APPLICATION OF GEOMETRIC GRAPH DISTANCES IN MACHINE LEARNING CLASSIFICATION

William Rodman

Oral Defense

Newcomb-Tulane Collage Honors Thesis

April 18th, 2024

ORAL DEFENSE AGENDA

50-minute presentation:

1. Academic background and previous research.
2. Research topics and hypothesis.
3. Research conducted.
4. Results and hypothesis test.
5. Continuing work.

30-minute deliberation:

1. Questions and comments.
2. Advisors satisfactory/unsatisfactory decision.

ACADEMIC BACKGROUND AND PREVIOUS RESEARCH

2020 - 2024

B.S. MATHEMATICS AND COMPUTER SCIENCE

- Statistical Coursework: Linear Algebra, Probability Theory, Statistics, Stochastics and Linear Models.
- Machine Learning Coursework: Machine Learning and Data Science.

2021 - 2023

COMPUTER SCIENCE RESEARCH ASSISTANCE

- Begun working on geometric distances in May of 2021 under Dr. Carola Wenk.
- Contributed to two geometric distance Python Packages on GitHub.

2023 - 2024

HONORS THESIS THAT COMBINES GEOMETRIC DISTANCE ALGORITHMS AND STATISTICAL MACHINE LEARNING

TOPICS WEAK FRÉCHET AND TRAVERSAL DISTANCE

Research Topics

WEAK FRÉCHET DISTANCE

- Polygonal curves.
- Distance measurements between two polynomial curves.
- Subproblem of the traversal distance.
- Free-space diagram visualization.

TRAVERSAL DISTANCE

- Geometric graphs.
- Distance measurements distance between two geometric graphs.
- Symmetric case of the traversal distance.
- Visualizing traversal distance free space area.

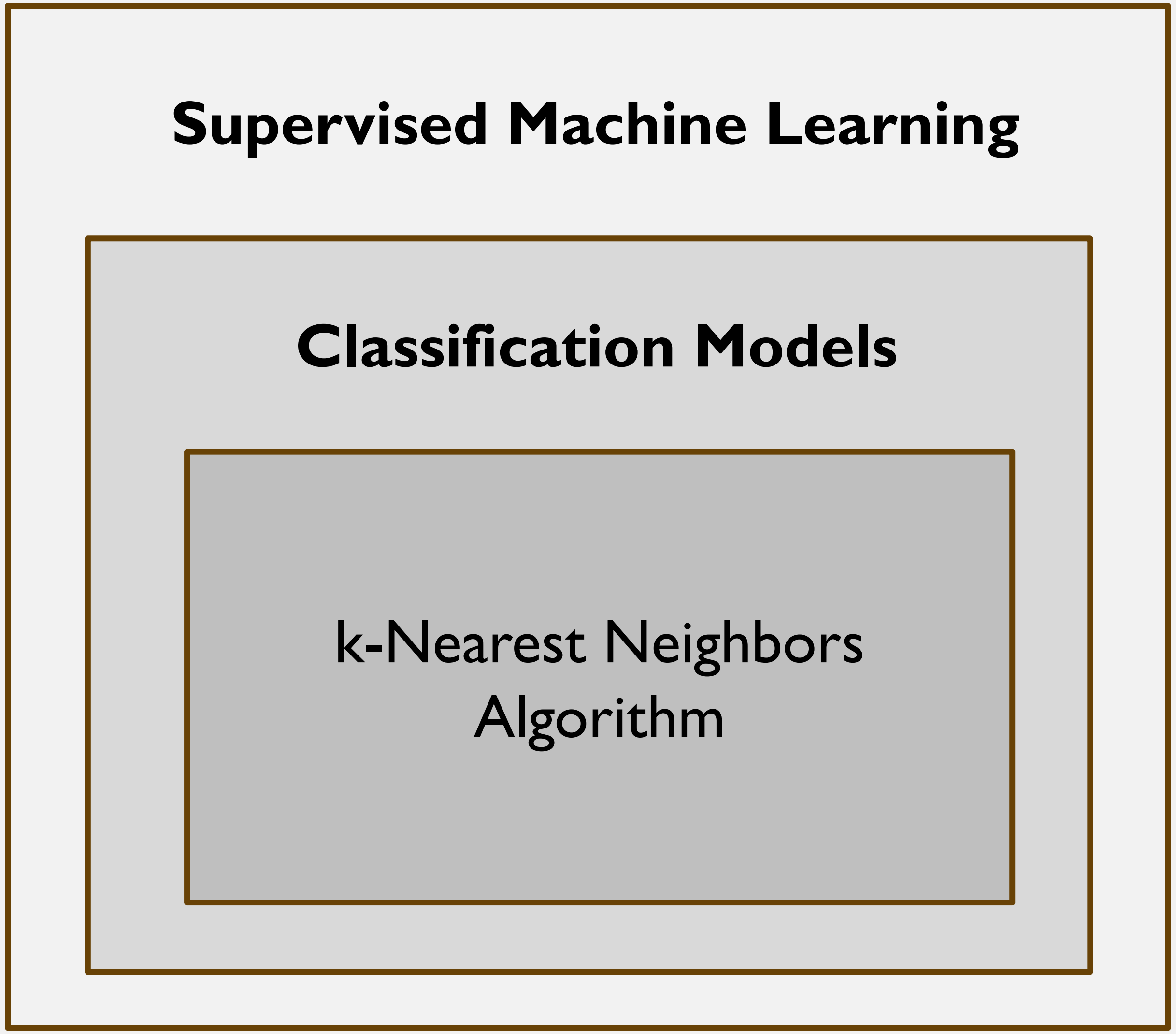
Comparable Distance

GRAPH EDIT DISTANCE

- Edit distances between two geometric graphs.

