ISSN: 2997-2256

Research Article

Journal of Investment, Banking and Finance

Algorithms for Estimating Linear Function in Data Mining

Thomas Hoang*

Denison University, USA

***Corresponding Author**
Thomas Hoang, Denison University, USA.

Submitted: 2026, Feb 18; **Accepted:** 2026, Mar 11; **Published:** 2026, Mar 20

Citation: Hoang, T. (2026). Algorithms for Estimating Linear Function in Data Mining. *J Invest Bank Finance, 4*(1), 01-08.

Abstract

The main goal of this topic is to showcase several studied algorithms for estimating the linear utility function to predict the users' preferences. For example, if a user comes to buy a car that has several attributes including speed, color, age, etc in a linear function, the algorithms that we present in this paper help with estimating this linear function to filter out a small subset that would be of best interest to the user among a million tuples in a very large database. In addition, the estimating linear function could also be applicable in getting to know what the data can do or predicting the future based on the data that is used in data science, which is demonstrated by the GNN, PLOD algorithms [1,2]. In the ever-evolving field of data science, deriving valuable insights from large datasets is critical for informed decision-making, particularly in predictive applications. Data analysts often identify high-quality datasets without missing values, duplicates, or inconsistencies before merging diverse attributes for analysis. Taking housing price prediction as a case study, various attributes must be considered, including location factors (proximity to urban centers, crime rates), property features (size, style, modernity), and regional policies (tax implications). Experts in the field typically rank these attributes to establish a predictive utility function, which machine learning models use to forecast outcomes like housing prices. Several data discovery algorithms, including address the challenges of predefined utility functions and human input for attribute ranking, which often result in a time-consuming iterative process, which the work of cannot overcome[1-4]. The notable enhancement uses a Graph Neural Network (GNN) algorithm that builds on previous approaches. The GNN algorithm leverages the power of graph neural networks and large language models to interpret text-based values that earlier models like PLOD could not handle, significantly improving the reliability of outcome predictions. GNN extends PLOD's capabilities by incorporating numerical and textual data, offering a comprehensive approach to understanding user preferences for data science and analytics applications.

Keywords: Data Systems, Machine Learning, Graph Neural Networks, Large Language Model, Decision Making

1. Introduction

The method of indistinguishability query for identifying tuples that are close to optimal from a user's perspective [5]. This query retrieves all tuples that are only a small margin below the optimal value according to the user's unknown utility function. This approach is based on the insight that users often struggle to differentiate between very similar options, and even slightly suboptimal tuples may possess attributes that make them appealing. In addition, this framework asks the user to make a limited number of comparisons, helping to refine an understanding of their preferences. This is a big advancement compared to traditional that ask users for utility functions to proceed with filtering out user's preference of tuples, like Top-K algorithms, in which the Top-K algorithm requires the user's utility function, which can be difficult to obtain and is not always applicable where user preferences are not well-defined [6-9].

In addition to estimating the user's preference for unknown utility functions. The estimating linear function could also be applicable in getting to know what the data can do or predicting the future based on the data that is used in data science. For instance, a scientist may prioritize specific attributes over others based on their domain expertise. For example, an apartment in a central area like New York

City with low crime rates and modern amenities may be favored over a remote historic house. Even within similar urban environments, decision-making can be complex—such as weighing the pros and cons of a centrally located apartment against a slightly cheaper option in the suburbs. Other considerations, such as neighborhood friendliness or environmental tranquility, might influence rankings. A utility function captures these variations by assigning importance to each attribute in the prediction process.

Many machine learning algorithms and applied machine learning approaches often require predefined utility functions, meaning the user already understands the weights for each coefficient in linear function (supposedly ranging from 0 to 1), which can be impractical or difficult for many users to specify [4,10,11]. However, these methods have limitations, as they may still produce data subsets that don't fully align with the user's actual utility function due to mismatches between the predicted and true utility functions.

