Knowledge Discovery & Data Mining

— Data Preprocessing —

Data Transformation

Instructor: Yong Zhuang

yong.zhuang@gvsu.edu

Based on the original version by Professor Jiliang Tang

Outline

- Data Transformation
  - Robust Normalization
  - l2 normalization
  - Handling Categorical Features
    - Ordinal Encoding
    - One-Hot Encoding for Nominal
  - Encoding Image Data
  - Encoding Text Data:
  - Common Approach
  - Bag of Words(BoW)
  - Term Frequency Inverse Document Frequency (TF-IDF),
  - Word Embedding:
    - Word2Vec Continuous Bag of Words Model Skip-Gram Model

Robust Normalization

Robust normalization scales the original data using the median and the interquartile range (IQR), which are less sensitive to outliers.

Suppose that Median_A is the median value and IQR_A is the interquartile range of an attribute A. Robust normalization maps a value v_i, of A to v'_i by computing:

$$v_i' = \frac{v_i - median_A}{IQR_A}$$

When to Use:

- Ideal when your data contains **outliers** or anomalies that could skew statistical measures like the mean and standard deviation.
- Helps bring all features to the same scale without being influenced by extreme values.

.

Robust Normalization

Example. Suppose that the quartiles of the values for the attribute income are as follows:

- First Quartile (Q1): $36,000
- Median (Q2): $48,000
- Third Quartile (Q3): $60,000

$$v_i' = \frac{v_i - median_A}{IQR_A}$$

Using robust normalization, a value of $73,600 for income is transformed to:

Robust Normalization

Example. Suppose that the quartiles of the values for the attribute income are as follows:

- First Quartile (Q1): $36,000
- Median (Q2): $49,600
- Third Quartile (Q3): $60,000

$$v_i' = \frac{v_i - median_A}{IQR_A}$$

Using robust normalization, a value of $73,600 for income is transformed to:

$$v_i' = \frac{\$73,600 - \$49,600}{\$24,000} = 1$$

L2 Normalization

L2 Normalization scales the original data such that each feature vector has a Euclidean length of 1, effectively projecting the data onto the unit circle or sphere.

Suppose that x_i is a feature vector. L2 Normalization maps x_i to x'_i by computing:

$$\mathbf{x}_i' = \frac{\mathbf{x}_i}{||\mathbf{x}_i||_2}$$

Where $||\mathbf{x}_i||_2$ is the Euclidean (L2) norm of x_i, calculated as: $||\mathbf{x}_i||_2 = \sqrt{\sum_{j=1}^{n} x_{ij}^2}$

- x_{ij} represents the j-th element of the feature vector x_i.
- n is the number of features in the vector

L2 Normalization

Example. Suppose that the feature vector X = [3, 4, 0], so using L2 normalization, X can be transferred to X' =

$$\mathbf{x}_i' = \frac{\mathbf{x}_i}{||\mathbf{x}_i||_2} \qquad ||\mathbf{x}_i||_2 = \sqrt{\sum_{j=1}^{n} x_{ij}^2}$$

L2 Normalization

Example. Suppose that the feature vector X = [3, 4, 0], so using L2 normalization, X can be transferred to X' =

$$\mathbf{x}_i' = \frac{\mathbf{x}_i}{||\mathbf{x}_i||_2} \qquad ||\mathbf{x}_i||_2 = \sqrt{\sum_{j=1}^{n} x_{ij}^2}$$

$$\mathbf{x}_i' = \frac{\mathbf{x}_i}{||\mathbf{x}_i||_2} = \left[\frac{3}{5}, \frac{4}{5}, \frac{0}{5}\right] = [0.6, 0.8, 0]$$

Ordinal Encoding for Ordinal

Ordinal Encoding is a technique used to convert categorical data into numerical values where the categories have a meaningful order or ranking. In this method, each unique category is assigned an integer based on its rank or order.