TOPICS CLASSIFICATION PROBLEMS AND K-NN

- Datasets of labeled geometric graphs:
  - English letter dataset.
  - Plant leaf dataset.
- Distance based algorithms in machine learning:
  - k-Nearest Neighbors algorithm.
- Evaluating multiclassification problems:
  - Macro average precision.
  - Macro average recall.



HYPOTHESIS

THE K-NEAREST NEIGHBORS MODEL, WHEN APPLYING TRAVERSAL DISTANCE AS ITS DISTANCE METRIC, IS:

1. PRECISE IN CLASSIFYING GEOMETRIC GRAPHS.
2. COMPUTATIONALLY EFFICIENT.

RESEARCH GEOMETRIC GRAPHS

DEFINITION

- G is a pair G = (V, E)
- V is the set of vertices (with vertex v in V) corresponding to a pair of two-dimensional coordinates (x, y).

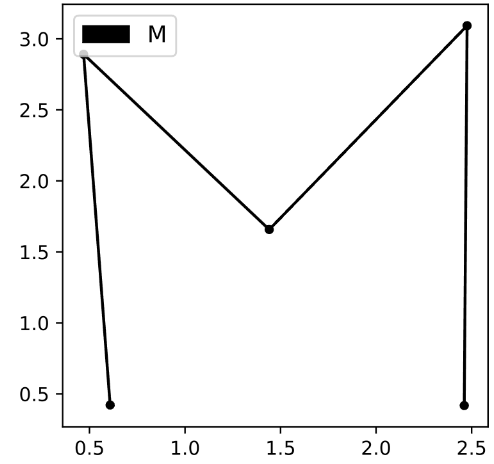
$$V = \{1, 2, \ldots, n\} \quad \text{where} \quad n \in \mathbb{N} \quad \text{such that} \quad v_i \mapsto (x_i, y_i)$$

- E is the set of edges (with edge e in E) that defines a line segment bound by two vertices.

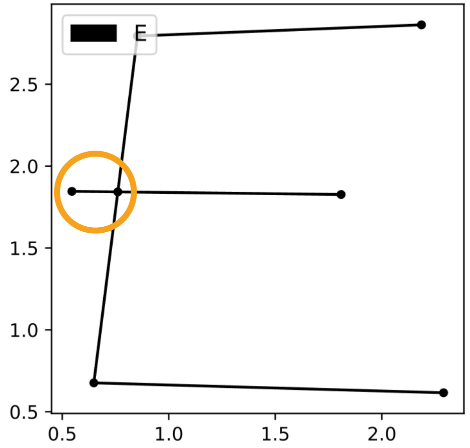
$$E = \{e_i | (v_i, v_j) \in V \times V\} \quad \text{such that} \quad e_i = (v_i, v_j)$$

Note: Edges have a geometric property.

Geometric graph (G) and polygonal curve (C).



Strictly geometric graph (G).



RESEARCH WEAK FRÉCHET DISTANCE

DEFINITION

- For two curves, C1 and C2, this is the minimum parameter (ϵ) that allows an agent to increment by segment through their parameter space.

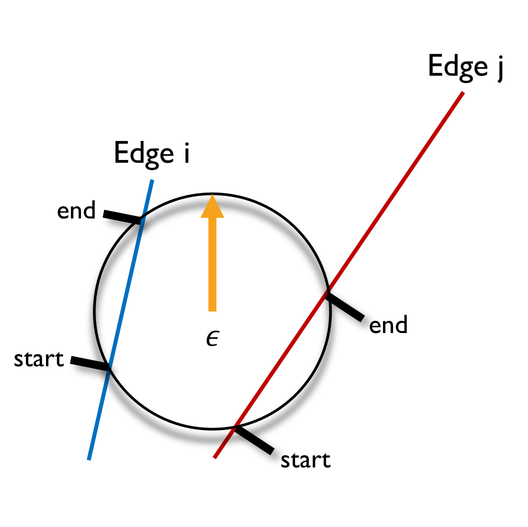
$$\delta_F(C_1, C_2) := \inf_{\alpha \to [0,1], \beta \to [0,1]} \max_{t \in [0,1]} \|C_1(\alpha(t)) - C_2(\beta(t))\|$$

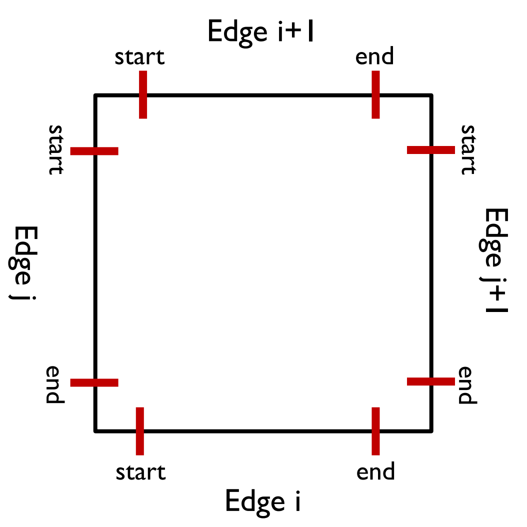
C1start is a(1).
C1 end is b(0).

