Project Report

Implementation of AID and THAID Decision Trees

From Scratch: A Deep Dive into Recursive Partitioning Algorithms

Authors: Djeri ALASSANI OUBENOUPOU, Asmae ELHAKIOUI and KASSIMI Achraf

Institution: Ensam Meknes

Course: Knowledge Data Discovery

Date: December 17, 2025

GitHub Repository:
https://github.com/1achraf1/AID-THAID-Implementation

Abstract

This project focuses on the implementation and analysis of two foundational decision tree algorithms: AID (Automatic Interaction Detector) and THAID (Theta Automatic Interaction Detector). Developed in the 1960s and 70s, these algorithms laid the groundwork for modern machine learning interpretation methods. The primary objective of this work is to reconstruct these algorithms from their original theoretical frameworks to understand the core mechanics of recursive partitioning.

The AID algorithm was implemented to handle regression tasks. It utilizes a variance-based splitting criterion, specifically aiming to minimize the within-group sum of squares (WSS) to identify interactions between predictors and a continuous dependent variable. Conversely, the THAID algorithm was implemented for classification tasks. It employs two distinct splitting criteria: the **Theta statistic (θ)**, which maximizes modal classification accuracy, and the **Delta statistic (δ)**, which maximizes the distributional difference (L_1 distance) between child nodes.

The implementation was developed from scratch (without relying on pre-built sklearn tree libraries) to demonstrate the low-level logic of node splitting, stopping criteria, and tree pruning. The performance of both models was evaluated on standard datasets, demonstrating the trade-offs between the greedy nature of these early algorithms and their interpretability, providing comparative insight into the evolution of decision tree methodologies.

Keywords: AID, THAID, Decision Trees, Regression, Classification, Recursive Partitioning, Theta Criterion, Delta Criterion, Machine Learning.

Contents

1 Introduction

1.1 Context and Motivation

Decision trees are among the most intuitive and interpretable machine learning algorithms. Their hierarchical structure mirrors human decision-making processes, making them particularly valuable in domains where model transparency is crucial, such as healthcare, finance, and policy-making. However, modern implementations like CART, C4.5, and Random Forests often obscure the fundamental principles that made decision trees revolutionary in the 1960s.

The AID (Automatic Interaction Detector) algorithm, introduced by Morgan and Sonquist in 1963 [1], represented a breakthrough in multivariate analysis by automatically detecting interactions between predictor variables. Prior to AID, researchers had to manually specify interaction terms, a process that was both tedious and prone to missing important relationships. THAID (Theta Automatic Interaction Detector), developed by Morgan and Messenger in 1973 [2], extended these principles to categorical dependent variables, introducing the Theta statistic as a splitting criterion.

Understanding these foundational algorithms provides insight into:

- The evolution of splitting criteria from variance reduction to information gain.
- The challenges of greedy recursive partitioning.
- The trade-offs between model complexity and interpretability.
- The computational considerations in tree-based learning.

1.2 Problem Statement

Modern machine learning libraries provide highly optimized implementations of decision tree algorithms, but they abstract away the underlying mechanics. This abstraction, while convenient, can lead to a superficial understanding of how these algorithms actually work. Specifically, understanding how variance minimization (AID) compares to mode maximization (THAID) requires a deep dive into the mathematical constraints of each method. This project addresses these questions by implementing AID and THAID from first principles.

2 Mathematical Foundations and Theory

This section details the theoretical framework underpinning decision trees, specifically focusing on the recursive partitioning logic used to create decision regions, and the distinct splitting criteria for AID (Regression) and THAID (Classification) as implemented in the codebase.

2.1 Recursive Partitioning and Decision Regions

The fundamental logic of a decision tree is the recursive partitioning of the feature space. Let $\mathcal{X}$ be the p-dimensional feature space such that $X \in \mathbb{R}^p$. The goal of the algorithm is to partition this space into M distinct, non-overlapping regions (or hyper-rectangles) $R_1, R_2, \ldots, R_M$.

