less than 18% of submitted papers were accepted as full papers in this international conference.

"Comparison of Recent Machine Learning Techniques for Gender Recognition from Facial Images" [1] was accepted as a full paper at the 27th Modern Artificial Intelligence and Cognitive Science Conference. MAICS is an international conference that exists to provide a forum for a wide range of research ideas in artificial intelligence and the cognitive sciences.

A presentation of this research on maximal empty hyper-rectangles was adapted for a general audience and was presented at SOURCE 2015 as an oral presentation. At SOURCE 2016, I presented an adapted version of this research on using deep learning for gender classification in an oral presentation.

CHAPTER II

LITERATURE REVIEW

In this chapter, I discuss prior literature on finding holes in data, I then introduce popular methods to classify gender from face images. In future chapters, I will show how to use holes to mine information from machine learning algorithms that are commonly used to classify gender.

A Review of Methods for Finding the Maximum Empty Rectangle

The problem of finding the maximum empty rectangle in a set of points has been studied in depth in low dimensions and has been proven intractable in high dimensions [11], [7].

In one dimension, the problem of finding the largest maximum space simplifies to the Maximum Gap problem [12] for which there exists a linear time solution based on the pigeonhole and bucketing principles. Preparata *et al.* suggest that no generalization of this approach seems possible in higher dimensions [12].

In two dimensions, the problem of finding the largest hole becomes the "largest rectangle" problem. One of the fastest solutions was proposed in [13] and runs in $O(n\,log^2\,n)$.

An efficient algorithm for finding the largest empty hyper-rectangle in 3D space also exists, developed by Datta, *et al.* [14]. Their algorithm runs in $O(n^3)$ and they report an average case complexity of $O(n\,log^4\,n)$.

The first algorithm to identify the largest maximal empty hyper-rectangle in an arbitrary number of dimensions was introduced in [3] and [15]. The algorithm runs in $O(n^{2k-1}\,k^3\,(log\,n)^2)$ time and $O(n^{2k-1})$ space. It relies on a heuristic *BigEnough()*

which allows small rectangles to not be considered. This heuristic substantially narrows the search space and improves execution time. In the context of large data sets, a main disadvantage of this algorithm is that it requires calculating and storing every empty hyper-rectangle larger than *BigEnough()* in memory, and that it requires that every element be processed before anything can be known about the size of large holes. Our implementation of [3] performs well with small data sets (fewer than 1000 entries with 5 or fewer dimensions), and I recommend their algorithm for small to mid-sized data sets when one can reasonably estimate *BigEnough()*.

Realizing the limits of this approach, Liu *et al.* [16] introduced a method based on decision tree induction to facilitate discovery of large and interesting hyper-rectangles.

Edmonds *et al.* [17, 18] utilized a technique similar to Aggarwal and Suri's method [13], with a focus on database applications. Their algorithm for calculating maximal empty hyper-rectangles runs in $O(k\,n^{2k-2})$ and has the space complexity in $O(k\,n^{k-1})$.

Eckstein proved that the maximal box problem (and thus the maximal empty rectangle problem) is NP-hard, by reduction from the maximal clique problem [7].

Another related approach involves a query point algorithm to find the largest maximal empty hyper-rectangle that contains only one query [19]. In this method, a quadtree is used to realize a significant speed improvement, and has the advantage that there is no need for all the found rectangle objects to be maintained in memory. The approach has been shown to be especially useful for query optimization in databases.

Geometric approaches to join operations that use information about holes in databases have had much recent success as demonstrated by [20].

A related problem, the bichromatic rectangle problem, deals with finding the largest empty rectangle that contains only blue but no red dots. Backer *et al.* described a set of solutions to this problem, and showed that it is identical to the maximum empty hyper-

rectangle problem. Furthermore, they present an exact algorithm running in $O(n^k\, log^{2k}\, n)$ time, for any $k \geq 3$ [11].

Dumitrescu *et al.* presented the first efficient $1 - \epsilon$ approximation algorithm for the problem of finding the size of the largest MEHR. They showed that the minimum size of any maximum maximal empty hyper-rectangle has a volume of $1/n$ on a unit cube [21] .

Lastly, a new Monte Carlo method for finding the largest hole in polynomial time [2], from which some of the material in this thesis is duplicated, will be introduced.

Computer Vision: Face Attribute Classification

Face attribute classification is an important area within computer vision. Recognizing faces is one of the first things that humans learn to do as infants [22], and building computers that can recognize facial features with near human accuracy is a major goal.

Classifiers such as Support Vector Machines (SVM) and Feedforward Neural Networks (FNN) are often used to classify images after the faces have been cropped out from the rest of the image, and possibly aligned and normalized. Various feature extraction methods such as Principal Component Analysis (PCA), independent component analysis, Fischer linear discriminants [23] [24], and edge detection algorithms can be used to encode useful information from the image that is fed into the classifier, leading to high levels of accuracy on many benchmarks. Other approaches use hand-crafted template features to find facial keypoints such as nose, eyes etc, while also using edge detection methods (Sobel) and line intensities to separate facial edges from wrinkles. The resulting feature information, when fed into a feedforward neural network, allows age and gender to be classified with overall 85% accuracy on two test sets with a total of 172 images in the FERET and FGNET databases [25].

Linear Discriminant Analysis (LDA) based approaches to the face recognition task promise invariance to differing illuminations [23]. This has been further studied in [26]. Fisher linear discriminant maximizes the ratio of between-class scatter to that of within-class scatter. Independent component analysis has been used on a small subset (500 images) of the FERET dataset, leading to 96% accuracy with an SVM classifier [27]. Likewise, PCA has been used in conjunction with a genetic algorithm that eliminates potentially unnecessary features. The remaining features were then fed to a feedforward neural network for training, and an overall 85% accuracy was obtained over 3 data sets [28]. Various information theory based metrics were also fused together to produce 99.13% gender classification accuracy on the FERET [29]. To overcome the challenge of inadequate contrast among facial features using histogram analysis, Haar wavelet transformation and Adaboost learning techniques have been employed, resulting in a 97.3% accuracy on the Extended Yale face database which contains 17 subjects under 576 viewing conditions [30]. Another experiment describes how various transformations, such as noise and geometric transformations, were fed in combination into a series of RBFs (Radial Basis Functions). RBF outputs were forwarded into a symbolic decision tree that outputs gender and ethnic class. 94% classification accuracy was obtained using the hybrid architecture on the FERET database [31].

Histogram of Oriented Gradients (HOG) is commonly used as a global feature extraction technique that expresses information about the directions of curvatures of an image. HOG features can capture information about local edge and gradient structures while maintaining degrees of invariance to moderate changes in illumination, shadowing, object location, and 2D rotation. HOG descriptors, combined with SVM classifiers, can be used as a global feature extraction mechanism [32], while HOG descriptors can be used on locations indicated by landmark-finding software in areas such as facial

expression classification [33]. One useful application of variations in HOG descriptors is the automatic detection of pedestrians, which is made easier in part because of their predominantly upright pose [34]. In addition, near perfect results were obtained in facial expression classification when HOG descriptors were used to extract features from faces that were isolated through face-finding software [35].

A recent technique proposed for face recognition is the Dual Tree Complex Wavelet Transform (DTCWT), due to its ability to improve operation under varying illumination and shift conditions when compared to Gabor Wavelets and DWT (Discrete Wavelet Transform). The Extended Yale B and AR face databases were used, containing a total 16128 images of 38 human subjects under 9 poses and 64 illumination conditions. It achieved 98% classification accuracy in the best illumination condition, while low frequency subband images at scale one (L1) achieved 100% [36].

Recent years have seen great success in image related problems through the use of Convolutional Neural Networks (CNN), thereby seeing the proliferation of a scalable and more or less universal algorithmic approach to solving general image processing problems, if enough training data is available. CNNs have had a great deal of success in dealing with images of subjects and objects in natural non-contrived settings, along with handling the rich diversity that these images entail. One investigation of CNN fundamentals involved training a CNN to classify gender on images collected on the Internet. 88% classification accuracy was achieved after incorporating laplacian regularization into training, and filters were shown to respond to the same features that neuroscientists have identified as fundamental cues humans use in gender classification [37]. Another experiment [38] uses a convolutional neural network on the Adience dataset for gender and age recognition. They used data augmentation and face cropping to achieve 86% accuracy for gender classification.

A method recently proposed by [10] uses an SVM with dropout, a technique inspired from newer deep learning methods, that has shown promise for age and gender estimation. Dropout involves dropping a certain percent of features randomly during training. They also introduce the Adiance dataset to fulfill the need for a set of realistic labeled images for gender and age recognition in quantities needed to prevent overfitting and allow true generalization [10].

Each of these methods represent data as a set of points in high dimensional space, which can be analyzed using the new Monte Carlo algorithm as shown later.