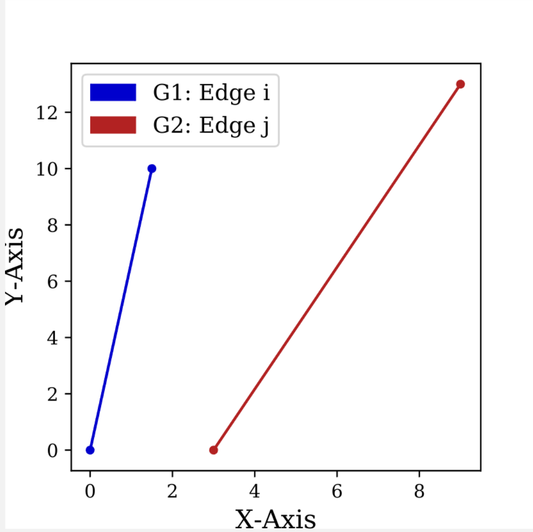
C2 start is b(0).
C2 end is a(1).

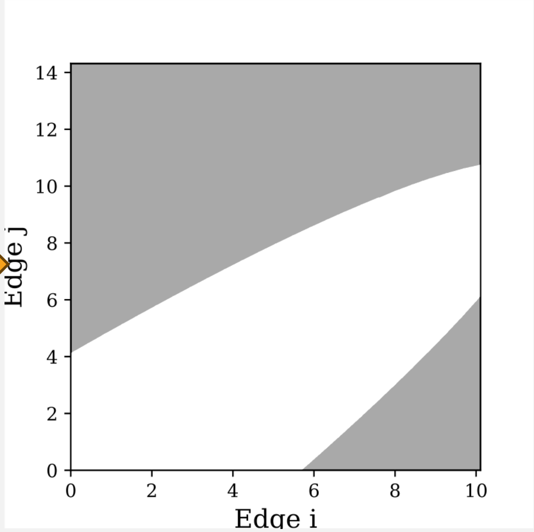
- When computing the Weak Fréchet distance algorithm, the decision problem considers whether the distance is at most ϵ.

$$\delta_F(G_1, G_2) \leq \epsilon$$





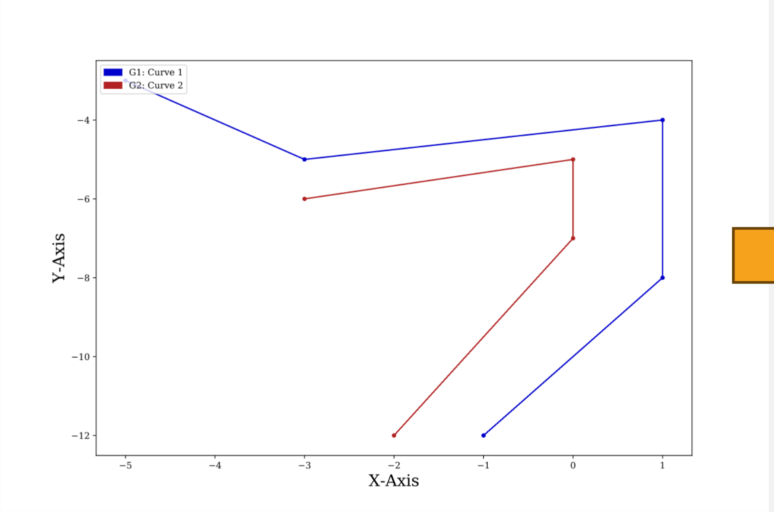




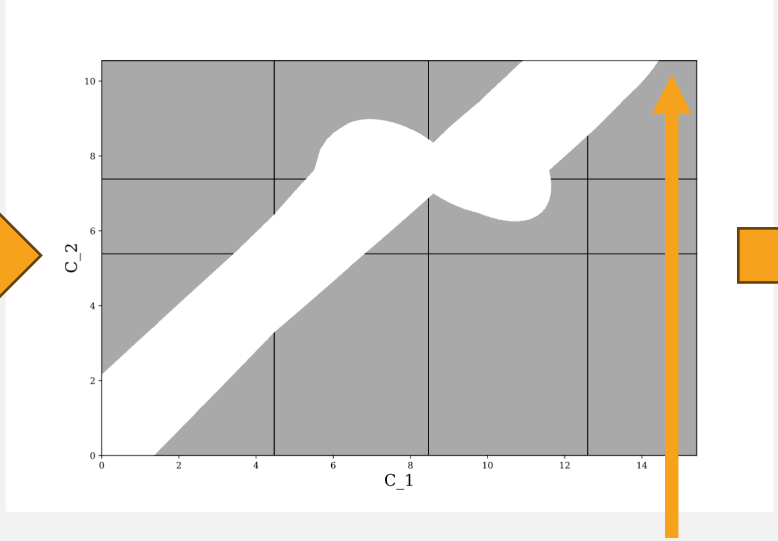
RESEARCH FREE-SPACE DIAGRAM

- Represents the parameter space of all possible e and v combinations for C1 and C2, where ϵ is an arbitrary threshold within the parameter space.
- The weak Fréchet Distance algorithm stores free-space as a dictionary of discrete (start, end) cell boundaries.

Two Polygonal Curves

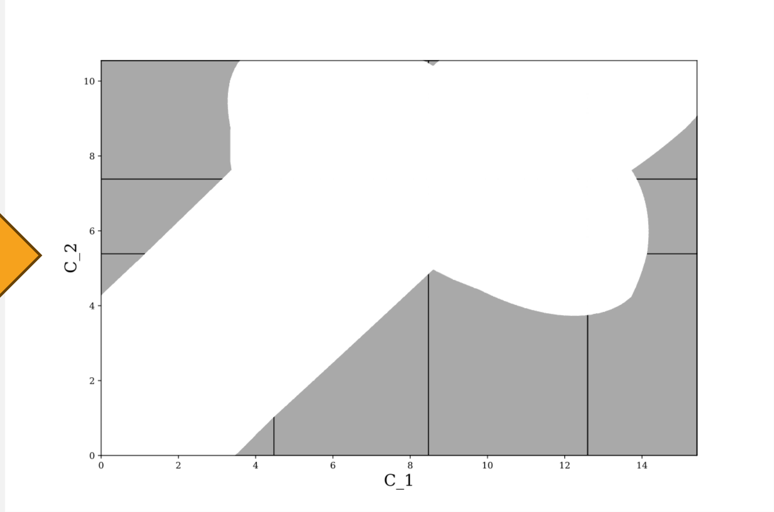


$\delta_F(C_1, C_2) > 2$



Free-space does not cover cell (4, 3) entirely!