Inspired by the approach of Indistinguishability Query, the approach, GNN (Graph Neural Networks and Large Language Models for Data Discovery), overcomes these challenges to predict the future. GNN leverages user-defined attribute rankings and incorporates advanced machine-learning techniques. By combining PLOD's numerical prediction capabilities with the power of Graph Neural Networks (GNNs) and Large Language Models (LLMs) for text-based data, GNN provides more accurate utility function estimations. This allows for selecting data subsets that more closely reflect user preferences and improves prediction outcomes. By integrating complex models capable of handling large, mixed-type datasets, GNN represents a significant advancement in multimodal predictive applications.

2. Problem Definition

Notation	Meaning
T	The set of all tuples
D	The dataset with both numerical and textual features
X	The set of all attributes (numerical and textual) in D
X^{num}	The set of numerical attributes in D
X^{text}	The set of textual attributes in D
x_j^{num}	The jth numerical attribute in X^{num}
x_k^{text}	The kth textual attribute in X^{text}
$\beta_{j,\text{num}}$	Coefficient for the jth numerical attribute
$\beta_{k,\text{text}}$	Coefficient for the kth textual attribute
h_k^{text}	Feature embedding for the kth textual attribute generated by LLM
h_k^{GNN}	Combined feature embeddings processed by GNN
$u(x)$	Utility function estimating the score for a tuple
$u_{\text{syn}}(x)$	Synthetic utility function based on initial coefficient estimates
$u_{\text{real}}(x)$	Final utility function used for optimal tuple selection
t	A subset of T containing optimal tuples

Table 1: Notation and Meaning for GNN Algorithm

Linear Function

- $LINEAR = \{f | f(x) = \sum_{i=1}^{d} f(x_i)$
  {where each $f(x_i)$ is a linear function.}

$$u(x) = a \cdot x_1 + b \cdot x_2 + c \cdot x_3 + \cdots + z \cdot x_d,$$

where $u(x)$ is the utility function, and each term $f(x_i)$ is linear.

2.1. Model Representation for GNN Algorithm, Adapted from GNN [8]

$$u(\mathbf{x}) = \sum_{j=1}^{m} \beta_{j,\text{num}} \cdot x_j^{\text{num}} + \sum_{k=1}^{n} \beta_{k,\text{text}} \cdot x_k^{\text{text}} + \epsilon$$

where:

- $\mathbf{x} = (x_1^{\text{num}}, \ldots, x_m^{\text{num}}, x_1^{\text{text}}, \ldots, x_n^{\text{text}})$ is the vector of attributes, with x_j^{num} representing numerical attributes and x_k^{text}

representing textual attributes.

- $\beta_{j,\text{num}}$ and $\beta_{k,\text{text}}$ are the coefficients for numerical and textual attributes, respectively.
- ϵ is the error term.

2.2. Goal, Adapted from GNN [8]

We want to estimate the coefficients $\beta_{j,\text{num}}$ and $\beta_{k,\text{text}}$ for the utility function $u(\text{x})$, combining both numerical and textual data, to predict the utility score, [1].

2.3. Numerical Data Processing, Adapted from GNN [1]

For the numerical attributes, we want to apply linear regression techniques to estimate the coefficients $\beta_{j,\text{num}}$. to minimize the sum of squared errors (SSE) between the observed utility scores and those predicted by the model, the model is trained as below:

$$SSE_{\text{num}} = \sum_{i=1}^{N} (u_i - \hat{u}_i^{\text{num}})^2 = \sum_{i=1}^{N} \left(u_i - \sum_{j=1}^{m} \beta_{j,\text{num}} \cdot x_{i,j}^{\text{num}} \right)^2$$

Where N is the number of data points. Adapted from as in estimating numerical data for the utility function [2].