CHAPTER III

A MONTE CARLO ALGORITHM FOR THE LARGEST EMPTY HYPER-RECTANGLE PROBLEM

A Monte Carlo Approach

In the following, the new Monte Carlo algorithm is described.

The Algorithm

Step 1: Algorithm Preparation

As a first step, retrieve points from a file or database, removing any attributes for which distance is undefined or where each element has the same value. Each remaining attribute column is assigned a number from 0 to k-1, and is treated as a dimension in a k dimensional space. Next, scale the data to the range [0, 1]. For each dimension, create a sorted list of attribute values with duplicates removed, which is important for a later step in being able to guarantee the emptiness of generated maximal empty hyper-rectangles. These lists can be considered as orthogonal projections of all points onto an axis.

Step 2: Generating Maximal Empty Rectangles

In this phase, create maximal empty hyper-rectangles using randomly generated query points that lie between any two points in the sorted list, explained in **Step 1**, as shown in Figure 2. In 2D, the nearest points to the randomly selected point in the positive and negative directions of each dimension specify the bounding points of an initial empty hyper-rectangle along that dimension. Because these points are distinct and sorted, it is provable that the resulting rectangle is empty.

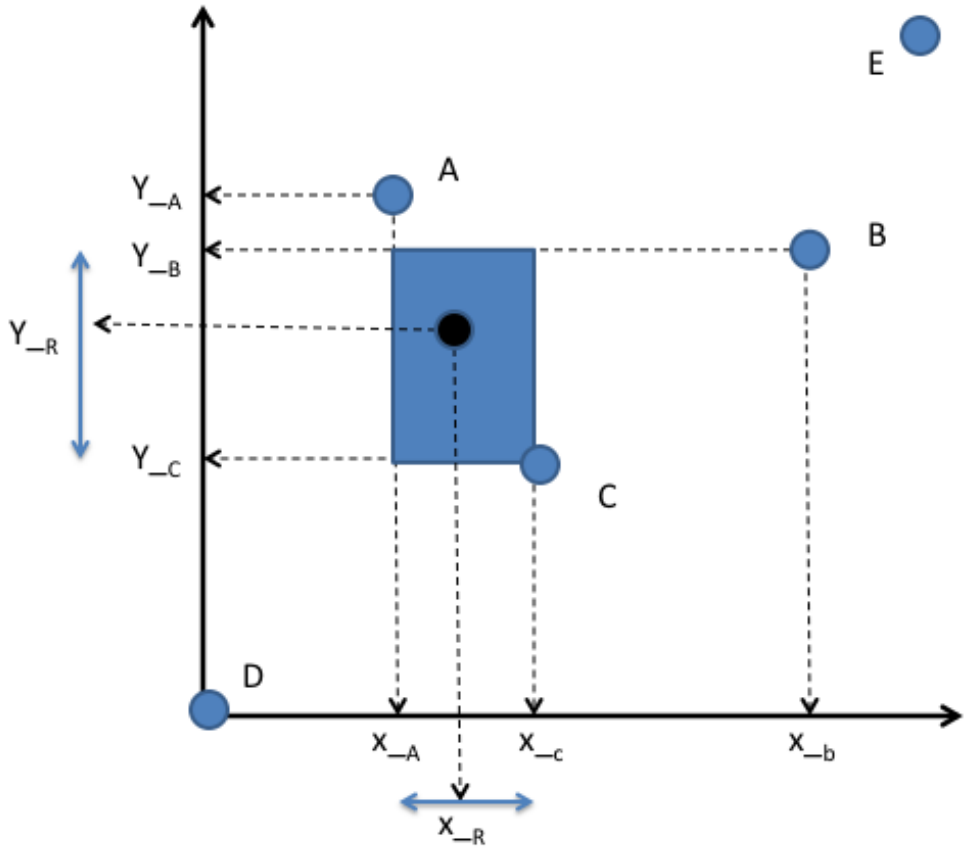


FIGURE 2: Generating maximal empty rectangles. Five data points, A through E, in 2D. Points D and E constrain the maximal size of dimensions x and y. Points A through C are projected onto the x and y axes, shown as y_a, y_b, y_c and x_a, x_b, x_c. A Monte Carlo randomly selected point, the black dot at coordinates x_r, y_r is chosen. Its x and y coordinates are expanded (blue arrows) in both directions, until another point in that coordinate is detected. The maximal expansion of both axes specifies the maximum bounding rectangle.

The found rectangle is empty due to the following:

- For a rectangle to be non-empty there must exist at least one point inside it.
- A point can be said to be inside a rectangle if, for each dimension k_i, the value of the point at k_i is between the upper and lower bounds of the rectangle at k_i.

– Assume the point is in the hyper-rectangle making the hyper-rectangle non empty. Then the value of a point at k_i is in a sorted list of distinct points, which means that the point must be either the upper bound or lower bound of the rectangle (from description of how rectangles are created). This leads to a contradiction because the point cannot be both a bounding point and a point between bounding points without having a value greater than itself or less than itself.

Step 3: Expansion

Once a rectangle has been found, enlarge it until it is maximal while still being empty. A hyper-rectangle is maximal and empty if it cannot be expanded along any dimension without containing one or more points. In Figure 2 the found rectangle is not maximal because it can be expanded into several directions before it borders either the boundary of the space or is abutting one of the points.

I have developed several strategies for expanding rectangles that produce maximal empty hyper-rectangles, but care must be taken not to exclude or bias rectangles with regard to any particular dimension. For example, if dimension 1 is always expanded first, then dimension 2, and so on, then dimension *k*-1 of the rectangle will usually have a much smaller width than it would have if it had been processed in random order. It is therefore important to randomize the order of expansion so that no single attribute is given any bias.

When expanding rectangles, there are 3 strategies that are useful depending on the features of the hyper-rectangles one is looking for.

In all expansion strategies the order in which dimensions are processed is shuffled.

Expansion Approach 1

For each dimension, maximally expand along that dimension before proceeding to the next. This strategy will find maximal rectangles that have many sides that are

shared with the unit square, and it is particularly useful when one wants to find ranges of values for which most attributes, if eliminated, have no meaningful impact on the resulting if/then rules. These rectangles will be maximally wide (bound by 0 and 1) in some dimensions, but more narrow in others, as shown in Figure 3.

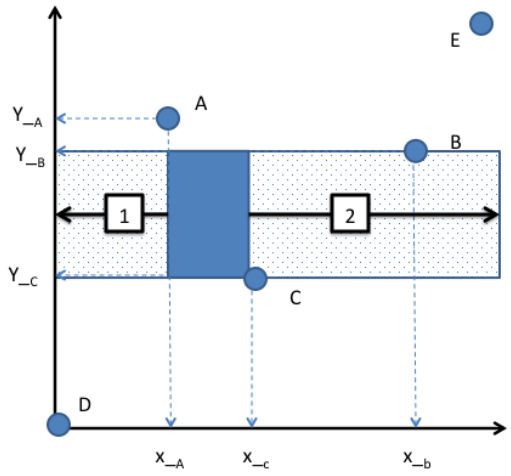
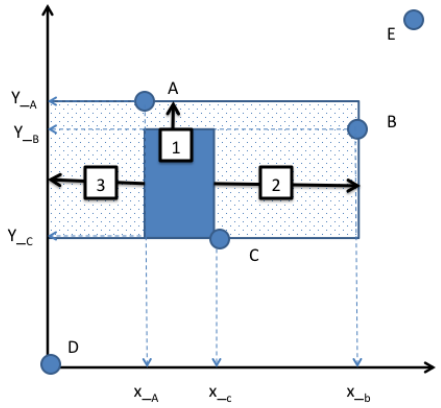


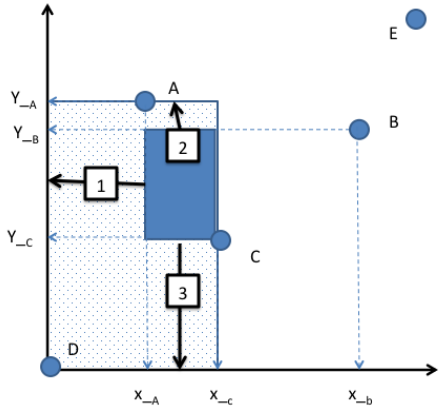
FIGURE 3: Expansion Approach 1 expanding maximally in a direction. Here, the *x* axis was randomly selected to expand first, and it was expanded first into the negative direction, then the positive (shown numbered as black boxes), to generate the maximum rectangle indicated with blue dots.

Expansion Approach 2

Expand along each dimension equally, stop expanding along a dimension when it becomes bound by a point and keep expanding along the other dimensions. This strategy tends to find maximal hyper-rectangles for which the widths along each dimension are similar.



