ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25may1735

Topological Machine Learning: Theoretical Foundations and a Custom Classifier on Artificial Data

Dr. Mitat Uysal[1]

[1] Software Engineer, Dept.-Dogus University

Publication Date: 2025/06/4

Abstract: Topological Machine Learning (TML) leverages tools from algebraic topology, especially persistent homology, to extract robust features from data that remain invariant under continuous deformations. This article presents a comprehensive overview of the theoretical underpinnings of TML, historical evolution, core equations, and a Python-based implementation of a topological handwritten digit classifier using artificial data. We avoid common libraries like sklearn and tensorflow, ensuring complete control and transparency of computation.

Keywords: *Topological Machine Learning,Topology,Homology,Topological Features in ML,Topological Digit Classifier.*

How To Site: Dr. Mitat Uysal (2025). Topological Machine Learning: Theoretical Foundations and a Custom Classifier on Artificial Data. *International Journal of Innovative Science and Research Technology,* 10(5), 3246-3248. https://doi.org/10.38124/ijisrt/25may1735

I. INTRODUCTİON

Machine learning models often rely on geometric and statistical structures. However, many real-world datasets have complex shapes or topologies, such as holes or connected components, that traditional ML algorithms overlook. Topological Data Analysis (TDA), particularly persistent homology, offers tools to capture such features, leading to the emergence of Topological Machine Learning [1–3].

➢ *Historical Background*

Topology entered data science through Morse theory and algebraic topology in the 20th century [4]. Key milestones:

- **1930s**: Morse theory was formalized by Marston Morse [5], showing how topology changes with smooth functions.
- **1940s–60s**: Homology theory matured, offering tools to detect k-dimensional holes [6].
- **1990s**: Persistent homology introduced by Edelsbrunner et al. [7], tracking topological features over multiple scales.
- **2000s**: TDA applied to sensor networks, biology, and ML [8–10].

➢ *Mathematical Foundations*

3.1 Simplicial Complex and Homology

A **simplicial complex** K consists of vertices V, edges E, triangles T, etc. Define the k**-chain** group C_k as:

$$C_k(K) = \left\{ \sum a_i \sigma_i \mid a_i \in \mathbb{Z}_2, \sigma_i \text{ is a } k\text{-simplex} \right\}$$

The **boundary operator** $\partial_k : C_k \to C_{k-1}$ satisfies $\partial_{k-1} \circ \partial_k = 0$.

Define the k**-th homology group** as:

$$H_k(K) = \frac{\ker(\partial_k)}{\mathrm{im}(\partial_{k+1})}$$

3.2 Persistent Homology

For a filtration $K_0 \subseteq K_1 \subseteq \cdots \subseteq K_n$, persistent homology tracks the **birth** and **death** of topological features over this sequence:

$$\text{Persistence Pair: } (\text{birth}, \text{death}) = (i, j) \text{ if a feature appears at } K_i \text{ and disappears at } K_j$$

Features are visualized using **persistence diagrams** and **barcodes** [11].

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25may1735

- *Persistent Homology*

For a filtration K0⊆K1⊆⋯⊆KnK_0 \subseteq K_1 \subseteq \dots \subseteq K_nK0⊆K1⊆⋯⊆Kn, persistent homology tracks the **birth** and **death** of topological features over this sequence:

Persistence Pair: (birth,death)=(i,j) if a feature appears at Ki and disappears at Kj\text{Persistence Pair: } (\text{birth}, \text{death}) = (i, j) \text{ if a feature appears at } K_i \text{ and disappears at } K_jPersistence Pair: (birth,death)=(i,j) if a feature appears at Ki and disappears at Kj

Features are visualized using **persistence diagrams** and **barcodes** [11].