Education	Education_encoded
High School	1
Bachelor's	2
PhD	4
Master's	3
Bachelor's	2

One-Hot Encoding for Nominal

One-Hot Encoding is a technique used to convert nominal (categorical) data into numerical form. This method transforms each category into a new binary column, where a value of 1 indicates the presence of the category and 0 indicates its absence.

- Converts a feature with n values to n binary features.
- Adds a new 0/1 feature for every category, having 1 (hot) if the sample has that category.
- Can significantly increase dimensionality if a feature has many unique categories, potentially leading to sparse data.
- Requires handling new categories in the test set that were not seen during training.

One-Hot Encoding for Nominal

Color	Color_Red	Color_Blue	Color_Green
Red	1	0	0
Blue	0	1	0
Green	0	0	1
Red	1	0	0

Encoding Image Data

When working with image data, pixel values are usually represented as integers in the range 0–255, indicating grayscale intensity levels. To properly preprocess the data for a neural network, follow these steps:

- Convert the image data type to float32 to ensure compatibility with the neural network's computations.
- Normalize the pixel values by dividing each value by 255, scaling them to a range of 0–1. This improves model performance by standardizing the input.

Encoding Text Data

Text encoding is the process of transforming text data into numerical form so that predictive algorithms can process it.

One common approach is to represent text as sequences of word indexes:

- Tokenization
- Assigning Unique Indexes to Words
- Representing Each Text as a List of Word Indexes
- Handling Varying Lengths
- Transformation into Float32 Tensors: One-Hot Encoding

Common approach

doc1: "The cat sat on the mat."

doc2: "The dog fell asleep."

Tokenize

[The, cat, sat, on, the, mat]

[The, dog, fell, asleep]

Vocabulary	Index
the	1
cat	2
sat	3
on	4
mat	5
dog	6
fell	7
asleep	8

doc1	
1	[1, 0, 0, 0, 0, 0, 0, 0]
2	[0, 1, 0, 0, 0, 0, 0, 0]
3	[0, 0, 1, 0, 0, 0, 0, 0]
4	[0, 0, 0, 1, 0, 0, 0, 0]
1	[1, 0, 0, 0, 0, 0, 0, 0]
5	[0, 0, 0, 0, 1, 0, 0, 0]

doc2	
1	[1, 0, 0, 0, 0, 0, 0, 0]
6	[0, 0, 0, 0, 0, 1, 0, 0]
7	[0, 0, 0, 0, 0, 0, 1, 0]
8	[0, 0, 0, 0, 0, 0, 0, 1]
0	[0, 0, 0, 0, 0, 0, 0, 0]
0	[0, 0, 0, 0, 0, 0, 0, 0]

Bag of Words(BoW)

doc1: "The cat sat on the mat."

doc2: "The dog sat on the log."

Tokenize

[The, cat, sat, on, the, mat]

[The, dog, sat, on, the, log]

Vocabulary	doc1	doc2
the	2	2
cat	1	0
sat	1	1
on	1	1
mat	1	0
dog	0	1
log	0	1

Simple, easy, and explainable.

Bag of Words(BoW)

Limitations:

Compound Word: AI, New York

Word Correlation: Cake, Baking

Polymorphous (Multiple Meanings): "Python" (Programming) vs. "python" (Animal)

Word Order: [Flight, GR, Chicago, from, to] (from GR to Chicago? or from Chicago to GR?)

Sparsity: With a large vocabulary, each vector contains many zeros (sparse).

Possible Solutions and Enhancements

Use N-grams: phrase, treating common phrases ("New York") as a single unit.

Stemming: Reduces words to their root form: coding, coded, codes, code => code

Term Frequency Inverse Document Frequency(TF-IDF)

TF-IDF is used to calculate the importance of a word in a document relative to a collection (or corpus) of documents. It provides a score (or weight) associated with each word to indicate its relevance within a specific document.