(a) Expanding ”up” first, then ”right,” and finally ”left”



(b) Expanding ”left,” then ”up” the ”down”

FIGURE 4: Expansion Approach 2. The order of how dimensions are expanded determines which hyper-rectangle is found. Numbers in squares specify the order of expansion. In (a), expanding in the ”up” direction first, followed by the ”right” direction, and finally to the ”left” gives a different maximum hyper-rectangle than if the expansion order were different. Note that the second expansion in (a) prevents the eventual maximum rectangle from proceeding below point C, but in (b) the different order of expansion finds a maximum rectangle that continues to the the bottom-most boundary formed by point D.

Expansion Approach 3

Expand a random amount along each dimension until the rectangle is maximally expanded in all dimensions without containing any points. This strategy contains no statistical bias for any particular type of hyper-rectangle.

Step 4: Finding interesting maximal empty hyper-rectangles

The goal is to find interesting maximal empty hyper-rectangles that give insight into the data set. I propose a method to find empty hyper-rectangles that are potentially interesting. As in [3] and [17], it is assumed that large empty rectangles, particularly rectangles with volumes close to the size of the largest maximal empty hyper-rectangle, have the greatest chances of being truly interesting. In order to exploit this, I take the approach of discounting any empty rectangle with a volume less than $\frac{1}{n}$.

This is reasonable because, according to Dumitrescu *et al.* [39], for a fixed k, the maximum volume is on the order of $\Theta(\frac{1}{n})$. The volume of the largest box is $\Omega(\frac{1}{n})$. Furthermore, unless the points are distributed in an equidistant manner, any large rectangle will have a greater volume than $\frac{1}{n}$. For this reason, maximal empty hyper-rectangles with volumes of $\frac{1}{n}$ or less can be described as clusters, or areas with many points close together. Figure 5 shows that, in a 1D space, five equidistant points create four maximal holes, each with areas proportional to approximately $\frac{1}{n}$.

FIGURE 5: Five equidistant points create four maximal holes each with an area $\frac{1}{n-1}$.

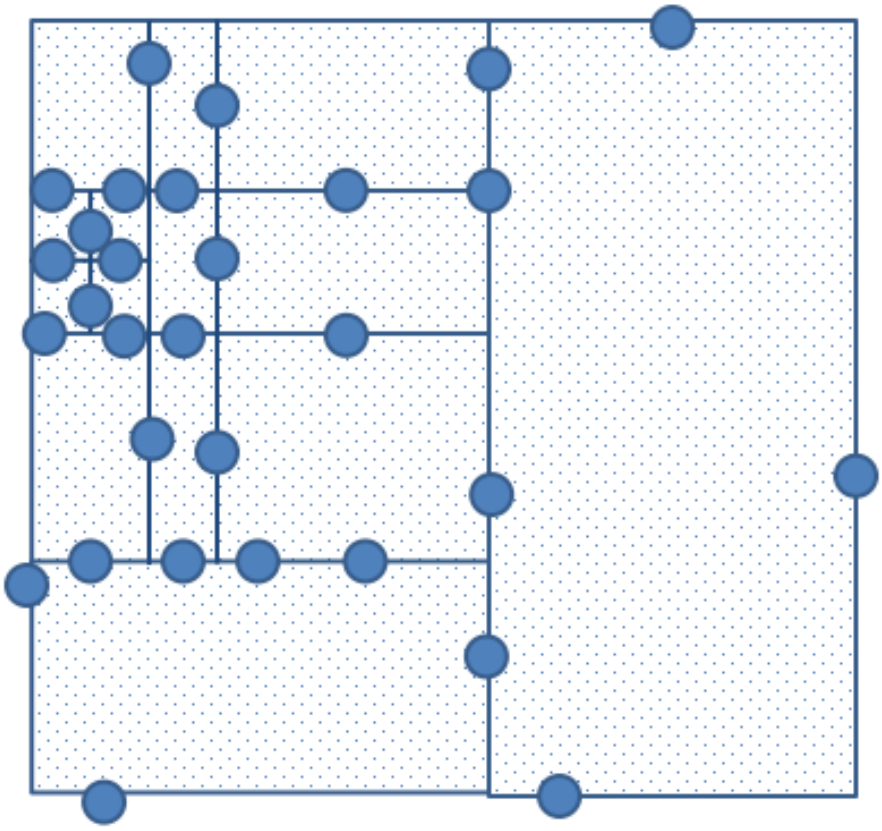


FIGURE 6: Visualization of Monte Carlo Approach. A dart randomly thrown on a board with both large and small rectangles has a significantly higher chance of landing in a large rectangle than in a small one.

Fortunately, due to the nature of this algorithm, rectangles with a volume less than $\frac{1}{n}$ are very rare, because there is a strong statistical bias for finding large rectangles. To visualize this, imagine a dart randomly landing on a board with both small and large rectangles drawn on it. The dart has a much higher chance of landing in a large rectangle than in a small one, as shown in Figure 6.

This tends to become even more of a factor as the number of dimensions k increases. For example, in one computational experiment on an artificial data set, the algorithm landed in a rectangle with a volume less than or equal to $\frac{1}{n}$ only 14 times out of 108,509 randomly selected initial points.

Algorithms 1 and 2 provide the pseudocode of my approach.

Algorithm 1 Pseudocode

```
1:  procedure FINDMEHRS(data, stop)                                   ▷
2:      data ← normalize(data);              ▷ Normalize 0 to 1 in each dimension
3:      projections ← createOrthProjections(data);
4:      n ← data.size()
5:      tooSmall ← 1/n
6:      c ← 0
7:      maxFound ← 0
8:      while c > stop do                                  ▷ Run Monte Carlo step
9:          mehr ← createMEHR(data, projections)
10:         if mehr.volume() > MaxFound then
11:             maxFound ← MEHR.VOLUME()
12:             MEHRLIST.APPEND(mehr)
13:             c ← 0
14:         else
15:             if mehr.volume() > tooSmall then
16:                 c ← c + 1
17:             end if
18:         end if
19:     end while
20:     mehrList.sort()
21: end procedure
```

Complexity

This algorithm's complexity is determined by the number of comparisons between the upper and lower bounds along each dimension with the value of a point on that dimension.

To create an empty hyper-rectangle, perform a binary search for the lower and upper bounds surrounding a randomly generated point. The number of comparisons is in $O(k \log n)$ if the number of dimensions is included in the input.

The enlargement step works by expanding the empty hyper-rectangle along each axis until it is no longer empty. The maximum number of expansions for each dimension

Algorithm 2 createMEHR

```
1: procedure CREATEMEHR(data, projections)
2:     for all k ∈ projections do
3:         r ← rand(0, 1)
4:         L_k,U_k ← findBoundingPoints(r, projections)        ▷ Binary Search to find bounds on k
5:     end for
6:     ehr ← createR(U, L)                                    ▷ Create MEHR from bounding points
7:     mehr ← expandRectangle(ehr, projections)               ▷ Expand rectangle using one of the strategies discussed return mehr
8: end procedure
```

is equal to the number of distinct points in the orthogonal projection of the unit hyper-cube onto that axis. This number is at most n. Therefore, the complexity of an expansion is in $O(k\,n)$. It is important to note that this maximum can only be realized in extreme cases, such as when all points are bounding points of the unit square in all dimensions.

Every time an expansion step is executed, it must be verified that the rectangle is still empty, which requires $2n$ comparisons on k dimensions meaning at most $2nk$ comparisons (the two comes from the fact that one must compare the upper and lower bounds of the rectangle with the value of the point). This means that the growth step is in $O(k^2\,n^2)$ with respect to the number of comparisons.

Considering both processes of creating an initial empty hyper-rectangle and growing that rectangle, the overall complexity of the algorithm is in $O(k^2\,n^2 + k\,log\,n)$ = $O(k^2\,n^2)$. This is the first published algorithm for finding holes in data which has polynomial time with respect to the number of dimensions.

Finally, the space complexity of the algorithm is in $O(n\,k)$ for storing the data points. I have thus accomplished my goal of designing an algorithm that scales well with the number of dimensions for finding maximal empty hyper rectangles in a set of points.

Validity

It is necessary not only to demonstrate that the algorithm is fast but also that, despite its use of randomization, it produces the largest hyper-rectangle with regularity. To accomplish this goal, I compare the algorithm's performance on various published machine learning benchmarks. In all cases, the full data set is used, and the size is only reduced in cases where it is necessary to obtain a result from existing algorithms for comparisons.

Liu's algorithm [3] was chosen for the purpose of comparison because it has been proven to identify the largest empty hyper-rectangle, and its *BigEnough()* heuristic greatly cut down on computation time compared to other solutions. It is also the computationally fastest published algorithm available for comparison. I measure the amount of time required for the algorithm to find the same rectangle produced by [3].

The goal of these experiments is to show that, despite starting with random points, in all of my experiments, my algorithm consistently finds the same largest empty hyper-rectangle in a reasonable amount of time. By "consistent" I understand that even if I cannot guarantee that the algorithm will always find the best solution, practically, this is usually the case.

To facilitate reproducibility I use four data sets from the well known UCI machine learning repository [40].

