Anomaly Detection for Cancer Prediction Using Convex Hull Method on a Normal Distribution Parameter Space

Julian Chandra Sutadi - 13522080
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
julian.sutadi@gmail.com

***Abstract*—This paper aims to apply the Convex Hull Method (Conv.Hull-PS) for anomaly detection in a cancer prediction model. A dataset of cancer patients with health-related features is analyzed and used to construct points inside a two-dimensional normal distribution parameter space. Another set of points is obtained from pairs of unknown data and known data. The area of intersection between the convex hull of the two sets is used to determine the value of anomalies, which are features suspected to be indicative of cancerous individuals.**

Keywords—divide and conquer; convex hull; anomaly detection; parameter space; learning algorithm.

I. Introduction

Cancer detection is crucial for effective treatment, and advanced computational methods offer promising tools for early identification. This paper explores anomaly detection for cancer prediction, using the Convex Hull Method within a normal distribution parameter space.

Anomaly detection, or outlier detection, is the identification of abnormal data points, that deviate from the expected behavior, making them inconsistent with the rest of the dataset. It is a concept useful in statistics and machine learning which attempts to identify unexpected changes in the dataset's normal behavior [1].

The convex hull, an important concept in computational geometry, can play an important role in anomaly detection. It is used to outline the shape of the convex polygon surrounding the data pair points in the two-dimensional parameter plane. Within cancer prediction, we construct a normal distribution parameter space based on health-related features of patients. Then, by employing the Convex Hull Method, we delineate the boundaries of known data points (from cancer patient records) and unknown data points.

Through this paper, we aim to demonstrate how convex hull, specifically the Conv.Hull-PS algorithm can be utilized in early cancer detection, ultimately improving outcomes for patients through early intervention and treatment.

II. Fundamental Theory

A. Convex Hull

Convex hull is an important part of computational geometry. A set of points on a planar plane is convex if and only if for every pair of points p and q on the plane, all line segments that have p and q as its endpoints are elements of the initial set of points. The convex hull of a set of points S is the smallest convex set (convex polygon) that contains S.

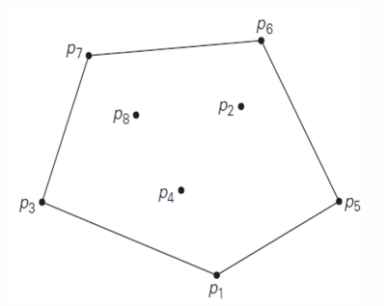

Fig. 1. Convex hull for eight points. [2]

One approach to find the convex hull of a set is by using the Quickhull algorithm which uses the Divide and Conquer approach, similar to that of Quicksort algorithm. For a set of points S in a 2-dimensional plane with n elements, the steps of Quickhull algorithm are as follows [2]:

1. Sort the elements by their x-coordinates in ascending order. If two elements have the same x-coordinate, sort them by their y-coordinates in ascending order. Let p_i be the i-th element of the sorted set.
2. The line connecting p_1 and p_n divides S into two parts: S_1 (the set of points on the left or above the line p_1p_n) and S_2 (the set of points on the right or below the line p_1p_n).
3. To check if a point is on the left (or above) a line formed by two points, use the determinant method

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \quad (1)$$

which is equivalent to

$$x_1y_2 + x_3y_1 + x_2y_3 - x_3y_2 - x_2y_1 - x_1y_3 \quad (2)$$

The point (x_3, y_3) is on the left side of the line (x_1, y_1) to (x_2, y_2) if the determinant result is positive.

4. All points in S that lie on the line p_1p_n (other than the pointsp_1 and p_n) cannot form a convex hull, so they can be ignored.
5. The set of points in S1 can form the upper part of the convex hull, and the set of points in S2 can form the lower part of the convex hull.
6. For a section (e.g. S_2), there are two possibilities:
    a. If there are no other points besides S_1, then the points p_1 and p_n form the convex hull of S_1.
    b. If S_1 is not empty, select a point that is the farthest from the line p_1p_n (call it p_{max}). If there are several points at the same distance, choose the point that maximizes the angle $p_{max}p_1p_n$. All points inside the triangle $p_{max}p_1p_n$ are ignored in further examination.
7. Determine the set of points that are on the left side of the line p_1p_{max} (which becomes $S_{1,1}$) and on the right side of the line p_1p_{max} (which becomes $S_{1,2}$).
8. Do steps 6 and 7 for S_2. Here we are dividing the problems into subproblems (Divide and Conquer).
9. Repeat from step 5 until there remains no point in the left or right of line p_ip_{max} or $p_{max}p_j$ for all generated p_{max} (p_{max} as defined in step 6).
10. Return the points p_1, p_n, and all generated p_{max}.