- **Term Frequency (TF)** Measures how often a word appears in a specific document. The higher the frequency of the term in the document, the higher its TF score or weight.

$$\mathrm{TF}(t,d) = \frac{\text{Number of times term } t \text{ occurs in document } d}{\text{Total number of terms in document } d}$$

- **Inverse Document Frequency (IDF)** Measures how common or rare a word is across the entire corpus. Words that appear in many documents (e.g., "the", "some", etc.) are less important and receive a lower score.

$$\mathrm{IDF}(t) = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing term } t}\right)$$

Term Frequency Inverse Document Frequency(TF-IDF)

- High TF-IDF: A word that occurs frequently in a document but rarely across the corpus, indicating high relevance to that document.
- Low TF-IDF: A word that occurs across many documents, making it less useful for identifying relevant content.

$$\text{TF-IDF}(t,d) = \text{TF}(t,d) \times \text{IDF}(t)$$

Term Frequency Inverse Document Frequency(TF-IDF)

doc1: "The cat sat on the mat."

doc2: "The dog sat on the log."

Tokenize

[The, cat, sat, on, the, mat]

[The, dog, sat, on, the, log]

$$\mathrm{TF}(t,d) = \frac{\text{Number of times term } t \text{ occurs in document } d}{\text{Total number of terms in document } d}$$

$$\mathrm{IDF}(t) = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing term } t}\right)$$

$$\text{TF-IDF}(t,d) = \mathrm{TF}(t,d) \times \mathrm{IDF}(t)$$

(using the base 10 logarithm)

Term Frequency Inverse Document Frequency(TF-IDF)

doc1: "The cat sat on the mat."

doc2: "The dog sat on the log."

Tokenize

[The, cat, sat, on, the, mat]

[The, dog, sat, on, the, log]

$$\mathrm{TF}(t,d) = \frac{\text{Number of times term } t \text{ occurs in document } d}{\text{Total number of terms in document } d}$$

$$\mathrm{IDF}(t) = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing term } t}\right)$$

$$\text{TF-IDF}(t,d) = \mathrm{TF}(t,d) \times \mathrm{IDF}(t)$$

(using the base 10 logarithm)

Vocabulary	TF_doc1	TF_doc2	IDF	doc1	doc2
the	2/6 = 0.333	2/6 = 0.333	log(2/2) = 0	0	0
cat	1/6 = 0.167	0	log(2/1) = 0.3	0.05	0
sat	1/6 = 0.167	1/6 = 0.167	log(2/2) = 0	0	0
on	1/6 = 0.167	1/6 = 0.167	log(2/2) = 0	0	0
mat	1/6 = 0.167	0	log(2/1) = 0.3	0	0
dog	0	1/6 = 0.167	log(2/1) = 0.3	0.05	0
log	0	1/6 = 0.167	log(2/1) = 0.3	0	0.05
				0	0.05