1. The Iris data set contains measurements of the septal and petal widths and lengths for 3 classes of iris plants with 150 items total and 4 dimensions.

2. The Combined Cycle power plant data set [41] contains 9568 attributes and 5 dimensions. The dimensions are composed of physical measurements such as

temperature, humidity, pressure, exhaust vacuum, and electric output as measured by sensors around the plant.

3. The User Knowledge modeling data set contains information about student knowledge of electrical DC machines. It has 5 dimensions and 255 items. I only use the first 100 points of this data set because the comparison algorithm was unable to finish within a reasonable amount of time on the full 255 points.

4. The Wilt dataset [42] is a high-resolution remote sensing data set containing information about tree wilt. I use the training set without the categorical data (i.e., 4339 5-dimensional vectors).

TABLE 2: Execution Time

data set	Liu's algorithm (s)	my algorithm (s)
1	3.845	0.077
2	521.426	97.46
3	1523.75	0.44
4	19916.4	3.6

The above table shows the runtime of my implementation of [3] compared with my algorithm using expansion strategy 3 averaged over 100 runs. The databases chosen represent a variety of types of data. Additionally, I found the same largest MEHR but in a fraction of the time, which is an added benefit of this approach.

An Application in Bioinformatics

To demonstrate the use of this algorithm, I apply it to a large data set of 39 metrics for approximately 5,000 protein structures. I then show how the algorithm's output can provide previously unknown insights about the data.

Motivation and Data Set Description

Proteins are three-dimensional dynamic structures that mediate virtually all cellular biological events. They are composed of long chains of amino acids, and knowing how they move in order to perform their functions is fundamental in designing medicines that regulate disease-causing proteins.

The Protein Data Bank (PDB) is a repository of the atomic x, y and z coordinates of over 100,000 protein structures that have been revealed using experimental methods [43]. Unfortunately, the prevalent experimental technique, X-ray crystallography, that is used to infer the locations of the atoms of a protein does not provide information about the mechanical – flexing and bending – properties of proteins [44].

Other techniques, both experimental and computational, provide supplementary information about proteins in addition to the information in a PDB file. These include Molecular Dynamics and rigidity analysis, as well as cavity data that describes a protein's surface in 3D.

Molecular Dynamics (MD) is one computational technique developed in the 1970s for inferring the positions and motions of atoms in a protein [45]. It involves solving various physics equations to infer how atoms in a protein move in response to repulsion and attraction forces among the amino acids in a protein.

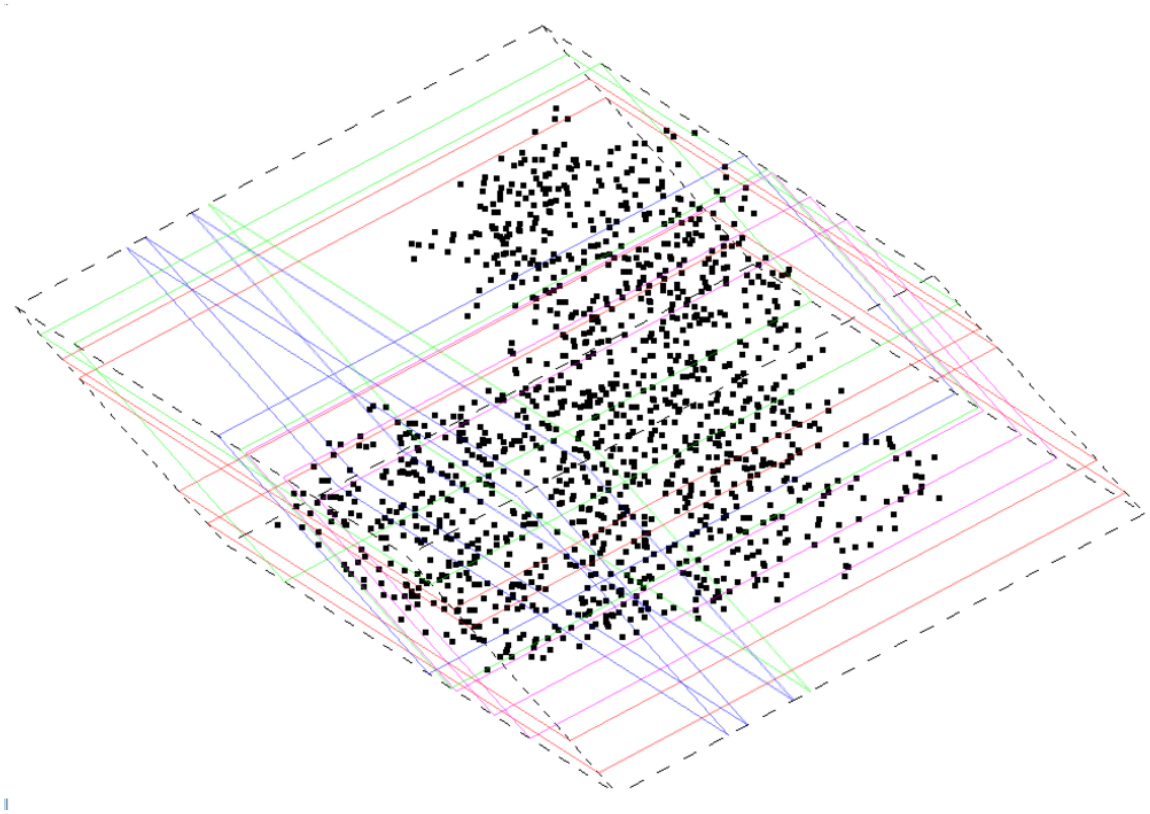
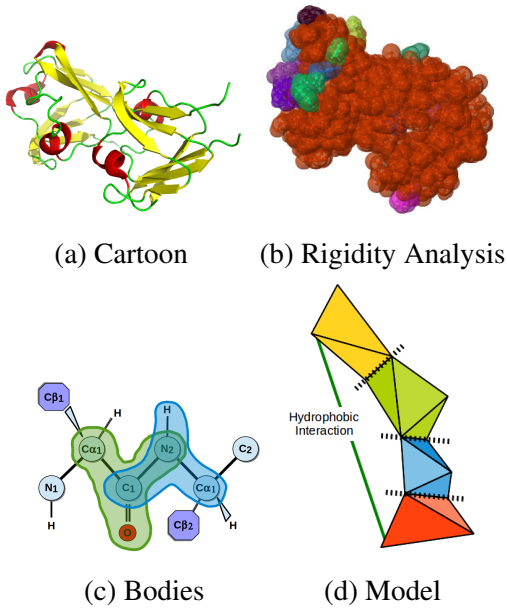


FIGURE 7: The 3D coordinates of HIV-1 protease graphed with several large holes found by the Monte Carlo algorithm.

According to Jacobs et al., rigidity analysis [46] [47] "is a fast graph-based method, complimentary to MD, that gives information about a protein's flexibility properties. In rigidity analysis, atoms and their chemical interactions are used to construct a mechanical model of a molecule, in which covalent bonds are represented as hinges, and other stabilizing interactions such as hydrogen bonds and hydrophobics are represented as hinges or bars. A graph is constructed from the mechanical model, and efficient algorithms based on the pebble game paradigm" [48] are used to analyze the rigidity of the graph. The rigidity properties of the graph are used to infer the rigidity properties of the protein structure, as shown in Figure 8.



(a) Cartoon (b) Rigidity Analysis

(c) Bodies (d) Model

FIGURE 8: The rigidity properties of biomolecule (PDB structure 4fd9, cartoon rendering in (a)) is shown in (b), where atoms in the same colored region specify atoms that are part of a rigid cluster. Bodies (c) identified from chemical properties of the atoms in the protein are used to generate a mechanical model on which a pebble game algorithm is run to infer the flexible and rigid regions.

Information about cavities on a protein's surface is other information that supplements the rigidity and molecular data about a protein structure. There exist efficient algorithms for identifying cavities on the surface of proteins. The open source software *fPocket* [49] was used to identify the cavities and the properties of their approximately 5,000 protein structures.

Combining these and other pieces of information yields high-dimensional data about each protein structure.

For this work, the data set is built up using biological data from the PDB, rigidity data as output by the KINARI rigidity analysis software [50, 51], and cavity data for a protein as output by *fPocket*. For 5,522 protein structures randomly selected from the Protein Databank, rigidity analysis and quantification of cavity information was

performed. The combined size of the zipped rigidity, cavity, and PDB data set is 2TB. For each protein structure, the metrics shown in Table 3 are extracted.

TABLE 3: Summary of 39 Dimensional Data.