2.4. Textual Data Processing, Adapted from GNN [1]

For the textual attributes, we try to estimate the coefficients $\beta_{k,\text{text}}$. Meanwhile, we let the GNN process the graph structure of the dataset, where each node represents an attribute, and each edge represents a relationship between attributes. LLM processes textual data and outputs feature embeddings that are fed into the GNN:

$$\mathbf{h}_k^{\text{text}} = \text{GNN}(\text{LLM}(x_k^{\text{text}}))$$

Our goal is to minimize the SSE for textual attributes:

$$SSE_{\text{text}} = \sum_{i=1}^{N} (u_i - \hat{u}_i^{\text{text}})^2 = \sum_{i=1}^{N} \left(u_i - \sum_{k=1}^{n} \beta_{k,\text{text}} \cdot h_{i,k}^{\text{text}} \right)^2$$

2.5. Synthetic Utility Function, Adapted from GNN [1]

In order to refine the model, we first estimate a synthetic utility function $u_{\text{syn}}(\text{x})$ based on the combined coefficients $\beta_{j,\text{num}}$ and $\beta_{k,\text{text}}$:

$$u_{\text{syn}}(\mathbf{x}) = \sum_{j=1}^{m} \beta_{j,\text{num}} \cdot x_j^{\text{num}} + \sum_{k=1}^{n} \beta_{k,\text{text}} \cdot h_k^{\text{text}}$$

2.6. Real Utility Function, Adapted from GNN [1]

The final utility function $u_{\text{real}}(\text{x})$ is then estimated using the synthetic utility function as shown below:

$$u_{\text{real}}(\mathbf{x}) = \sum_{j=1}^{m} \gamma_{j,\text{num}} \cdot x_j^{\text{num}} + \sum_{k=1}^{n} \gamma_{k,\text{text}} \cdot h_k^{\text{text}} + \delta$$

where $\gamma_{j,\text{num}}$ and $\gamma_{k,\text{text}}$ are the refined coefficients, and δ is the error term.

2.7. Final goal

The main goal of GNN algorithm is to identify the optimal subsets $t \subset T$ that maximize the real utility function $u_{\text{real}}(\text{x})$:

$$\text{Optimal Subsets} = \arg\max_{t \subset T} \sum_{i \in t} u_{\text{real}}(\mathbf{x}_i)$$

3. About the Indistinguishability Query Algorithm

car	MPG	SR	MPG + 20SR
c_1	59	5	159
c_2	36	4	116
c_3	46	5	164
c_4	34	5	134
c_5	35	5	158

Table 2: Example of the Indistinguishability Query. All the Highlighted Tuples are 0.05-Indistinguishable from the Optimal (c_3) for a User with the Given Utility Function

Example: Consider Table I, as in Indistinguibility Query, which lists cars c_1 through c_5 [5]. Suppose Alice values fuel efficiency (MPG) and safety rating (SR) and has a utility function (unknown to her) given by f (MPG, SR) = MPG + 20SR. With this function, her top choice is $p* = p3$ since it achieves the highest utility score of 164. With an indistinguishability parameter of $\epsilon = 0.05$, Alice would be interested in any car that achieves at least $1/(1 + \epsilon) \approx 95.24\%$ of this maximum utility, which is approximately 156.2 (0.9524 × 164). Therefore, the indistinguishability query should return cars $\{c_1,3,c_5\}$, as highlighted in the table. Note that, while c_1 differs significantly from Alice's ideal car c_3, it provides a similar utility value for her preferences.

4. About the Plod and GNN Algorithm

The workflow of the algorithm GNN: First, users rank the dataset's attributes, which are then scaled to the range {0, 1} based on the rankings. GNN estimates the coefficients for each attribute using machine learning for numerical data and GNN and LLM techniques for textual data. These coefficients are then converted into scores within the same range. The process continues by synthesizing an initial utility function based on these coefficients. GNN then refines this estimate to generate a final utility function, offering a robust approach to aligning predicted data subsets with user preferences.