$\delta_F(C_1, C_2) \leq 4$



C1 consists of 4 segments.
C2 consists of 3 segments.

4 x 3 = 12 total free-space cells.
4 of 12 free-space cells are empty.

Zero free-space cells are empty.
Free-space covers both curves entirely.

RESEARCH TRAVERSAL DISTANCE

DEFINITION

- For two geometric graphs G1 = (V1, E1) and G2 = (V2, E2). The traversal distance from G2 to G1 is:

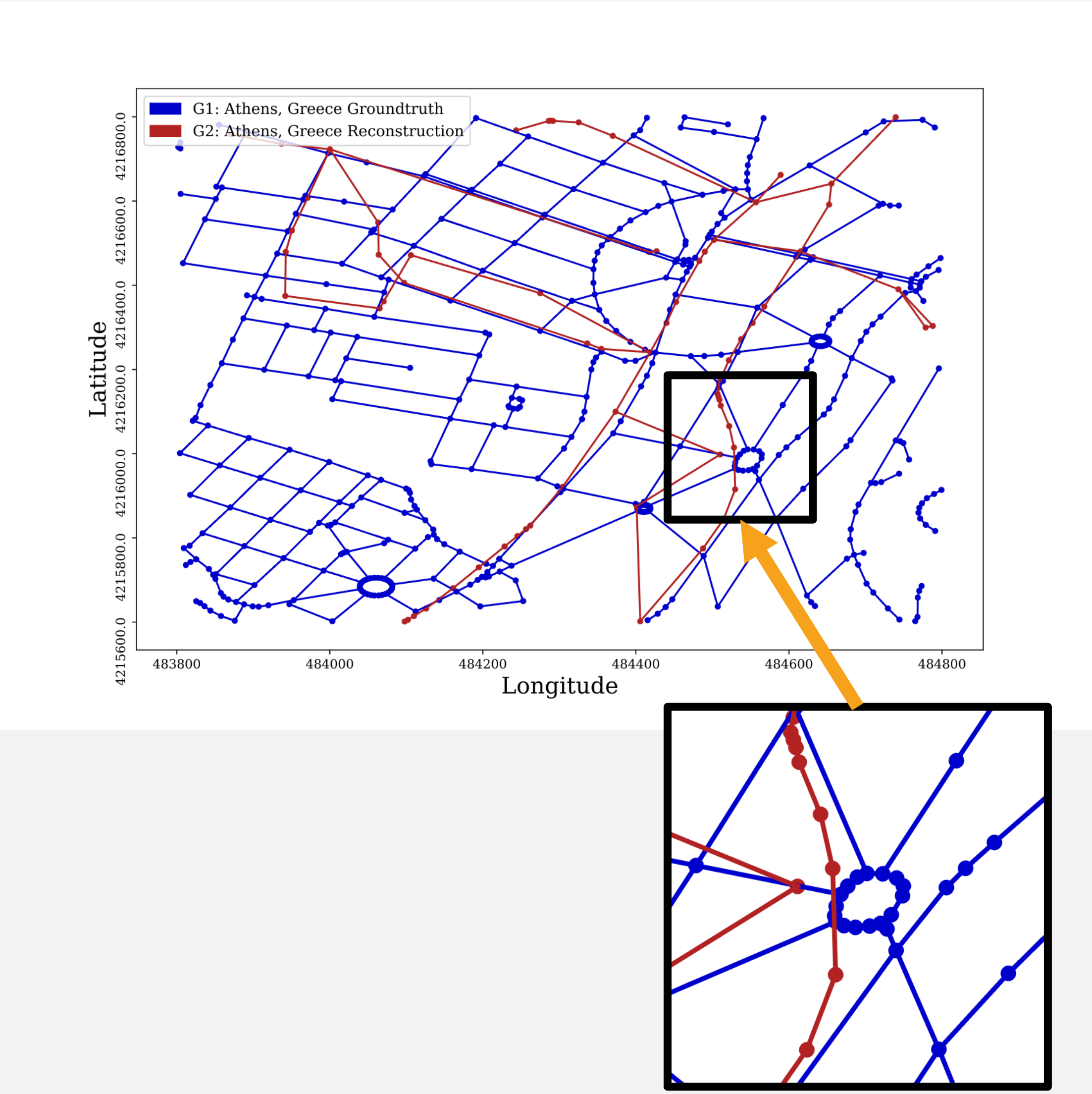
$$\delta_T(G_1, G_2) = \inf_{f,g} \max_{t \in [0,1]} \|f(t) - g(t)\|$$

Continuous parametrization entirely over G2.

Continuous parametrization partially over G1.

- Similar decision problem considers whether the distance is at most ϵ.

$$\delta_T(G_1, G_2) \leq \epsilon$$



RESEARCH SYMMETRIC TRAVERSAL DISTANCE

PROPERTIES

- The entirety of G2 must be traversed to satisfy the traversal distance definition, whereas G1 does not require complete traversal.
- The traversal distance is asymmetric; specifically, the distance from G1 to G2 may not be equal to the distance from G2 to G1.