➢ *Topological Features in Machine Learning*

Features derived from persistence diagrams are fed into ML pipelines through:

- Persistence landscapes $\lambda_k(t) \in \mathbb{R}$
- Persistence images $PI(x, y) \in \mathbb{R}$
- Vectorizations, e.g., Betti curves, lifespans

[12-15]

➢ *Python Program: Topological Digit Classifier(Figure-1)*

- *Artificial Data Generation*

We simulate simple digits 0 to 3 using pixel-style patterns (8×8 grid):

```
import numpy as np
import matplotlib.pyplot as plt
def generate_digit_data():
    patterns = {
        0:
np.array([[0,1,1,1,0],[1,0,0,0,1],[1,0,0,0,1],[0,1,1,1,0]]),
        1: np.array([[0,1,0],[1,1,0],[0,1,0],[1,1,1]]),
        2: np.array([[1,1,1],[0,0,1],[1,1,1],[1,0,0]]),
        3: np.array([[1,1,1],[0,0,1],[0,1,1],[1,1,1]])
    }
    data = []
    labels = []
    for label, pattern in patterns.items():
        for _ in range(50):  # 50 samples per digit
            noise = np.random.randint(0, 2, pattern.shape)
            sample = np.clip(pattern + noise, 0, 1)
            padded = np.zeros((8,8))
            padded[:sample.shape[0],  :sample.shape[1]] =
sample
            data.append(padded)
            labels.append(label)
    return np.array(data), np.array(labels)
```

➢ *Topological Feature Extraction: Counting Connected Components*

```
python
KopyalaDüzenle
def count_components(image):
    visited = np.zeros_like(image)
    count = 0
    def dfs(x, y):
        if x<0 or x>=8 or y<0 or y>=8 or image[x,y]==0
or visited[x,y]==1:
            return
        visited[x,y] = 1
        for dx in [-1,0,1]:
            for dy in [-1,0,1]:
                dfs(x+dx, y+dy)
    for i in range(8):
        for j in range(8):
            if image[i,j]==1 and visited[i,j]==0:
                dfs(i,j)
                count += 1
    return count
```

➢ *Classifier Using Nearest Topological Count*

```
def classify_by_topology(data, labels):
    features = np.array([count_components(img) for img
in data])
    predictions = []
    for f in features:
        closest_label = labels[np.argmin(np.abs(features -
f))]
        predictions.append(closest_label)
    return np.array(predictions)
```

➢ *Evaluation and Visualization*

```
data, labels = generate_digit_data()
preds = classify_by_topology(data, labels)

accuracy = np.mean(preds == labels)
print("Topological Classifier Accuracy:", accuracy)

for i in range(4):
    plt.subplot(1,4,i+1)
    plt.imshow(data[i], cmap='gray')
    plt.title(f'True: {labels[i]}, Pred: {preds[i]}')
plt.tight_layout()
plt.show()
```

II. CONCLUSİON

Topological Machine Learning introduces a new perspective in feature extraction, especially for non-Euclidean data structures. Through persistent homology and its variants, topological signatures prove valuable in classification, robustness to noise, and generalization. The digit classifier example shows that even simple topological counts (connected components) provide meaningful discriminative features.[16-20]

REFERENCES

[1]. Carlsson, G. (2009). Topology and data. *Bulletin of the American Mathematical Society*, 46(2), 255–308. https://doi.org/10.1090/S0273-0979-09-01249-X

[2]. Edelsbrunner, H., & Harer, J. (2010). *Computational Topology: An Introduction*. American Mathematical Society.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25may1735