Graph Neural Networks and Large Language Models (GNN). The GNN algorithm shows the strengths of both GNNs and LLMs, and PLOD handles datasets, including numerical and text features, by asking users to rank attributes and then leveraging GNNs and LLMs, and PLOD to estimate utility function coefficients. Therefore, GNN offers a novel approach to data discovery that does not require explicit utility function definitions from the user. This method improves upon PLOD by incorporating advanced feature extraction and relational modeling capabilities, making it more robust to the complexities of multimodal data. Thus, GNN offers promising and reliance predicting data science and analytics applications.

5. Mathematical Proof for GNN Algorithm

By taking advantage of PLOD and Graph Neural Networks and Large Language Models, the GNN algorithm combines numerical and textual data processing using the advantage of PLOD as in , Graph Neural Networks (GNNs) as in several studies], and Large Language Models (LLMs) as in thus, to estimate the coefficients of a utility function that ranks the importance of different attributes to identify optimal subsets of data [2,12-16].

6. Utility Function and Error Term

Adapted from GNN, the real utility function is defined as:

$$u_{real}(x) = \sum_{j=1}^{m} y_{j,num} \cdot x_{j,num} + \sum_{k=1}^{n} y_{k,text} \cdot h_{k,text} + \delta$$

where the below input can be denoted as:

- m: Number of numerical attributes.
- n: Number of textual attributes.
- $x_{j,m}$: Numerical attribute values.
- $h_{k,xt}$: Textual embeddings.
- $y_{j,m}$, $y_{k,text}$: Coefficients for numerical and textual attributes, respectively.
- δ: Error term.

The error term δ is defined as the residual difference:

$$\delta_i = u(x_i) - \left(\sum_{j=1}^{m} y_{j,num} \cdot x_{j,num,i} + \sum_{k=1}^{n} y_{k,text} \cdot h_{k,text,i} \right)$$

6.1. Error Bound Derivation

To derive the bound for δ, we consider contributions from numerical attributes, textual embeddings, and inherent noise.

6.2. Numerical Contribution

For the numerical attributes, the contribution to the error is bounded by:

$$\left| \sum_{j=1}^{m} y_{j,num} \cdot x_{j,num} \right| \leq \|y_{num}\|_1 \cdot \max(X_{num})$$

Where:
Algorithms for estimating linear function in data mining

- $\|y_{num}\|_1 = \sum_{j=1}^{m} |y_{j,num}|$: The ℓ_1-norm of numerical coefficients.
- $\max(X_{num})$: Maximum value of numerical attributes.

6.3. Textual Contribution

For the textual attributes, the contribution to the error is bounded by:

$$\left| \sum_{k=1}^{n} y_{k,text} \cdot h_{k,text} \right| \leq \|y_{text}\|_1 \cdot \max(H_{text})$$

where:

- $\|y_{text}\|_1 = \sum_{k=1}^{n} |y_{k,text}|$: The ℓ_1-norm of textual coefficients.
- $\max(H_{text})$: Maximum magnitude of textual embeddings.

6.4. Inherent Noise

The inherent noise ϵ accounts for the variability or randomness in the data that the model cannot explain. Thus, the error term includes a base line noise component ϵ.

6.5. Final Error Bound

$$|\delta| \leq \|y_{num}\|_1 \cdot \max(X_{num}) + \|y_{text}\|_1 \cdot \max(H_{text}) + \epsilon$$

Combining the contributions, the error term δ is bounded by:
Expressing in terms of dataset parameters:

- $T = m + n$: Total number of attributes (numerical + textual).
- N: Number of rows in the dataset.
- q: Number of user-provided questions.