DEFINITION

- For G1 and G2, the symmetric traversal distance between the two geometric graphs is determined by the maximum epsilon value from both combinations of traversal distances such that:

$$\delta_{ST}(G_1, G_2) = \max\{\delta_T(G_1, G_2), \delta_T(G_2, G_1)\}$$

RESEARCH TRAVERSAL DISTANCE ALGORITHM

GRAPH DATA STRUCTURE

- nodes dictionary contains coordinate points of vertices. $nodes = \{1 : (x_1, y_1), 2 : (x_2, y_2), \ldots n : (x_n, y_n)\}$
- nodeLinks dictionary contains adjacency lists of edges. $nodeLinks = \{1 : \{j \in e_1\}, 2 : \{j \in e_2\}, \ldots n : \{j \in e_n\}\}$

j vertices neighboring v1.

FREE-SPACE DATA STRUCTURE

- cellBoundary contains the starting and ending points of free-space for the wall of a cell. $cellBoundary(e_i, v_j) = (start, end) \quad \text{where} \quad 0 \leq start < end \leq 1$
- Cell_Boundries dictionary contains combinations of cellBoundary. $cell\_boundaries = \{(e, v) : cellBoundary(e, v)) \mid e \in E_1, E_2 \quad v \in V_1, V_2\}$

RESEARCH TRAVERSAL DISTANCE ALGORITHM

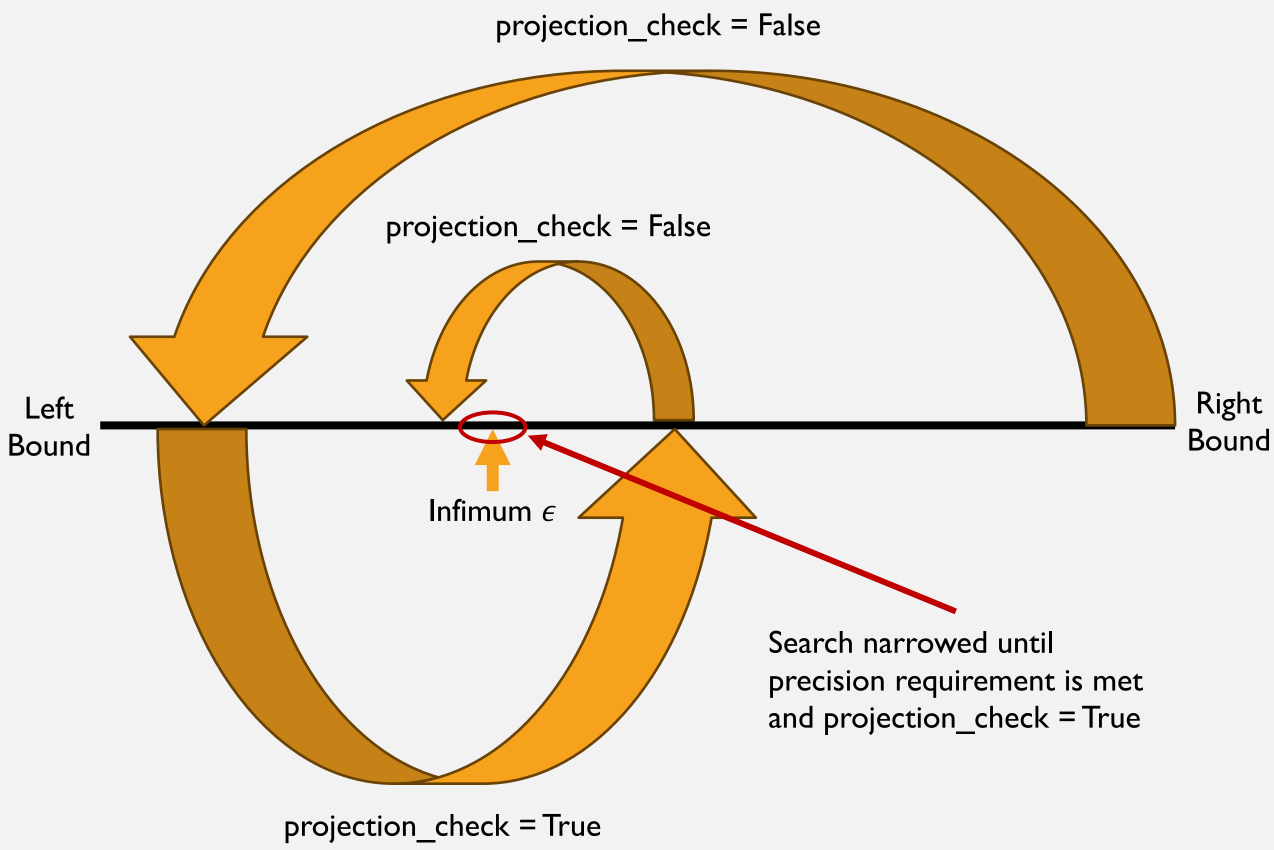
PROCEDURE

- **Step 1:** Compute the Cell Boundaries.
  - DFSTraversalDist searches for and computes all free-space cellBoundary's using a Depth-First Search algorithm.
  - Writes each cellBoundry to cell_boundaries.
- **Step 2:** Verify the Traversal of G2.
  - projection_check determines if the projection of cell_boundaries onto G2 covers the the graph entirety.

$$projection\_check = \begin{cases} \text{True} & \delta_T(G_1, G_2) \leq \epsilon \\ \text{False} & \delta_T(G_1, G_2) > \epsilon \end{cases}$$

- **Step 3:** Binary Search for Traversal Distance.
  - binarySearch approximates the infimum of the traversal distance equation.
  - Incorporates DFSTraversalDist and projection_check.