For any input vector x, the model predicts a response $\hat{y}$ based on the region into which x falls. Mathematically, the function $f(x)$ can be expressed as:

$$f(x) = \sum_{m=1}^{M} c_m \cdot \mathbb{I}(x \in R_m) \tag{1}$$

Where:

- R_m represents the m-th region (leaf node).
- c_m is the constant prediction for region R_m.
- $\mathbb{I}(\cdot)$ is the indicator function, returning 1 if the condition is true and 0 otherwise.

2.2 AID: Regression via Variance Reduction

The Automatic Interaction Detector (AID) is designed for regression problems where the target variable Y is continuous. The objective of AID is to define regions R_m such that the variance within each region is minimized.

2.2.1 Splitting Criterion: WSS Minimization

To evaluate the quality of a split, AID uses the Within-Group Sum of Squares (WSS). For a parent node t containing N_t samples, the Total Sum of Squares (TSS) is:

$$TSS_t = \sum_{i \in t} (y_i - \bar{y}_t)^2 \tag{2}$$

When a candidate split divides node t into a left child t_L and a right child t_R, we calculate the WSS for the children:

$$WSS(t_L, t_R) = \sum_{i \in t_L} (y_i - \bar{y}_{t_L})^2 + \sum_{j \in t_R} (y_j - \bar{y}_{t_R})^2 \tag{3}$$

The optimal split is the one that maximizes the reduction in error:

$$\Delta_{AID} = TSS_t - WSS(t_L, t_R) \tag{4}$$

This is equivalent to finding the split that minimizes the combined variance of the child nodes.

2.3 THAID: Classification via Theta or Delta

The Theta Automatic Interaction Detector (THAID) is designed for classification. Unlike Entropy or Gini impurity used in modern trees (like C4.5 or CART), THAID allows for two distinct optimization strategies: maximizing modal accuracy (θ) or maximizing distributional separation (δ).

2.3.1 Splitting Criterion 1: The Theta Statistic (θ)

This criterion corresponds to the default `criterion='theta'` in the implementation. It aims to maximize the number of correctly classified samples in the child nodes (zero-one accuracy).

Let a split divide a parent node into a left child (t_L) and a right child (t_R). Let N_{total} be the number of samples in the parent. The count of the most frequent (modal) class in a node is given by $C_{max}(t)$. The score is calculated as:

$$S_\theta = \frac{C_{max}(t_L) + C_{max}(t_R)}{N_{total}} \tag{5}$$

This criterion prefers splits that create "pure" nodes dominated by a single class.

2.3.2 Splitting Criterion 2: The Delta Statistic (δ)

This criterion corresponds to `criterion='delta'`. Instead of focusing only on the majority class, it measures how different the probability distributions of the classes are between the two child nodes.

Let K be the number of classes. Let $p_{L,k}$ be the probability of class k in the left child, and $p_{R,k}$ be the probability of class k in the right child. The score is defined as the L_1 distance (Manhattan distance) between the two probability vectors:

$$S_\delta = \sum_{k=1}^{K} |p_{L,k} - p_{R,k}| \tag{6}$$

A higher δ indicates that the left and right nodes have very different class compositions, implying a strong interaction between the split feature and the dependent variable.

2.3.3 Heuristic for High Cardinality Categorical Features

When a categorical feature has cardinality $C >$ max_categories, an exhaustive search ($2^{C-1} - 1$ splits) is computationally expensive. The implementation uses a specific heuristic:

1. Identify the majority class y_{maj} of the current parent node.
2. For each category c in the feature, calculate the conditional probability:
$$P(c) = P(Y = y_{maj} \mid X = c) \tag{7}$$
3. Sort the categories in descending order based on $P(c)$.
4. Treat the sorted categories as ordinal values and perform a linear split search.