B. *Normal Distribution*

The normal distribution (also known as the Gaussian distribution) is a statistical distribution proposed by Carl Friedrich Gauss in 1809 which is by far the most important statistical distribution [3]. Below are some characteristics of the normal distribution:

1. Parameters: μ, σ
2. Range: $(-\infty, \infty)$
3. Notation: $\mathrm{N}(\mu, \sigma^2)$
4. Probability density function (pdf):

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-(x-\mu)^2/2\sigma^2} \quad (3)$$

The shape of the pdf is symmetrical and similar to a bell. It is therefore commonly called a "bell curve".

5. Cumulative density function (cdf):

$$\int_{-\infty}^{\infty} f(x)dx = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} \mathrm{e}^{-(x-\mu)^2/2\sigma^2} dx \quad (4)$$

A normal distribution is said to be standard (standard normal distribution) if the parameters are $\mu = 0$ and $\sigma^2 = 1$.

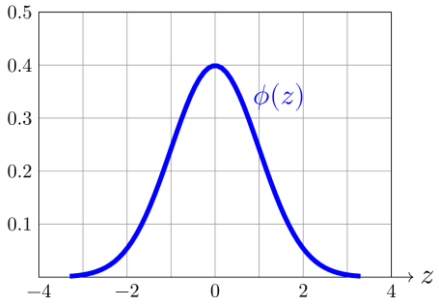

Fig. 2. A standard normal distribution. [3]

There are two notions of normal distribution that are important in anomaly detection.

1. The sum of normal random variables is also a normal random variable.
2. In analyzing data that are distributed normally, a common way of finding outliers is to calculate the value of 3σ. This value becomes a threshold to evaluate datum x, which will be classified as an outlier if $x < -3\sigma$ or $x > 3\sigma$.

C. *Central Limit Theorem*

Suppose $X_1, X2, \ldots, X_n$ are independent and identically distributed random variables with statistical parameters μ and σ. For each n, let S_n denotes the sum and $\bar{X}$ the average of $X_1, X2, \ldots, X_n$. By calculating the value of mean (μ) and variance (σ^2) from both S_n and $\bar{X}_n$, it will be discovered that both S_n and $\bar{X}_n$ have the same standardization value

$$Z_n = \frac{S_n - n\mu}{\sigma\sqrt{n}} = \frac{\bar{X}_n - \mu}{\sigma/\sqrt{n}} \quad (5)$$

The Central Limit Theorem (CLT) then states that for large n,

$$\bar{X}_n \approx \mathrm{N}(\mu, \sigma^2 n), \quad S_n \approx \mathrm{N}(n\mu, n\sigma^2), \quad Z_n \approx \mathrm{N}(0,1). \quad (6)$$

Therefore, according to the CLT, the mean distribution of random variables for a large number of data will be a standard normal distribution regardless of the underlying distribution of the random variables.

D. *Anomaly Detection Using Convex Hull on a 2D Normal Distribution Parameter Space*

Anomaly Detection is a task in machine learning of finding unusual patterns (or outliers) that do not conform to the expected behavior of the dataset. Reference [4] proposed a method of anomaly detection called Conv.Hull-PS that involves an analysis of the area of overlapping region between two convex hulls over a 2-dimensional normal distribution parameter space (μ, σ).

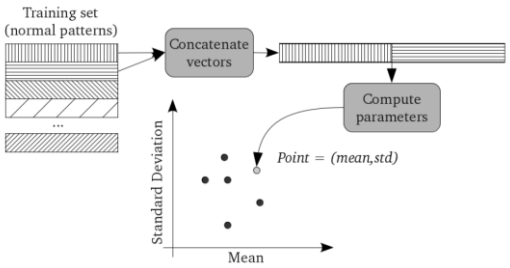

Fig. 3. Steps of computing points in the normal distribution parameter space. [4]

In the Conv.Hull-PS method, one convex hull is obtained from the set of points where each element is calculated from normal data pairs. Another convex hull is obtained from the set of points where each element is calculated from a newly observed datum and a known normal datum. The area of the first convex hull is denoted as H_N and the second area of H_U. The area of intersection between H_N and H_U is denoted as I.

A datum is classified as anomalous if it is above a certain threshold d calculated based on the values H_N, H_U, and I as follows

$$d = (H_N - I) + (H_U - I) \quad (7)$$

This value d can be obtained by applying learning algorithms over a training set, cross-validation set, and test set while trying to find the value d that produces the best accuracy.