Dimension(s)	Description
1-7	**Biological Data, Number of :** atoms, total bonds, hydrogen bonds, single covalent bonds, double covalent bonds, hydrophobic interactions, resonance bonds
8-28	**Residue Data, Number of :** total residues, ALA, ARG, ASN, ASP, CYS, GLN, GLU, GLY, HIS, ILE, LEU, LYS, MET, PHE, PRO, SER, THR, TRP, TYR, VAL
29-33	**Rigidity Properties, Number of :** Hinges, Bodies, Degrees of Freedom (DOF), size of largest rigid cluster, average cluster size
34, 35	**Largest Cavity :** Surface Area in Å^2, number of residues
36, 37	**Second Largest Cavity :** Surface Area in Å^2, number of residues
38, 39	**Third Largest Cavity :** Surface Area in Å^2, number of residues

Notes: Upper-case three-letter combinations refer to the 20 naturally occurring amino acids.

Experimental Setup

The following experimental results are conducted on a 12-core Intel® Xeon® 2.00GHz server with a NUMA memory architecture and 16,434,424 kB of memory.

The number of attributes in the 5,522 protein structures is 39, which gives a theoretical maximum of up to 5522^{28} or approximately 6.007576×10^{104} maximal empty hyper-rectangles per the upper-bounds approach mentioned in [3]. Note that using the approach in [3], all 5522^{28} rectangles would have to be found and then ranked according to size to find the largest ones, which is computationally infeasible.

My goal was to determine if my algorithm can find the largest rectangles, or, failing that, rectangles that were large enough to be interesting. I conducted 100 experiments with 100,000 rectangles found as a stop condition using expansion strategy 1. Each experiment averaged 20 minutes to complete on one processor and generated approximately 250MB of data. All rectangles along with metadata were output in a plain text file.

Experimental Results

Each of the 100 experiments located the same largest hyper-rectangle, having a volume of 0.4375 on the unit hyper-cube.

From the hyper-rectangles generated by this algorithm, I wanted to find out which (if any) of the 39 dimensions in the protein data were most frequently the bounding conditions of the largest found hyper-rectangles.

In addition, it is desirable to infer interesting relationships among the 39 dimensions. For example, is it true that there is a correlation between the number of Alanine (ALA) residues in a protein and the size of the largest cavity? The question posed in the general sense is the following:

> **If dimension x is a condition that very frequently is a bounding condition of the 100 largest hyper-rectangles, does that coincide with dimension y not being a bounding condition, and vice versa?**

To achieve this, I developed scripts to generate if/then rules based on dimensions in the data set that were bounding regions for the largest hyper-rectangles found. The if/then rules are easily explained with a 2D example. In Figure 9, on an input data set specified by the blue points in 2D, the maximal rectangle shown is described by the following rule:

If x is between 0.3 and 0.8 and y is between 0 and 1, then the rectangle bound by these points is empty.

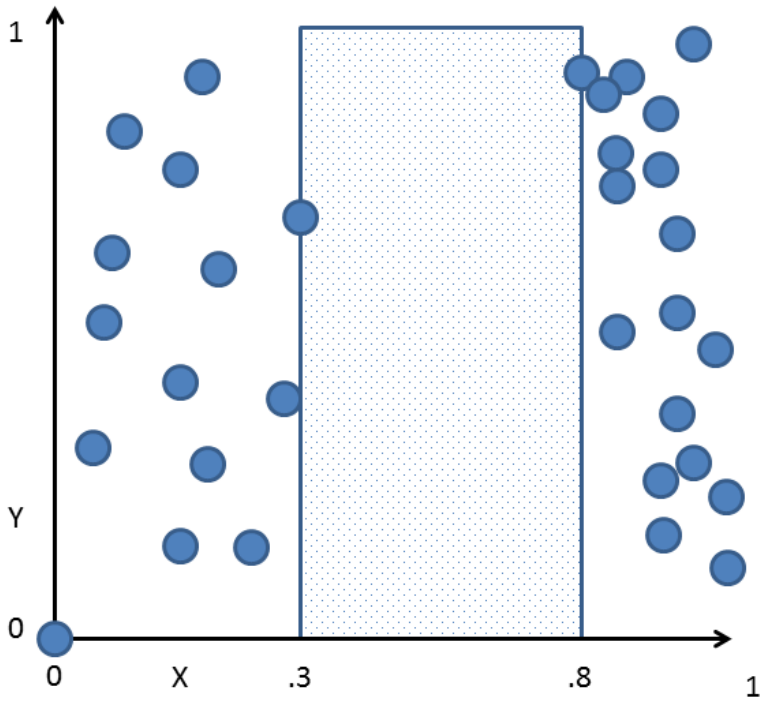


FIGURE 9: Explanation of if/then rule. In this 2D example, the maximum rectangle is defined by: x is between 0.3 and 0.8 and y is between 0 and 1.

Extracting the if/then rules from the hyper-rectangle output generated by the algorithm, I find several interesting relationships among the 39-dimensional data.

For instance, whenever the Histidine (HIS) count in a protein is in the range 81 to 144, then Glutamic Acid (GLU) is always 34 and the Serine (SER) count is always 74. This implies a relationship between Histidine, Glutamine and Serine counts in cases where the Histidine count is greater than 81 but less than 144. It also implies that Histidine counts specifically, and residue counts in general have the greatest impact on the location of holes within the data set. Although I do not ascribe any biological implications based on these findings, what is important is that these relations would not

have been found had I used any algorithm that had to exhaustively enumerate all large hyper-rectangles.

To investigate the likelihood that the Monte Carlo approach finds the largest hyper-rectangles without needing to enumerate them all, I also explore the number of rectangles that this algorithm considers before finding the (most-likely) largest one. These experiments all involve expansion strategy 1. I observe the following:

- The least number of rectangles that had to be considered before the largest was found is 117
- The highest number of rectangles that had to be considered before the largest was found is 12,183
- The median number of rectangles that had to be considered before the largest was found is 1,816.5
- The average number of rectangles that had to be considered before the largest was found is 2724.34

From these observations, it can be deduced that this algorithm only needs to examine a small fraction of the theoretical maximum of 6.007576×10^{104} possible hyper-rectangles.

Why does this approach work so well? The explanation is that this approach exploits the fact that large hyper-rectangles take up more space and thus are more likely to be found by chance.

CHAPTER IV

A COMPARISON OF MACHINE LEARNING TECHNIQUES USED IN COMPUTER VISION

Although this chapter focuses specifically on the use of machine learning techniques for gender classification, these techniques are all used widely in computer vision for tasks ranging from self driving cars to identity verification. None of these techniques are specific to gender classification. The intent is to show how these techniques are used to solve a single, well defined, problem using gender classification as an example.

Data Sets

A number of databases exist that can be used to benchmark gender classification algorithms. Most image sets that contain gender labels suffer from insufficient size, and because of this, two of the larger publicly available datasets: Color-FERET [9] and Adience [10] were chosen.

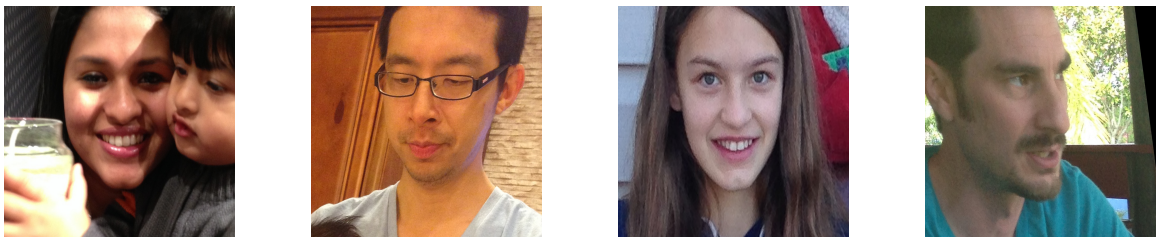


FIGURE 10: Randomly selected images from the Adience dataset illustrating the wider range of photographic conditions found.

Color FERET Version 2 was collected between December 1993 and August 1996 and made freely available with the intent of promoting the development of face

FIGURE 11: Randomly selected images from the FERET dataset show similarities in lighting, pose, subject, background, and other photographic conditions.

recognition algorithms. Images in the FERET Color database are 512 by 768 pixels and are in PPM format. They are labeled with gender, pose, name, and other useful labels.

Although FERET contains a large number of high quality images in different poses and with varying face obstructions (beards, glasses, etc), they all have certain similarities in quality, background, pose, and lighting which make them very easy for modern machine learning methods to correctly classify. In the following experiments, all 11338 images in FERET for which gender labels exist were used.

As machine learning algorithms are increasingly employed to process images of varying quality with vast differences in scale, obstructions, focus, and which are often acquired with consumer devices such as web cams or cellphones, benchmarks such as FERET have become less useful. To address this issue, datasets such as LWS (labeled faces in the wild) and most recently Adience have emerged. LWS lacks gender labels but it has accurate names from which gender can often be deduced automatically with reasonable accuracy.

Adience is a recently released benchmark that contains gender and approximate age labels separated into 5 folds to allow duplication of results published by the database authors. It was created by collecting Flickr images and is intended to capture all

variations of pose, noise, lighting, and image quality. Each image is labeled with age and gender. It is designed to mimic the challenges of "real world" image classification tasks where faces can be partly obscured or even partly overlapping (for example in a crowd or when an adult is holding a child and both are looking at the camera). Eidlinger, *et al.* published a paper where they used a SVM and filtering to classify age and gender along with the release of Adience [10].