BINARY SEARCH ILLUSTRATION



RESEARCH TRAVERSAL DISTANCE ALGORITHM

TIME COMPLEXITIES

- All three algorithmics steps of the traversal distance run in polynomial time.
- binarySearch algorithm searches continuous epsilon by implementing discretization.
- Time complexity of the traversal distance is a product of DFSTraversalDistance, projection and binarySearch.

Algorithm	Worst-Case Time Complexity
DFSTraversalDistance	$O((\|V_1\| + \|E_1\|) \times (\|V_2\| + \|E_2\|))$
projection_check	$O((\|V_1\| \times \|E_2\|) + (\|V_2\| \times \|E_1\|))$
binarySearch	$O(\log_2 \Delta) \quad \text{where} \quad \Delta = \frac{right - left}{precision}$
Traversal Distance	$O(\log_2 \Delta \times (\|V_1\|\|E_2\| + \|V_2\|\|E_1\| + \|V_1\|\|E_1\| + \|V_2\|\|E_2\|))$

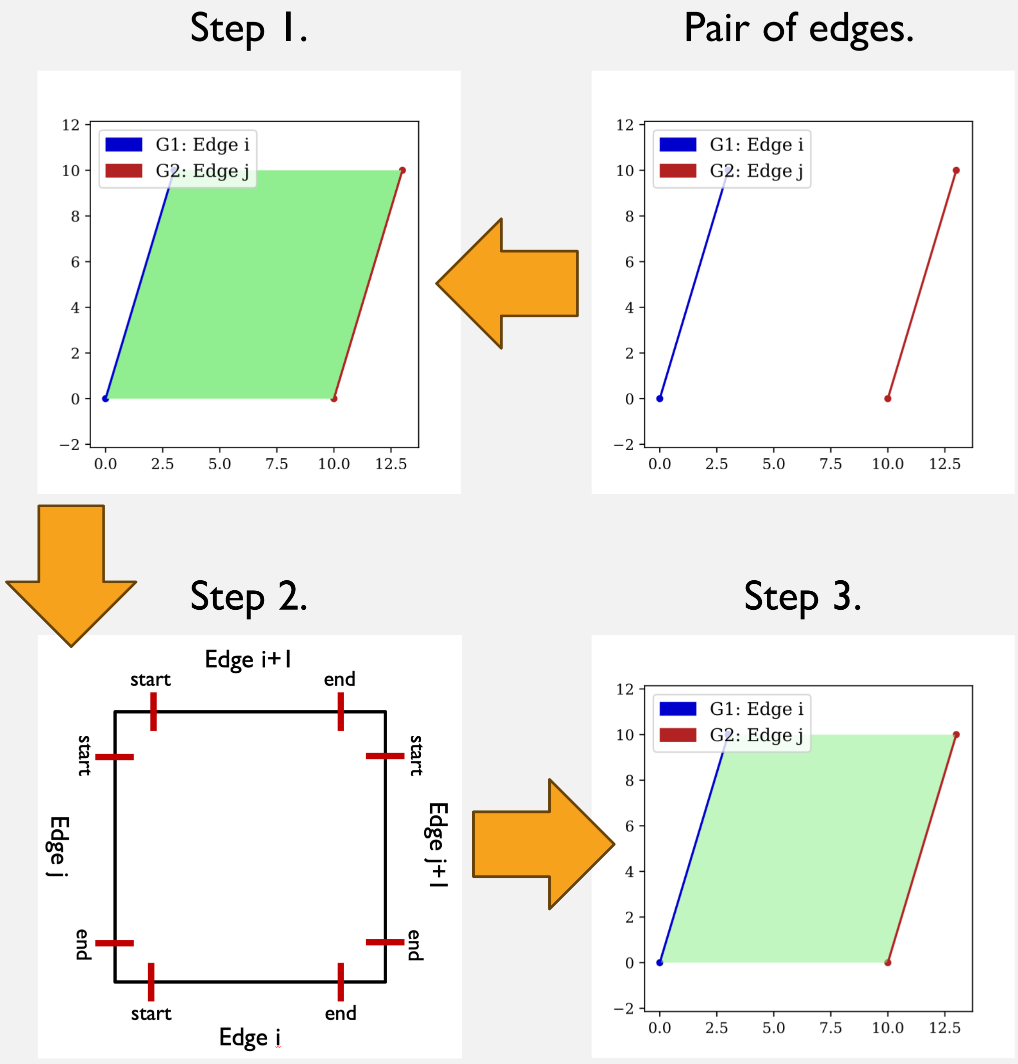
Traversal Distance = binarySearch x (DFSTraversalDistance + projection_check)

RESEARCH TRAVERSAL DISTANCE VISUALIZATION

Challenge: Visualizing traversal distance free-space in 2D free-space diagram is infeasible.