3 Implementation Strategy

This section outlines the distinct algorithmic approaches used for AID and THAID, reflecting the logic in the Python class structure.

3.1 Algorithm 1: AID (Regression)

The AID implementation focuses on sorting continuous features and finding the cut-point that minimizes variance.

Algorithm 1 AID Split Finding (Regression)

Input: Node indices I, Dataset X, y
$best\_gain \leftarrow -\infty$, $best\_split \leftarrow$ None
for each feature f **do**
    Sort data by feature f
    Calculate Total Sum of Squares (TSS) for current node
    for each unique split point s between values **do**
        Partition I into I_L (left) and I_R (right)
        Calculate $WSS_L = \sum(y_L - \bar{y}_L)^2$
        Calculate $WSS_R = \sum(y_R - \bar{y}_R)^2$
        $Gain \leftarrow TSS - (nl * WSS_L + nr * WSS_R)$
        if $Gain > best\_gain$ **then**
            $best\_gain \leftarrow Gain$
            $best\_split \leftarrow (f, s)$
        end if
    end for
end for
return $best\_split$

3.2 Algorithm 2: THAID (Classification)

The THAID implementation handles numeric and categorical features differently, using specific heuristics for categorical data to avoid the computational cost of testing all combinations.

Algorithm 2 THAID Split Finding (Classification)

Input: Node indices I, Dataset X, y, Criterion $crit \in \{\theta, \delta\}$
$best\_score \leftarrow -1$, $best\_split \leftarrow$ None
for each feature f **do**
    if f is Numeric **then**
        Sort data by f
    else ▷ f is Categorical
        if Unique Categories $\leq$ Max_Categories **then**
            Exhaustive: Generate all combinations
        else
            Heuristic:
            1. Find majority class y_{maj} of node
            2. Score cats by $P(y = y_{maj} \mid X = cat)$
            3. Sort cats desc and treat as ordinal
        end if
    end if
    for each candidate split s **do**
        Partition I into I_L (left) and I_R (right)
        Calculate class counts $N_{L,k}$ and $N_{R,k}$
        if $crit =$ 'theta' **then**
            $Score \leftarrow \frac{\max_k(N_{L,k}) + \max_k(N_{R,k})}{N_{total}}$
        else if $crit =$ 'delta' **then**
            $p_{L,k} \leftarrow N_{L,k} / \sum N_L; \quad p_{R,k} \leftarrow N_{R,k} / \sum N_R$
            $Score \leftarrow \sum_k |p_{L,k} - p_{R,k}|$
        end if
        if $Score > best\_score$ **then**
            $best\_score \leftarrow Score$
            $best\_split \leftarrow (f, s)$
        end if
    end for
end for
return $best\_split$

4 Lessons Learned

Through this implementation project, several key insights emerged:

- **Simplicity vs. Performance:** While AID and THAID are simpler than modern algorithms, they perform reasonably well on appropriate datasets, highlighting that algorithmic sophistication isn't always necessary.
- **Interpretability Trade-offs:** The transparency of these algorithms comes at the cost of potentially lower accuracy compared to ensemble methods, emphasizing the importance of choosing algorithms based on problem requirements.
- **Historical Context:** Understanding the evolution from AID/THAID to modern algorithms (CART, C4.5, Random Forests) provides valuable perspective on why certain design decisions were made.

References

[1] Morgan, J. N., & Sonquist, J. A. (1963). *Problems in the analysis of survey data, and a proposal.* Journal of the American Statistical Association, 58(302), 415–434.

[2] Morgan, J. N., & Messenger, R. C. (1973). *THAID: A sequential analysis program for the analysis of nominal scale dependent variables.* Survey Research Center, Institute for Social Research, University of Michigan.

[3] Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees.* CRC press.

[4] Quinlan, J. R. (1986). *Induction of decision trees.* Machine learning, 1(1), 81–106.

[5] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction* (2nd ed.). Springer Science & Business Media.