III. Implementation

In this section, a demonstration of the convex-hull method for anomaly detection will be conducted. The focus of this section is to demonstrate the utilization of convex-hull method, particularly that which utilizes the Quickhull approach for anomaly detection. The learning algorithm workflow is adapted accordingly. The program is written in Python.

A. Dataset

The dataset used in this paper is provided by [5] and contains medical and lifestyle information for 1500 patients, structured to predict the presence of cancer based on various features. The dataset has been preprocessed and cleaned. Below is the description for each feature in the dataset

TABLE I. Dataset Feature Description

Feature	Type	Range/Values	Description
Age	Integer	20 to 80	Represents the patient's age.
Gender	Binary	0: Male 1: Female	Represents the gender of the patient.
BMI	Continuous	15 to 40	Represents the Body Mass Index of the patient.
Smoking	Binary	0: No 1: Yes	Indicates the smoking status of the patient.
GeneticRisk	Categorical	0: Low 1: Medium 2: High	Represents the genetic risk level for cancer.
PhysicalActivity	Continuous	0 to 10	Represents the number of hours per week spent on physical activities.
AlcoholIntake	Continuous	0 to 5	Represents the number of alcohol units consumed per week.
CancerHistory	Binary	0: No 1: Yes	Indicates whether the patient has a personal history of cancer.
Diagnosis	Binary	0: No Cancer 1: Cancer	The target variable indicating the cancer diagnosis status.

B. Convex Hull Algorithm

The convex hull algorithm used is the Quicksort algorithm as described in the Fundamental Theory section with several modifications to improve the time and space complexity of the code. Below are the steps for the Quickhull algorithm

1. Check the number of points. If it is less than three then return the points.
2. Sort the points and find the first and last point from the sorted set of points.
3. Divide the set into points to the left and to the right of the line whose endpoints are the first and last points.
4. Find the hull of the two sets using the find_hull function. The find_hull function utilizes the distance function. This distance function will compute the cross products from vector formed by the three points (a point, and two other points that form the line) then return negative if it is to the right of the line and positive if it is to the left of the line.
5. The find_hull function is called recursively.
6. The Quickhull function will return an array containing the first point, the result from the find_hull function for the left set, the the last point, and the result from the find_hull function for the right set of points.

The optimization is made in the 4th step, allowing the algorithm to compute the distance only once and using the sign of the distance to determine the orientation of the point to the line.

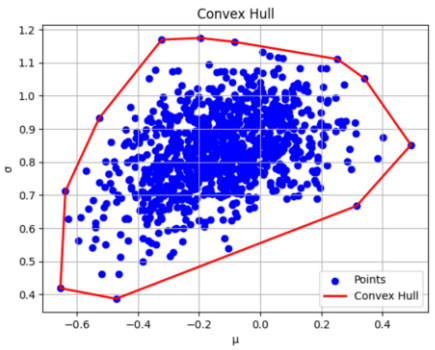

Fig. 4. Convex hull of a set of points

C. Data Preparation

Since the dataset is preprocessed, minor preparation is done by the author. A normalization process is done for each feature of the dataset using the z-score normalization method. The data is divided into two data frames, namely the diagnosis_0_data (no cancer) and anomaly (cancer).

D. Finding The Suitable Threshold

In order to find the suitable threshold of d, the model is being trained by the following steps:

1. Classify the data according to the value of the 'Diagnosis' attribute.
2. Set a value d_{init}.
3. 85% of the data from the no cancer data is used as a training set. The remaining 15% is used in a test set with 50% of the data from the cancer data.
4. Compute the convex hull H_N of the training set.
5. For each datum from the test set, pair it with a point from the training set. Compute the convex hull H_U of this set of pairs.
6. Compute the value d from the resulting set. If d is greater than d_{init}, the predicted value is 1, otherwise it is 0.
7. Compute the accuracy of model with the value d_{init}.
8. Repeat from step 2 for each iteration.
9. Select the value d_{init} that gives the best accuracy.