All 19370 of the aligned images from Adience that had gender labels, were used to create the training and testing sets for all experiments that used Adience.

Using the included labels and meta data in the FERET and Adience datasets, two files containing reduced size 51x51 pixel data with values normalized between 0 and 1, followed by a gender label were generated. The images are then resized to 51x51 because this produced the best quality images after Anti-Aliasing. Later experiments with 100x100 and 12x12 images demonstrated that 51x51 provides fair performance for the range of methods used.

Classification Methods

The accuracy of CNN and several SVM based classifiers were compared. The analysis is limited to methods involving these two approaches because they are among the most effective and most prevalently used methods reported in the literature for Gender Classification.

Gender classifications with SVM is performed on the raw image pixels along with different well known feature extraction methods, namely DTCWT, PCA, and HOG. Training is done separately on two widely differing datasets consisting of gender labeled human faces: Color FERET, a set of images taken under similar conditions with good image quality, and Adience, a set of labeled unfiltered images intended to be especially

challenging for modern machine learning algorithms. Adience was designed to present all variations in appearance, noise, pose, and lighting, that can be expected of images taken without careful preparation [10].

The following steps were used in experiments in this chapter:

- Uniformly shuffle the order of images.
- Use 70% as training set, 30% as testing set.
- Train with training set.
- Record correct classification rate on testing set.

Steps 1-4 are repeated 10 times for each experiment using freshly initialized classifiers.

I now report on results of 18 experiments, 16 of which use SVMs and two of which use CNN.

SVM Classification

Both linear and RBF kernels were used, with each constituting a separate experiment using the SVC implementation included as part of scikit-learn[52] with $C = 100$ parameter set.

In one experiment, raw pixels are fed into the SVM. Other experiments used the following feature extraction methods: PCA, HOG, and DTCWT. Feature extraction was applied to images uniformly without using face finding software to isolate and align the face.

Histogram of Oriented Gradients

HOG descriptors, combined with SVM classifiers, can be used as a global feature extraction mechanism [32], while HOG descriptors can be used on locations indicated by landmark-finding software in areas such as facial expression classification [33].

The standard HOG implementation from the scikit-image library [53] was used.

For every image in the Adience and FERET databases, HOG descriptors were uniformly calculated. Nine orientation bins were used, and each histogram was calculated based on gradient orientations in the 7x7 pixel non-overlapping cells. Normalization was performed within each cell (i.e., 1 x 1). The result was fed into a SVM (SVC class from scikit-learn).

Principal Component Analysis

PCA is a statistical method for finding correlations between features in data. When used on images of faces, the resulting images are often referred to as Eigenfaces. PCA is used for reducing the dimensionality of data by eliminating non-essential information from the dataset and is frequently used in both image processing and machine learning.

To create the Eigenfaces, the RandomizedPCA tool within scikit-learn, which is based on work by [54] and [55] was used. The resulting Eigenfaces were then used in linear and RBF SVM.

Convolutional Neural Network

For the learning stage, I used a CNN with 3 hidden convolutional layers and one softmax layer. The training was done using a GTX Titan X GPU using the Theano [56] based library Pylearn2 [57] and CUDNN[58] libraries. Stochastic gradient descent was used as the training algorithm with a momentum of 0.95, found by trial and error.

Learning rates under 0.001 did not show any improvement. Increasing the learning rate above around 0.005 results in decreased classification accuracy.

A general outline of the structure of the CNN is:

- Hidden layer 1: A Rectified Linear Convolutional Layer using a kernel shape of 4x4, a pool shape of 2x2, a pool stride of 2x2 and 128 output channels. Initial weights are randomly selected with a range of 0.5.
- Hidden layer 2: A Rectified Linear Convolutional Layer using a kernel shape of 4x4, a pool shape of 2x2, a pool stride of 2x2 and 256 output channels. Initial weights are randomly selected with a range of 0.5.
- Hidden layer 3: A Rectified Linear Convolutional Layer using a kernel shape of 3x3, a pool shape of 2x2, a pool stride of 2x2 and 512 output channels. Initial weights are randomly selected with a range of 0.5.
- Softmax layer: Initial weights randomly set between 0 and 0.5. Output is the class (male or female).

Experimental Results

Tables 4 and 5 summarize the classification accuracy of each approach on each data-set after random shuffling and separation into 70% training and 30% testing sets. For each method the grayscale pixels were used as the features, either directly to the classifier, or to the filter mentioned. For example, HOG+SVM[RBF] indicates that the pixels were used as input to a HOG filter, the output of which is used as the input to a SVM with an RBF kernel.

I also trained a smaller CNN with an architecture optimized for extremely reduced facial images (12x12).

TABLE 4: Mean Classification Accuracy and Standard Deviation on Adience.

Method	Mean (%)	SD
CNN	96.1	0.0029
PCA+SVM[RBF]	77.4	0.0071
SVM[RBF]	77.3	0.0046
HOG + SVM[RBF]	75.8	0.006
HOG+SVM[linear]	75.0	0.0053
PCA+ SVM[linear]	72.0	0.0032
SVM[linear]	70.2	0.0052
DTCWT on SVM[RBF]	68.5	0.0059
DTCWT on SVM[linear]	59.0	0.0046

Notes: 70% of images used for training and 30% used for testing. Experiment repeated 10 times.

TABLE 5: Mean Classification Accuracy and Standard Deviation on FERET.

Method	Mean (%)	SD
CNN	97.9	0.0058
DTCWT on SVM[RBF]	90.7	0.0047
PCA+SVM[RBF]	90.2	0.0063
SVM[RBF]	87.1	0.0053
HOG+SVM[RBF]	85.6	0.0042
HOG+SVM[linear]	84.6	0.0024
DTCWT on SVM[linear]	83.3	0.0047
PCA+SVM[linear]	81.0	0.0071
SVM[linear]	76.5	0.0099

Notes: 70% of images used for training and 30% used for testing. Experiment repeated 10 times.

Accuracy on test set was 94% without data augmentation and 95% with data augmentation. It is surprising that results within 3% of the performance of the CNN on 51x51 images can be reached after losing so much information. Another surprising result is that the accuracy of PCA with SVM increased to 92%.

DTCWT was both the second best method (after CNN) and the very worst method examined; its performance has the greatest degree of variability depending on the dataset. It performs very well when objects are consistent in location and scale. CNN outperformed all methods. Even the worst CNN experiment on the most difficult dataset

performed better than the best of any other method on the easiest dataset. This is not a surprising outcome. It was found that HOG filters with SVM, without the usual additional models, provide no benefit on their own over raw pixel values for this experimental setup. PCA ties with DTCWT for the best performance on FERET but performs better than DTCWT on Adiance. As expected, RBF methods performed better than linear SVM classifiers; however, unexpectedly, this did not hold true for Adiance, where differences in filters were enough to cancel out the effect of RBF in some cases. Every time a filter was used on FERET, RBF was better than linear with filters. This did not hold for Adience. None of the filters worked particularly well on Adience, with only PCA slightly outperforming raw pixels for the RBF classifier.

On the FERET dataset DTCWT is better, but on Adience, it is worse. This would lend support to the idea that DTCWT seems to work better (in theory) on images that are more similar to FERET (uniform lighting, no complex backgrounds, no extreme warping, pixelation, or blurring).

Using an initial momentum of 0.95 tended to promote fast convergence without getting stuck in local minimum. I used a momentum of 0.95 and a learning rate of 0.001.

Using this setup, an average valid classification rate of 98% on FERET and 96% on Adience was achieved, which is better than the previous highest reported results according to [38] on Adience, but I do not recommend direct comparison of these results with theirs because of different experimental protocols used.

One of my aims was to investigate the use of the dual tree complex wavelet transform (DTCWT) on the face feature classification task. Several recent papers report success in using DTCWT in gender recognition from frontal face images citing the benefits of partial rotation invariance. It is somewhat unclear how to best use this for “in the wild” images.

CHAPTER V

MAXIMAL EMPTY HYPER-RECTANGLES IN COMPUTER VISION

Previously, I discussed a new algorithm for finding holes in high dimensional space that, until now, have been difficult to find, and demonstrated a novel application in computational biology. I then compared nine machine learning techniques for computer vision on two public machine learning datasets. I will now explore the types of information that can be learned from such holes for computer vision.

Holes in Images

Two types of holes may prove useful: holes in a single 2D image, and holes in an image set. In the case of the former, one wants to find areas of high contrast or areas which contribute most significantly to the overall variance in one image.

Holes in One Image

Grayscale images are typically represented as a 2 dimensional matrix of n by m values called pixels. The largest holes in these images represent values between the greatest changes in color (for color images) or luminance for grayscale images values. To preserve structural information, take the luminance as a third dimension making the dataset 3-Dimensional. Likewise, color images can be represented as 5-Dimensional data of x, y, red, green, and blue.