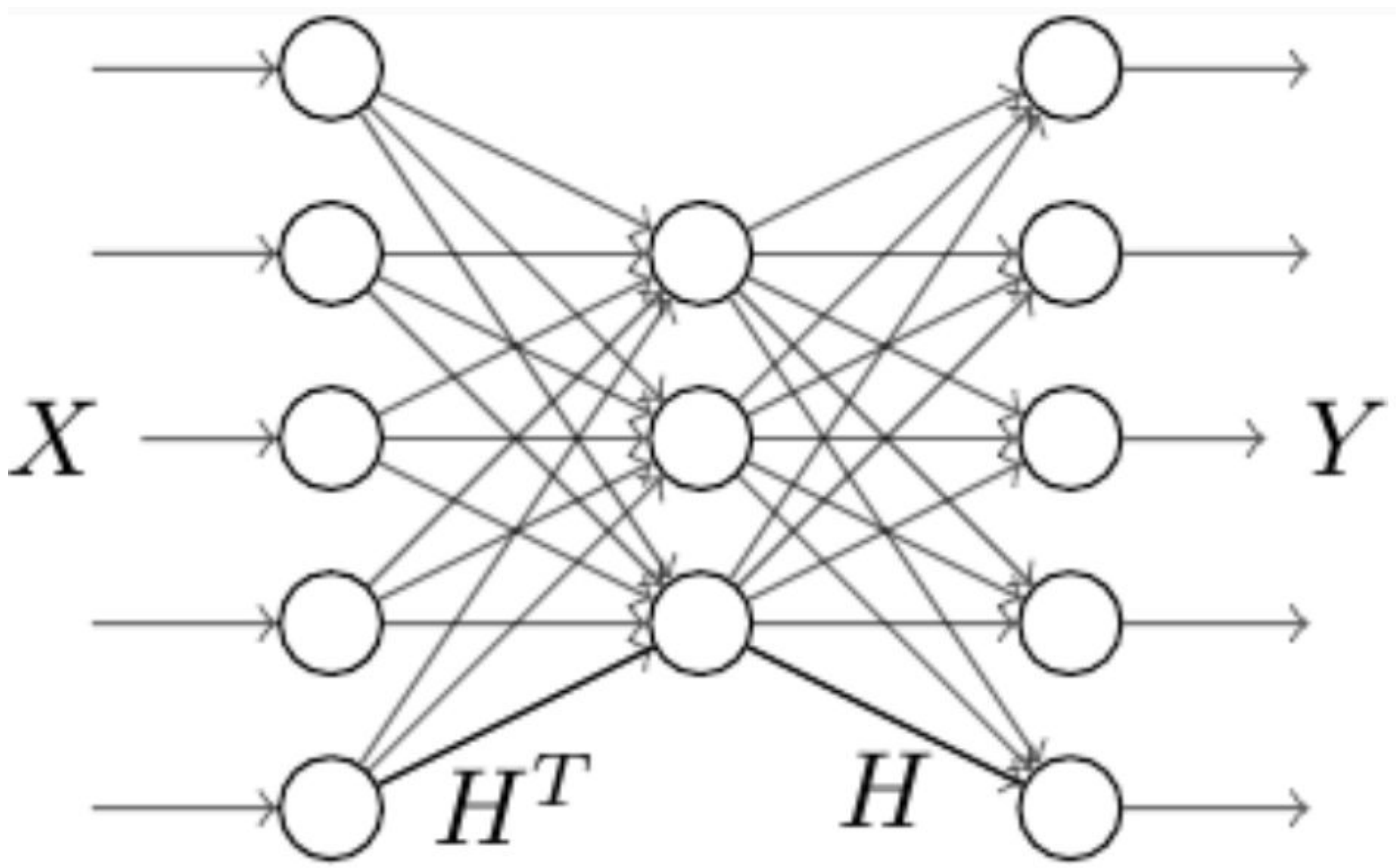
Word Embedding: Word2Vec

Continuous Bag of Words Model

plots are made with the ggplot2 package in R

Input is average of surrounding words:

x_i = ("plots" + "made" + "package" + "R") / 4



Target is average of surrounding words:

y_i = "ggplot2"