The following is the pseudocode for each function involved in the process

```
function calculate_parameter_space_points(pairs, dataset):
  # Initialize an empty array
  concatenated_array = create_empty_array(length(pairs), 2 * number_of_columns_in_dataset)

  # Iterate through each pair of indices
  for idx, pair in enumerate(pairs):
    # Concatenate the corresponding rows from the dataset
    concatenated_values = concatenate_rows(dataset.loc[pair[0]], dataset.loc[pair[1]])

    # Assign the concatenated values to the array
    concatenated_array[idx] = concatenated_values

  # Calculate the means and standard deviations for each row of the concatenated array
  means = calculate_means(concatenated_array)
  stds = calculate_stds(concatenated_array)

  # Create the parameter space points by stacking means and standard deviations horizontally
  parameter_space_points = stack_horizontally(means, stds)

  return parameter_space_points
```

```
function generate_output_convex_hull(original_df, new_element):
  new_row_df = pd.DataFrame([new_element], columns=original_df.columns)
  dataset = original_df._append(new_row_df, ignore_index=True)

  original_indexes = np.arange(len(original_df))
  new_index = len(original_df)

  pairs = np.column_stack((original_indexes, np.full_like(original_indexes, new_index)))

  output_parameter_space_points = calculate_parameter_space_points(pairs, dataset)
  return quickhull(output_parameter_space_points)
```

```
function plot_convex_hulls(convex_hull_points_normal, convex_hull_points_anomaly, data_normal=None, data_anomaly=None):
  """
  Plots the convex hulls of normal and anomaly points on a 2D parameter space.

  Parameters:
  - convex_hull_points_normal: numpy array of points forming the convex hull for normal data
  - convex_hull_points_anomaly: numpy array of points forming the convex hull for anomaly data
  - data_normal: (optional) original normal data points to plot
  - data_anomaly: (optional) original anomaly data points to plot
  """
  plt.figure(figsize=(5, 4))

  # Plot original normal data points
  if data_normal is not None:
    plt.scatter(data_normal[:, 0], data_normal[:, 1], color='green', label='Normal Data')

  # Plot original anomaly data points
```

```
  if data_anomaly is not None:
    plt.scatter(data_anomaly[:, 0], data_anomaly[:, 1], color='red', label='Anomaly Data')

  # Plot convex hull for normal data
  hull_normal = ConvexHull(convex_hull_points_normal)
  for simplex in hull_normal.simplices:
    plt.plot(convex_hull_points_normal[simplex, 0], convex_hull_points_normal[simplex, 1], 'green', lw=2, label='Normal Hull' if simplex[0] == 0 else "")

  # Plot convex hull for anomaly data
  hull_anomaly = ConvexHull(convex_hull_points_anomaly)
  for simplex in hull_anomaly.simplices:
    plt.plot(convex_hull_points_anomaly[simplex, 0], convex_hull_points_anomaly[simplex, 1], 'blue', lw=2, label='Anomaly Hull' if simplex[0] == 0 else "")

  plt.xlabel('μ')
  plt.ylabel('σ')
  plt.legend()
  plt.title('Convex Hulls for Normal and Anomaly Data')
  plt.show()
```

```
function intersection_area(hull1, hull2):
  """
  Calculate the intersection area of two convex hulls.

  Parameters:
  hull1 (list of list of floats): Coordinates of the first convex hull.
  hull2 (list of list of floats): Coordinates of the second convex hull.

  Returns:
  float: The area of the intersection of the two convex hulls.
  """
  # Create polygons from the convex hull coordinates
  polygon1 = Polygon(hull1)
  polygon2 = Polygon(hull2)

  # Calculate the intersection of the two polygons
  intersection = polygon1.intersection(polygon2)

  # Return the area of the intersection
  return intersection.area
```

```
function param_d(hull1, hull2):
  """
  Calculate the parameter d for two convex hulls.

  Parameters:
  hull1 (list of list of floats): Coordinates of the first convex hull.
  hull2 (list of list of floats): Coordinates of the second convex hull.

  Returns:
  float: The parameter d.
  """
  polygon1 = Polygon(hull1)
  polygon2 = Polygon(hull2)

  # Calculate the intersection of the two polygons
  intersection = polygon1.intersection(polygon2)

  return polygon1.area + polygon2.area - 2 * intersection.area
```

```
list_of_d_values = []
list_of_accuracy_score = []

normal_size = len(diagnosis_0_data)
anomaly_size = len(anomaly)

d = 0.25
num_of_iteration = 10
increment = (0.65 - d) / num_of_iteration

for i in range(10):
    print("Iteration -", str(i+1))

    d += increment
    list_of_d_values.append(d)

    normal_train_size = int(0.85 * normal_size)
    normal_train_indices = np.random.choice(normal_size, normal_train_size, replace=False)
```

```
    normal_train_set =
diagnosis_0_data.iloc[normal_train_indices]


    # Create normal test set from indices not in
training set
    normal_test_set =
diagnosis_0_data.drop(normal_train_set.index)


    # Randomly select indices for anomaly test set
    anomaly_test_indices =
np.random.choice(anomaly_size, int(0.5 *
anomaly_size), replace=False)
    anomaly_test_set =
anomaly.iloc[anomaly_test_indices]


    # Combine normal and anomaly test sets
    test_set = pd.concat([normal_test_set,
anomaly_test_set])


    true_labels = np.zeros(len(test_set))
    true_labels[len(normal_test_set):] = 1  # Set
labels for anomaly instances


    selected_train_pairs =
np.array(list(combinations(normal_train_set.index,
2)))


    selected_train_pairs_indices =
np.random.choice(len(selected_train_pairs), 500,
replace=False)
    selected_train_pairs =
selected_train_pairs[selected_train_pairs_indices]
    train_parameter_space_points =
calculate_parameter_space_points(selected_train_pa
irs, normal_train_set)
    train_convex_hull =
quickhull(train_parameter_space_points)


    predictions = np.zeros(len(test_set))


    for idx in range(len(test_set)):
        test_parameter_space_points =
generate_output_convex_hull(normal_train_set,
test_set.iloc[idx])
        test_convex_hull =
quickhull(test_parameter_space_points)
        predictions[idx] = 1 if
param_d(train_convex_hull, test_convex_hull) > d
else 0


    accuracy = accuracy_score(true_labels,
predictions)
    print("Accuracy:", accuracy)
    print()
```

```
    list_of_accuracy_score.append(accuracy)
```