FIGURE 12: Step 1, example grayscale image with a large hole

4	0	0	0	0	0	240
3	0	0	0	0	0	240
2	155	155	155	155	155	240
1	200	200	200	200	200	240
0	240	240	240	240	240	240
	0	**1**	**2**	**3**	**4**	**5**

FIGURE 13: Step 2, pixel values of image above

1	0	0	0	0	0	1
0.75	0	0	0	0	0	1
0.5	0.6458	0.6458	0.6458	0.6458	0.6458	1
0.25	0.8333	0.8333	0.8333	0.8333	0.8333	1
0	1	1	1	1	1	1
	0	**0.2**	**0.4**	**0.6**	**0.8**	**1**

FIGURE 14: Step 3, scaled pixel values

As would be expected visually, the largest hole is found to be the area between the first two rows and the upper last column. This demonstrates that holes can be used to identify large local variations in color or intensity, and may have a use in image analysis as feature extractors or edge detectors.

Holes in Image Sets

Machine learning algorithms usually treat each pixel as an attribute (or feature), and each image as a point in $n \times m$ dimensional space where n and m are the width and

height of the image in pixels. Holes found in this space can be used to discover ranges of pixel values that are not present in the image set and to analyze the geometry of this high dimensional space.

Previously, it was not feasible to analyze such holes, but with the introduction of the new algorithm, such analysis is now possible.

Holes in CNN

A CNN learns filters that transform the feature space of images into another space. Since the feature space and the convolutional space at each layer is of high dimensionality, it has not previously been feasible to analyze the holes in this space.

For this experiment, I created a 100x100 gray scale version of FERET, and trained a CNN using it with the same experimental methodology as in the previous chapter. The software package Keras, which provides compatibility with both Theano and Tenserflow was used for this task because it provides more convenient access to the layer details during classification. The accuracy achieved was 95.4% without augmentation. The 2.5% difference in performance from the experiments reported in the previous chapter can be explained by the different implementations and architectures used.

The architecture of the CNN used in this experiment is as follows:

- A Convolutional layer with a size of 2x2 and output size of 64 connected to a rectified linear activation unit.
- A Convolutional layer with a size of 2x2 and output size of 64 connected to a rectified linear activation unit.
- A Maxpooling layer with a pool size of 2x2
- A Dropout layer (25%)

- A Convolutional layer with a size of 2x2 and output size of 128 connected to a rectified linear activation unit.
- A Convolutional layer with a size of 2x2 and output size of 128 connected to a rectified linear activation unit.
- A Maxpooling layer with a pool size of 2x2
- A Dropout layer (25%)
- A Convolutional layer with a size of 3x3 and output size of 256 connected to a rectified linear activation unit.
- A Convolutional layer with a size of 3x3 and output size of 256 connected to a rectified linear activation unit.
- A Maxpooling layer with a pool size of 2x2
- A Dropout layer (25%)
- A fully connected layer with size 1028, dropout of 50%
- A fully connected layer with size 512, dropout of 50%
- A fully connected layer with size 256, dropout of 50%
- A softmax layer with output of male or female class labels.

Adadelta [59] was used for training with cross-entropy as the error function.

I then passed FERET images into the CNN and used PCA to reduce the dimensionality of the data as output by 3 convolutional layers in a trained CNN to a 30-dimensional space. I then located the largest empty hyper-rectangle found in that

space and observed the changes in rectangles present in the output of these trained layers. Expansion strategy 1 was used for finding the holes.

Input layer: 0.17573

Second convolutional layer: 0.138413

Fourth convolutional layer: 0.199075

Sixth convolutional layer: 0.1436

Note that the largest holes are only taking up $< 20\%$ of the volume of the unit hyper cube, which is less than the author would have expected for a 30 Dimensional space. For perspective, the smallest hole found for the input layer was 0.000119134 indicating significant variability in the size of found holes but not as much as that found in the datasets analyzed in chapter 3. It is observed that the relative similarity in the size of the rectangles in this space made it more difficult for the algorithm to find the largest rectangle. The worst case for the algorithm is perfectly uniform, equidistant points, and while this data is not uniform it is closer to being such than the other datasets analyzed.

It is interesting that in each of the 4 layers examined, the size of the largest found hyper-rectangle was always between 13 and 19.

Although these results are preliminary, the use of holes provides us with information about the changes in the geometry as images pass through each layer. In the future, more research may be done on different datasets and without dimensionality reduction.

CHAPTER VI

CONCLUSIONS

In this thesis, the first algorithm for finding holes in high dimensional data that runs in polynomial time with respect to the number of dimensions has been developed. Such large hyper-rectangles have been shown to be particularly useful in data mining and data analytics [4, 16, 18] which motivated the development of a new technique for finding them that can be used in high-dimensional data sets such as those used in computer vision tasks. From the MEHRs that are found , one can extract the corresponding if/then rules, which can be used to locate interesting relationships among dimensions.

It was shown that this new algorithm can be used to mine empty hyper rectangles on datasets that would evade other methods, all of which are exponential in the number of dimensions. I have demonstrated the use of this new algorithm on a 39-dimensional data set containing more than 5,000 entries, and have inferred interesting relationships among the dimensions that could not have been found with other published hole finding algorithms.

This supports the premise that a Monte Carlo approach is likely to find large, useful empty hyper-rectangles quickly, simply because the probability of landing in a larger rectangle is greater than landing in a smaller one and in many cases it may be unnecessary to enumerate every hyper-rectangle before concluding that one has likely found the largest.

Much of the previous work on automatic gender classification uses differing datasets and experimental protocols that can make direct comparisons between reported results misleading. I have compared nine different machine learning methods used in gender recognition on two benchmarks, using identical research methodology to allow a

direct comparison between the efficacies of the different classifiers and feature extraction methods. In addition to providing updated information on the effectiveness of these algorithms, I provide directly comparable results.

One of the aims of this study was to explore gender classification using recent learning algorithms. I carried out experiments on several state-of-the-art gender classification methods. The accuracy of these methods on two very different data sets ("In the wild" verses posed images) were compared.

To the extent of my knowledge, this was the first use of DTCWT on a large $\geq 15,000$ database of "in the wild" images, specifically addressing gender classification. I have achieved an average accuracy of 98% (FERET) and 96% (Adience), which is better than the previous highest reported results (according to [38]) on Adience using a CNN.

It is observed that DTCWT seems to work better ($\approx 90\%$) on images that are more similar to FERET (uniform lighting, no complex backgrounds, no extreme warping, pixelation, or blurring). The Adience and FERET data sets are relatively large, which explains why the CNN method generally outperforms other methods: it is known that deep learning performs well when large training sets are being used.

Finally, I demonstrated how the new Monte Carlo based algorithm could be used to discover empty hyper-rectangles in the high dimensional image space used by machine learning algorithms as well as within images themselves.

REFERENCES CITED

[1] J. Lemley, S. Abdul-Wahid, D. Banik, and R. Andonie, "Comparison of recent machine learning techniques for gender recognition from facial images," in *Modern Artificial Intelligence and Cognitive Science Conference*, pp. 97–102, MAICS, 2016.

[2] J. Lemley, F. Jagodzinski, and R. Andonie, "Big holes in big data: A Monte Carlo algorithm for detecting large hyper-rectangles in high dimensional data," in *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, vol. forthcoming, IEEE press, 2016.

[3] B. Liu, L.-P. Ku, and W. Hsu, "Discovering interesting holes in data," *IJCAI (2)*, pp. 930–935, 1997.

[4] C. Ordonez, E. R. Omiecinski, S. B. Navathe, and N. F. Ezquerra, "A clustering algorithm to discover low and high density hyper-rectangles in subspaces of multidimensional data.," tech. rep., 1999.

[5] A. Dumitrescu and M. Jiang, "Computational geometry column 60," *ACM SIGACT News*, vol. 45, no. 4, pp. 76–82, 2014.

[6] J. Backer and J. M. Keil, "The mono-and bichromatic empty rectangle and square problems in all dimensions," *LATIN 2010: Theoretical Informatics*, pp. 14–25, 2010.

[7] J. Eckstein, P. L. Hammer, Y. Liu, M. Nediak, and B. Simeone, "The maximum box problem and its application to data analysis," *Computational Optimization and Applications*, vol. 23, no. 3, pp. 285–298, 2002.

[8] N. Kingsbury, "Complex wavelets for shift invariant analysis and filtering of signals," *Applied and Computational Harmonic Analysis*, vol. 10, no. 3, pp. 234 – 253, 2001.

[9] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, "The feret evaluation methodology for face-recognition algorithms," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 10, pp. 1090–1104, 2000.

[10] E. Eidinger, R. Enbar, and T. Hassner, "Age and gender estimation of unfiltered faces," *Information Forensics and Security, IEEE Transactions on*, vol. 9, no. 12, pp. 2170–2179, 2014.