VISUALIZING FREE-SPACE AREA STEPS

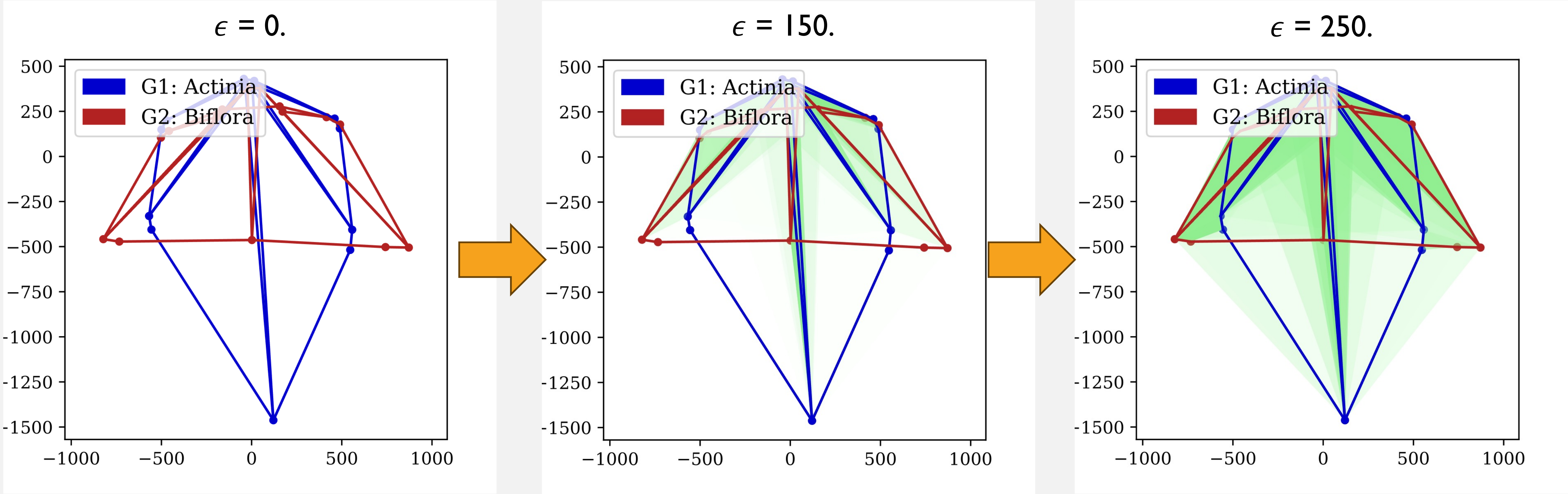
- **Step 1:** Apply quadrilateral colored area between the line segments of two edges; for all combinations of edges.
- **Step 2:** Compute the percentage of area within a free-space cell covered by free-space.
    - Empty free-space cell: 0%
    - Full free-space cell: 100%
- **Step 3:** Apply percentage as a transparency to the quadrilateral-colored area.



RESEARCH TRAVERSAL DISTANCE VISUALIZATION

PLANT LEAF EXAMPLE

The percentage of transparency decreases for overlapping areas, transparency decreases for quadrilateral-colored areas as ϵ increases in the following example:

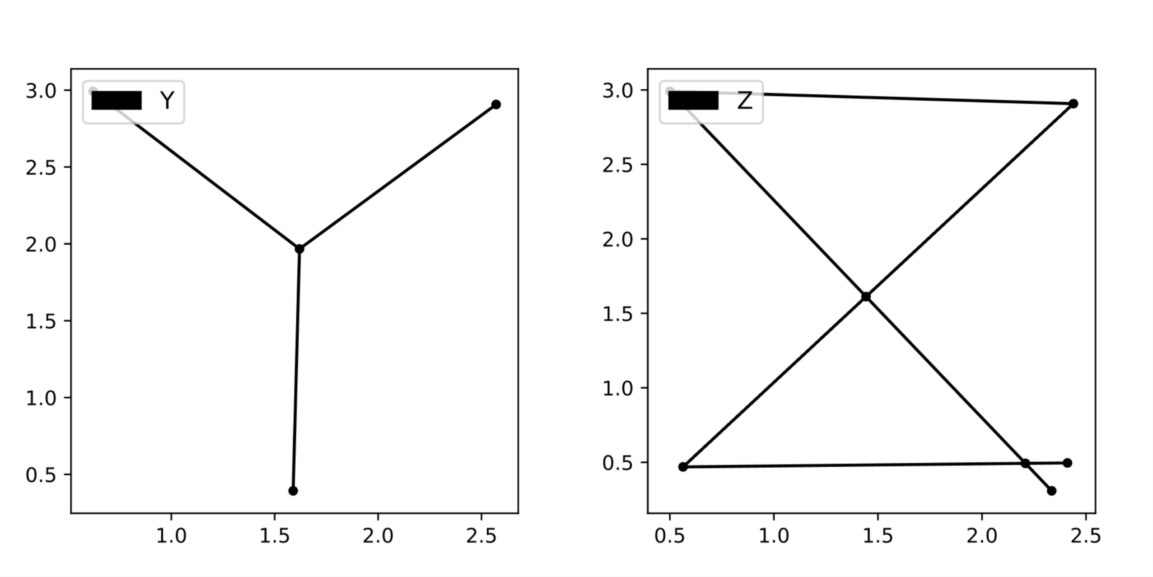


TEST ENGLISH LETTER DATASET

METADATA

- Distorted drawings of English letters with no curvature.
- Contains 2,250 labeled geometric graphs.
- Categorized into 15 distinct classes of 150 observations.
  - A, E, F, H, I, K, L, M, N, T, V, W, X, Y and Z.

DATASET SAMPLES



- Observation “Y” is visibly recognizable.
- Observation “Z” is visibly unrecognizable.

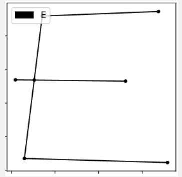
TEST K-NEAREST NEIGHBORS ALGORITHM

DESCRIPTION

- Named KNeighborsClassifier.
- Python 3 script.
- k-Nearest Neighbors distance metrics are required to be symmetric.
- Developed custom k-Nearest Neighbors algorithm that implements the symmetric traversal distance.