IV. Results

From a random sampling of 10 data, it is observed that the value of parameter d ranges from around 0.25 to 0.65. The training is conducted based on this sampling. Ten iterations are done as explained in the previous section is conducted. The results are as follows

TABLE II. Accuracy Results for Each Iteration

Iteration	d-value	Accuracy
1	0.29	0.8166666666666667
2	0.33	0.7976190476190477
3	0.37	0.7047619047619048
4	0.41	0.7095238095238096
5	0.45	0.5452380952380952
6	0.49	0.6333333333333333
7	0.53	0.3904761904761905
8	0.57	0.4595238095238095
9	0.61	0.3476190476190476
10	0.65	0.3476190476190476

Therefore, the model performs best with a d-value of 0.29, giving an accuracy of 81.67% in the test set of the data.

V. Conclustion

The Conv.Hull-PS anomaly detection method has proven itself to be capable of detecting anomalous features in possible cancer patients, despite being tested in a simplified machine learning workflow. With this algorithm, the underlying distribution of each feature is unimportant, as all are represented with a point in the normal distribution parameter space.

Overall, the findings suggest that the Conv.Hull-PS algorithm can serve as a valuable tool for early cancer detection, potentially leading to improved outcomes for patients through early intervention and treatment. However, further research and validation are necessary to confirm the effectiveness and generalizability of this approach across different datasets and populations.

Appendix

The code implementation used in this paper can be seen and retrieved on my Kaggle notebook:

https://www.kaggle.com/code/julianchandrasutadi/anomaly-detection-using-convex-hull-method

Acknowledgment

The author would like to extend heartfelt appreciation to Dr. Nur Ulfa Maulidevi, S.T, M.Sc. who served as the author's instructor this past semester. Additionally, the author wants to express gratitude to all IF2211 Algorithm Strategies lecturers, Dr. Ir. Rinaldi Munir, S.T., M.T. and Ir. Rila Mandala, M.Eng., Ph.D., who have provided the academic resources necessary for completing this paper.

REFERENCES

[1] J. Barnard and C. Stryker. (2023). "Anomaly Detection". [Online]: https://www.ibm.com/topics/anomaly-detection. Accessed June 12th

[2] Munir, Rinaldi. (2024). "Algoritma Divide and Conquer (Bagian 4)". [Online]: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-(2024)-Bagian4.pdf. Accessed June 12th

[3] J. Orloff and J. Bloom. (2022). "Introduction to Probability and Statistic". [Online]: https://ocw.mit.edu/courses/18-05-introduction-to-probability-and-statistics-spring-2022/mit18_05_s22_probability.pdf. Accessed June 12th

[4] G.B.P, Costa et al., (2013). "Partially supervised anomaly detection using convex hulls on a 2D parameter space". [Online]: https://gbpcosta.github.io/assets/publications/Costa_PSL2013.pdf. Accessed June 12th

[5] R.E. Kharuoa, (2024), "Cancer Prediction Dataset", https://www.kaggle.com/datasets/rabieelkharoua/cancer-prediction-dataset

DECLARATION OF ORIGINALITY

I, the undersigned below, the Author of this paper, hereby declare that this paper is my own writing, not an adaptation or translation of someone else's paper, and not plagiarized

Bandung, 12 Juni 2024

Julian Chandra Sutadi 13522080