The error bound becomes:

$$|\delta| \leq \frac{1}{\sqrt{N}} \left(m \cdot \max(X_{num}) + n \cdot \max(H_{text}) \right) + q \cdot \epsilon$$

Thus, the bounds for the error of GNN [8] is:

$$0 \leq |\delta| \leq \frac{1}{\sqrt{N}} \left(m \cdot \max(X_{num}) + n \cdot \max(H_{text})\right) + q \cdot \epsilon$$

Algorithm 1 GNN: Graph Neural Networks and Large Language Models for Data Discovery [8]

Require: Dataset D with features $X = \{x_1, x_2, \ldots, x_n\}$ (both numerical and text), and labels y
Ensure: Subsets of optimal tuples $t \subseteq T$ with utility scores
1: **User Ranking:** Ask the user to rank all attributes $x_i \in X$
2: **for** each attribute $x_i \in X$ **do**
3: Scale down values of x_i to the range $\{0, 1\}$ based on ranking
4: **Coefficient Estimation:**
5: Train a Machine Learning model on numerical data to estimate coefficients β_{num}
6: Use Graph Neural Network (GNN) and Large Language Model (LLM) on text data to estimate coefficients β_{text}
7: Combine β_{num} and β_{text} to form overall coefficients $\beta = \{\beta_{num}, \beta_{text}\}$
8: **Synthetic Utility Function:**
9: Estimate synthetic utility function U_{syn} based on coefficients β
10: **Real Utility Function:**
11: Refine and estimate real utility function U_{real} coefficients based on U_{syn}
12: **Optimal Subsets Selection:**
13: Use U_{real} to score and filter the dataset D to find subsets of optimal tuples t
14: **Return** Optimal subsets t

7. GNN: Graph Neural Networks and Large Language Models for Data Discovery [1]

In lines 1-3:, we are gonna let the GNN algorithm ask the user to rank all attributes in the dataset X, which contains numerical and textual data. Based on the user's ranking, the attributes are scaled down to a normalized range $\{0, 1\}$. In the lines 4-8:, We then try to estimate the coefficients for each attribute β_{num} for numerical data. After this process, we then try to estimate the coefficients β_{text} for textual data. Following, the algorithm then combines these estimated coefficients into a comprehensive set $\beta = \{\beta_{\text{num}}, \beta_{\text{text}}\}$ that shows the overall importance of each attribute. In the following lines 9-10: After that of using the combined coefficients, the algorithm tries to estimate a synthetic utility function U_{syn}, which is a step designed to approximate the actual utility function. Next in the lines 11-12:, GNN tries to refine this synthetic utility function to produce the actual utility function U_{real}. Next in lines 13-14: Finally, we then try with the real utility function, applied to score and filter the dataset D to identify subsets of optimal tuples t. After all, in line 15: The algorithm finally returns the optimal subsets t as the final output.

8. Experiments

The comparison of precision demonstrates the accuracy when all algorithms run on the same inputs that return how many retrieved results are relevant to the query.

9. Why GNN Excels Over Other Algorithms

Algorithm	Limitations	Why GNN Excels
PLOD	Fails to capture relationships between tuples and lacks semantic understanding of textual data.	GNN integrates: • Numerical modeling using SSE_{num} for minimizing error in numerical predictions. • Textual understanding using embeddings $h_{k,text}$ generated by LLMs and refined by GNN. • Relational learning through graph-based message passing.
Top-K	Assumes independence of attributes and ignores relationships between tuples.	GNN dynamically adjusts rankings by: • Capturing relationships through graph propagation. • Combining numerical and textual features in a unified synthetic utility function.

Skyline	Computationally expensive for high-dimensional datasets and lacks semantic understanding for textual data.	GNN efficiently handles high-dimensional data by: • Scaling computations using graph structures. • Jointly optimizing numerical and textual attributes via the real utility function.
Multi-objective	Requires manual weight tuning, leading to biases and inconsistencies in results.	GNN optimizes holistically by: • Learning weights dynamically for both numerical and textual features. • Identifying optimal subsets of tuples via the maximization of the real utility function.