[3]. Chazal, F., & Michel, B. (2017). An introduction to topological data analysis. *arXiv preprint*, arXiv:1710.04019. https://arxiv.org/abs/1710.04019
[4]. Munkres, J. R. (1984). *Elements of Algebraic Topology*. Addison-Wesley.
[5]. Morse, M. (1934). *The Calculus of Variations in the Large*. American Mathematical Society.
[6]. Hatcher, A. (2002). *Algebraic Topology*. Cambridge University Press.
[7]. Edelsbrunner, H., Letscher, D., & Zomorodian, A. (2002). Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4), 511–533. https://doi.org/10.1007/s00454-002-2885-2
[8]. Zomorodian, A., & Carlsson, G. (2005). Computing persistent homology. *Discrete & Computational Geometry*, 33(2), 249–274. https://doi.org/10.1007/s00454-004-1146-y
[9]. Singh, G., Mémoli, F., & Carlsson, G. (2007). Topological methods for the analysis of high dimensional data sets and 3D object recognition. *Eurographics Symposium on Point-Based Graphics*, 91–100. https://doi.org/10.2312/SPBG/SPBG07/091-100
[10]. Lum, P. Y., et al. (2013). Extracting insights from the shape of complex data using topology. *Nature Biotechnology*, 31(6), 545–552. https://doi.org/10.1038/nbt.2572
[11]. Adams, H., Emerson, T., Kirby, M., Neville, R., Peterson, C., Shipman, P., Chepushtanova, S., Hanson, E., Motta, F., & Ziegelmeier, L. (2017). Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18(8), 1–35.
[12]. Cohen-Steiner, D., Edelsbrunner, H., & Harer, J. (2007). Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1), 103–120. https://doi.org/10.1007/s00454-006-1276-5
[13]. Wasserman, L. (2018). Topological data analysis. *Annual Review of Statistics and Its Application*, 5, 501–532. https://doi.org/10.1146/annurev-statistics-031017-100045
[14]. Bendich, P., Wang, B., & Mukherjee, S. (2016). Persistent homology analysis of brain artery trees. *The Annals of Applied Statistics*, 10(1), 198–218. https://doi.org/10.1214/15-AOAS886
[15]. Reininghaus, J., Huber, S., Bauer, U., & Kwitt, R. (2015). A stable multi-scale kernel for topological machine learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4741–4748. https://doi.org/10.1109/CVPR.2015.7299040
[16]. Hofer, C., Kwitt, R., Niethammer, M., & Uhl, A. (2017). Deep learning with topological signatures. In *Advances in Neural Information Processing Systems (NeurIPS)*, 30.
[17]. Carrière, M., Cuturi, M., & Oudot, S. (2020). PersLay: A neural network layer for persistence diagrams and new graph topological signatures. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, PMLR 108, 2786–2796.
[18]. Bauer, U., & Kerber, M. (2014). PHAT – Persistent homology algorithm toolbox. *Mathematics in Computer Science*, 8(1), 93–115. https://doi.org/10.1007/s11786-014-0186-0
[19]. Maria, C., Boissonnat, J.-D., Glisse, M., & Yvinec, M. (2014). The GUDHI library: Simplicial complexes and persistent homology. *SoftwareX*, 5, 70–75. https://doi.org/10.1016/j.softx.2016.09.002
[20]. Oudot, S. Y. (2015). *Persistence Theory: From Quiver Representations to Data Analysis*. American Mathematical Society.

➢ *Output of the Code*

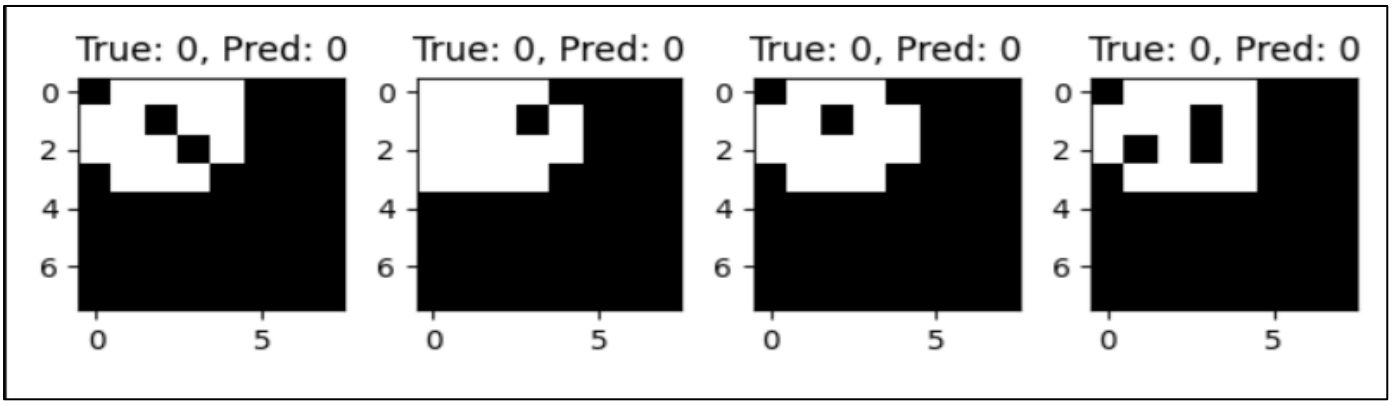

Fig 1 Handwritten Digits Identification