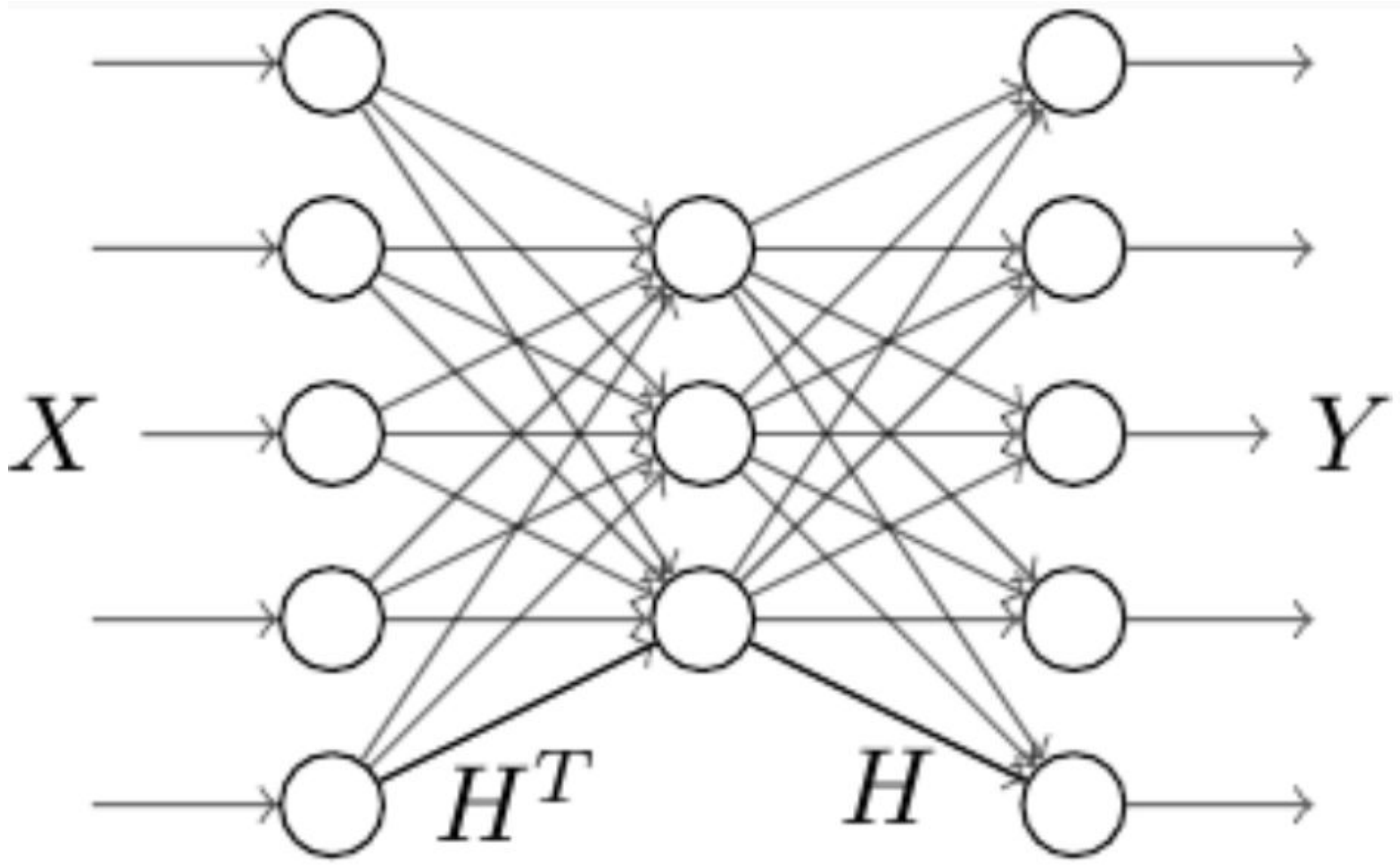
Word Embedding: Word2Vec

Skip-Gram Model

plots are made with the ggplot2 package in R

Input is center word:

x_i = “ggplot2”



Target is average of surrounding words:

y_i = (“plots” + “made” + “package” + “R”) / 4

Summary

Data Transformation

- Robust Normalization
- l2 normalization
- Handling Categorical Features
  - Ordinal Encoding
  - One-Hot Encoding for Nominal
- Encoding Image Data
- Encoding Text Data:
- Common Approach
- Bag of Words(BoW)
- Term Frequency Inverse Document Frequency (TF-IDF),
- Word Embedding:
  - Word2Vec Continuous Bag of Words Model Skip-Gram Model