Table 3: Why GNN Excels Compared to Other Algorithms

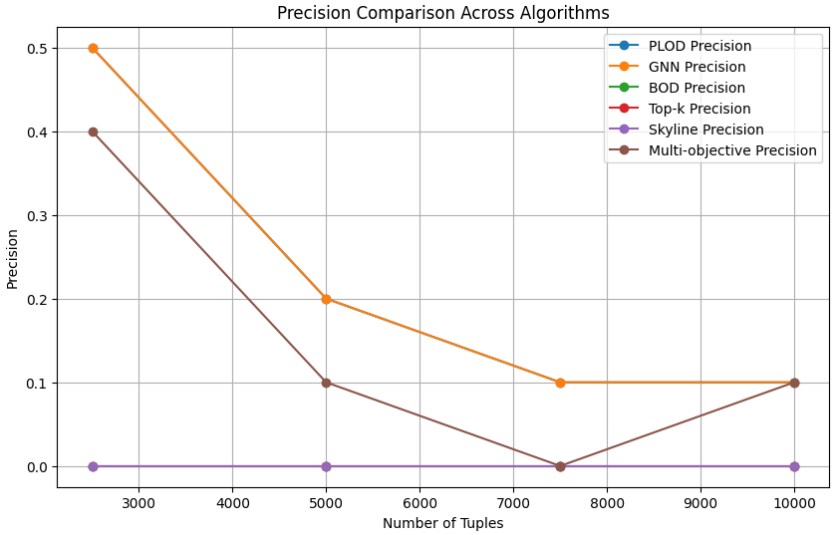

Figure 1: Precision Comparison With Changes in Number of Tuples

6. Conclusion

The GNN performs with the highest precision compared to the previous approaches, as in Figure 1, but takes a comparable amount of time to accomplish. This happens because machine learning is used to estimate the coefficients compared to other approaches for numerical values, and graph neural networks and large language models are used to understand and weigh the textual values from a huge dataset, in which other approaches only understand the numerical values input from the user.

References

1. Hoang, T. (2024). Gnn: Graph neural network and large language model for data discovery. *arXiv preprint arXiv:2408.13609*.
2. Hoang, T. (2024). PLOD: Predictive Learning Optimal Data Discovery. *Available at SSRN 4882133*.
3. Hoang, T. (2024). BOD: Blindly Optimal Data Discovery. *arXiv preprint arXiv:2401.05712*.
4. Galhotra, S., Gong, Y., & Fernandez, R. C. (2023, April). Metam: Goal-oriented data discovery. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)* (pp. 2780-2793). IEEE.
5. Lall, A. (2024, May). The Indistinguishability Query. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)* (pp. 475-487). IEEE.
6. Qin, L., Yu, J. X., & Chang, L. (2012). Diversifying top-k results. *arXiv preprint arXiv:1208.0076*.
7. Soliman, M. A., Ilyas, I. F., & Chang, K. C. C. (2006, April). Top-k query processing in uncertain databases. In *2007 IEEE 23rd international conference on data engineering* (pp. 896-905). IEEE.
8. Khosla, C., & Kakkar, P. (2015). Top-k Query Processing Techniques in Uncertain Databases: A Review. *International Journal of Computer Applications, 120*(20).
9. Xiao, G., Wu, F., Zhou, X., & Li, K. (2016). Probabilistic top-k range query processing for uncertain databases. *Journal of Intelligent & Fuzzy Systems, 31*(2), 1109-1120.
10. Fernandez, R. C., Abedjan, Z., Koko, F., Yuan, G., Madden, S., & Stonebraker, M. (2018, April). Aurum: A data discovery system.

In *2018 IEEE 34th International conference on data engineering (ICDE)* (pp. 1001-1012). IEEE.
11. Gong, Y., Zhu, Z., Galhotra, S., & Fernandez, R. C. (2023, April). Ver: View discovery in the wild. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)* (pp. 503-516). IEEE.
12. Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.
13. Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in neural information processing systems, 30*.
14. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2018, February). Graph attention networks. In *International conference on learning representations* (Vol. 6, p. 2).
15. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems, 33*, 1877-1901.
16. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019, June). Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers) (pp. 4171-4186).