[11] J. Backer and J. M. Keil, "The bichromatic rectangle problem in high dimensions.," in *CCCG*, pp. 157–160, 2009.

[12] F. Preparata, *Computational geometry : an introduction*. New York: Springer-Verlag, 1985.

[13] A. Aggarwal and S. Suri, "Fast algorithms for computing the largest empty rectangle," in *Proceedings of the third annual symposium on Computational geometry*, pp. 278–290, ACM, 1987.

[14] A. Datta and S. Soundaralakshmi, "An efficient algorithm for computing the maximum empty rectangle in three dimensions," *Information Sciences*, vol. 128, no. 1, pp. 43–65, 2000.

[15] L.-P. Ku, B. Liu, and W. Hsu, "Discovering large empty maximal hyper-rectangle in multi-dimensional space," 1997.

[16] B. Liu, K. Wang, L.-F. Mun, and X.-Z. Qi, "Using decision tree induction for discovering holes in data," *PRICAI98: Topics in Artificial Intelligence*, pp. 182–193, 1998.

[17] J. Edmonds, J. Gryz, D. Liang, and R. J. Miller, "Mining for empty rectangles in large data sets," *Database TheoryICDT 2001*, pp. 174–188, 2001.

[18] J. Edmonds, J. Gryz, D. Liang, and R. J. Miller, "Mining for empty spaces in large data sets," *Theoretical Computer Science*, vol. 296, no. 3, pp. 435–452, 2003.

[19] G. Gutiérrez and J. R. Paramá, "Finding the largest empty rectangle containing only a query point in large multidimensional databases," in *Scientific and Statistical Database Management*, pp. 316–333, Springer Berlin Heidelberg, 2012.

[20] M. Abo Khamis, H. Q. Ngo, C. Ré, and A. Rudra, "Joins via geometric resolutions: Worst-case and beyond," in *Proceedings of the 34th ACM Symposium on Principles of Database Systems*, PODS '15, (New York, NY, USA), pp. 213–228, ACM, 2015.

[21] A. Dumitrescu and M. Jiang, "On the largest empty axis-parallel box amidst n points," *CoRR*, vol. abs/0909.3127, 2009.

[22] B. I. Fagot and M. D. Leinbach, "Gender-role development in young children: From discrimination to labeling," *Developmental Review*, vol. 13, no. 2, pp. 205–224, 1993.

[23] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 7, pp. 711–720, 1997.

[24] Y. Wu, Y. Zhuang, X. Long, F. Lin, and W. Xu, "Human gender classification: A review," *arXiv preprint arXiv:1507.05122*, 2015.

[25] T. R. Kalansuriya and A. T. Dharmaratne, "Neural network based age and gender classification for facial images," *ICTer*, vol. 7, no. 2, 2014.

[26] J. Bekios-Calfa, J. M. Buenaposada, and L. Baumela, "Revisiting linear discriminant techniques in gender recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 4, pp. 858–864, 2011.

[27] A. Jain, J. Huang, and S. Fang, "Gender identification using frontal facial images," in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pp. 4–pp, IEEE, 2005.

[28] Z. Sun, X. Yuan, G. Bebis, and S. J. Loui, "Neural-network-based gender classification using genetic search for eigen-feature selection," in *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, vol. 3, pp. 2433–2438, IEEE, 2002.

[29] C. Perez, J. Tapia, P. Estévez, and C. Held, "Gender classification from face images using mutual information and feature fusion," *International Journal of Optomechatronics*, vol. 6, no. 1, pp. 92–119, 2012.

[30] P. Laytner, C. Ling, and Q. Xiao, "Robust face detection from still images," in *Computational Intelligence in Biometrics and Identity Management (CIBIM), 2014 IEEE Symposium on*, pp. 76–80, IEEE, 2014.

[31] S. Gutta, H. Wechsler, and P. J. Phillips, "Gender and ethnic classification of face images," in *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*, pp. 194–199, IEEE, 1998.

[32] P. A. Torrione, K. D. Morton, R. Sakaguchi, and L. M. Collins, "Histograms of oriented gradients for landmine detection in ground-penetrating radar data," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 52, no. 3, pp. 1539–1550, 2014.

[33] O. Déniz, G. Bueno, J. Salido, and F. De la Torre, "Face recognition using histograms of oriented gradients," *Pattern Recognition Letters*, vol. 32, no. 12, pp. 1598–1603, 2011.

[34] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, IEEE, 2005.

[35] P. Carcagnì, M. Del Coco, M. Leo, and C. Distante, "Facial expression recognition and histograms of oriented gradients: a comprehensive study," *SpringerPlus*, vol. 4, no. 1, pp. 1–25, 2015.

[36] M. Sultana, M. Gavrilova, R. Alhajj, and S. Yanushkevich, "Adaptive multi-stream score fusion for illumination invariant face recognition," in *Computational Intelligence in Biometrics and Identity Management (CIBIM), 2014 IEEE Symposium on*, pp. 94–101, IEEE, 2014.

[37] A. Verma and L. Vig, "Using convolutional neural networks to discover cogntively validated features for gender classification," in *Soft Computing and Machine Intelligence (ISCMI), 2014 International Conference on*, pp. 33–37, IEEE, 2014.

[38] G. Levi and T. Hassner, "Age and gender classification using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 34–42, 2015.

[39] A. Dumitrescu and M. Jiang, "On the largest empty axis-parallel box amidst n points," *Algorithmica*, vol. 66, no. 2, pp. 225–248, 2013.

[40] M. Lichman, "UCI machine learning repository." http://archive.ics.uci.edu/ml, 2013.

[41] P. Tüfekci, "Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods," *International Journal of Electrical Power & Energy Systems*, vol. 60, pp. 126–140, 2014.

[42] B. A. Johnson, R. Tateishi, and N. T. Hoan, "A hybrid pansharpening approach and multiscale object-based image analysis for mapping diseased pine and oak trees," *International Journal of Remote Sensing*, vol. 34, no. 20, pp. 6969–6982, 2013.

[43] J. Westbrook, H. M. Berman, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The protein data bank," *Nucleic Acids Research*, vol. 28, pp. 235–242, 2000.

[44] J. Kendrew, G. Bodo, H. Dintzis, R. Parrish, H. Wyckoff, and D. Phillips, "A three-dimensional model of the myoglobin molecule obtained by x-ray analysis," *Nature*, vol. 181, no. 4610, pp. 662–666, 1958.

[45] D. Case and M. Karplus, "Dynamics of ligand binding to heme proteins," *Journal of Molecular Biology*, vol. 132, no. 3, pp. 343–368, 1979.

[46] D. Jacobs, A. Rader, M. Thorpe, and L. Kuhn, "Protein flexibility predictions using graph theory," *Proteins 44*, pp. 150–165, 2001.

[47] D. Jacobs and M. Thorpe, "Generic rigidity percolation: the pebble game," *Physics Review Letters*, vol. 75, pp. 4051–4054, 1995.

[48] D. Jacobs and B. Hendrickson, "An algorithms for two-dimensional rigidity percolation: the pebble game," *Journal of Computational Physics*, vol. 137, pp. 346–365, 1997.

[49] P. Schmidtke, V. Le Guilloux, J. Maupetit, and P. Tufféry, "fpocket: online tools for protein ensemble pocket detection and tracking," *Nucleic Acids Research*, vol. 38, no. suppl 2, pp. W582–W589, 2010.

[50] N. Fox, F. Jagodzinski, and I. Streinu, "Kinari-lib: a C++ library for pebble game rigidity analysis of mechanical models," in *Minisymposium on Publicly Available Geometric/Topological Software, Chapel Hill, NC, USA*, June 2012.

[51] N. Fox, F. Jagodzinski, Y. Li, and I. Streinu, "KINARI-Web: A server for protein rigidity analysis," *Nucleic Acids Research*, vol. 39 (Web Server Issue), pp. W177–W183, 2011.

[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[53] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, "scikit-image: image processing in Python," *PeerJ*, vol. 2, p. e453, 6 2014.

[54] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.

[55] P.-G. Martinsson, V. Rokhlin, and M. Tygert, "A randomized algorithm for the decomposition of matrices," *Applied and Computational Harmonic Analysis*, vol. 30, no. 1, pp. 47–68, 2011.

[56] T. T. D. Team, R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, *et al.*, "Theano: A python framework for fast computation of mathematical expressions," *arXiv preprint arXiv:1605.02688*, 2016.

[57] I. J. Goodfellow, D. Warde-Farley, P. Lamblin, V. Dumoulin, M. Mirza, R. Pascanu, J. Bergstra, F. Bastien, and Y. Bengio, "Pylearn2: a machine learning research library," *arXiv preprint arXiv:1308.4214*, 2013.

[58] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," *arXiv preprint arXiv:1410.0759*, 2014.

[59] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.