EXAMPLE PREDICTION

k = 3
Training observations: 6
Target observation:



Compute distances:

Distance	0.29	0.98	0.07	1.12	0.66	0.87
Label	E	N	E	A	L	E

Sort distances:

Distance	0.07	0.29	0.66	0.87	0.98	1.12
Label	E	E	L	E	N	A

k nearest distances:

Distance	0.07	0.29	0.66
Label	E	E	L

Sum label instances:

Label	E	L
Count	2	1

Predicted class is most common instance.

TEST MACHINE LEARNING PIPELINE

How the custom KNeighborsClassifier algorithm was tested on the English letter dataset:

Model

Symmetric Traversal Distance Algorithm

Distance Metric.

k-Nearest Neighbors Algorithm

Testing

- Training and testing dataset split uniformly random.
  - Training observations: 1794
  - Testing observations : 449
- binarySearch parameters:
  - Search space: (0, 3)
  - Precision: 0.001
  - Worst-case scenario: 12 steps.

Evaluation

- Multi classification performance metrics:
  - Macro-average precision.
  - Macro-average recall.
- Measured precision and recall scores for k values ranging from 0 to 40.

Code Snippet:

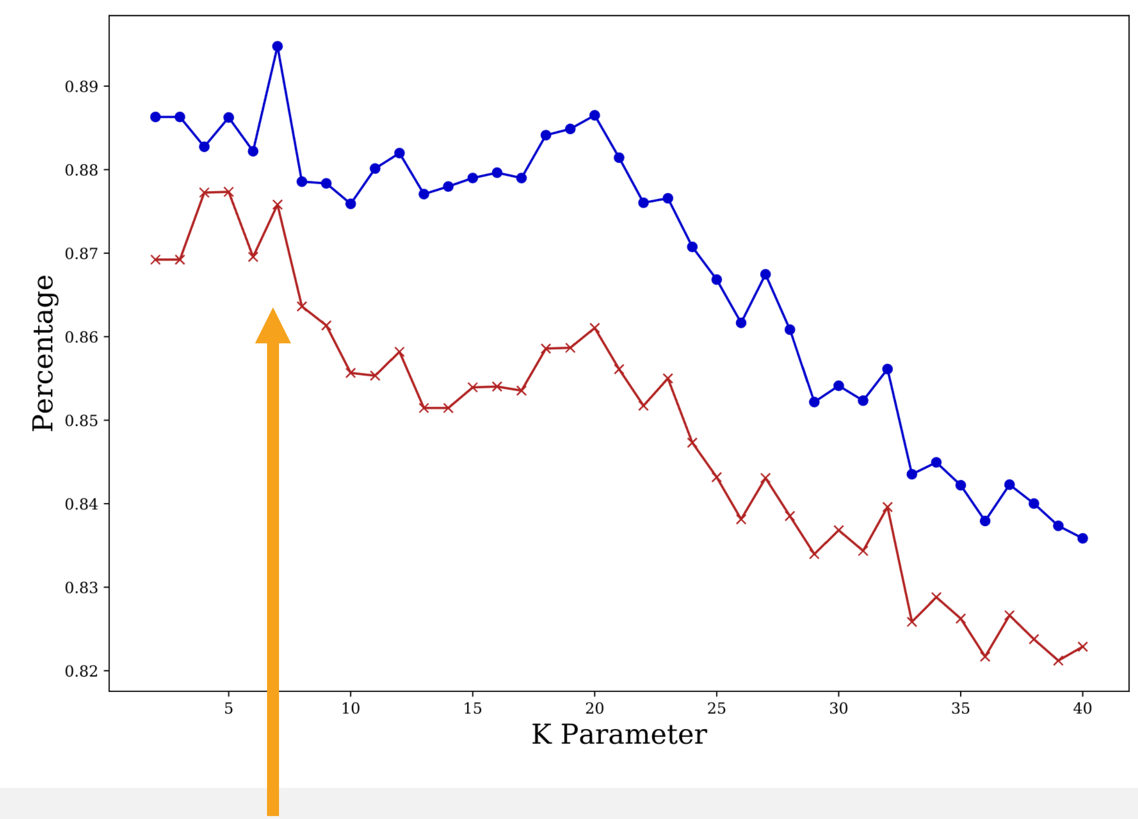
```
model = KNeighborsClassifier(n_neighbors=40, mean='max', left=0, right=3, precision=0.001)
model.fit(X_train, y_train)
```

TEST RESULTS

EVALUATION RESULTS

- Model runtime: ~7 hours.
- Achieved highest macro-average precision when k = 7.
- Macro-average precision: 85.9%
- Macro-average Recall: 87.6%

Macro-Average Precision and Recall as Value of k Increases.



Macro-average precision peaks when k = 7.

TEST REVISITING HYPOTHESIS

TRAVERSAL DISTANCE AND GRAPH EDIT DISTANCE COMPARISON

Algorithm	Score (%)	Metric	Observations	Complexity
KNeighborsClassifier	89.5	Macro-Average Precision	2,250	Polynomial
Graph Edit Distance k-Nearest Neighbors	99.6	Undefined Precision*	6,750	NP-Hard

Is the KNeighborsClassifier algorithm computationally efficient? **Yes**

Is the KNeighborsClassifier algorithm precise in classification? **No****

* The multi classification metric needs to be confirmed.

** The KNeighborsClassifier needs to be tested on entire dataset.

CONTINUING WORK

DEADLINES

- Oral Defense Completion: April 19th
- Final Potential Research Meeting: May 1st
- Thesis Paper Submission: May 3rd

PENDING TOPIC REVISIONS

- Free-space description.
- Free-space diagram illustration.
- projection_check algorithm definition.
- KNeighborsClassifier comparison test.

QUESTIONS AND COMMENTS