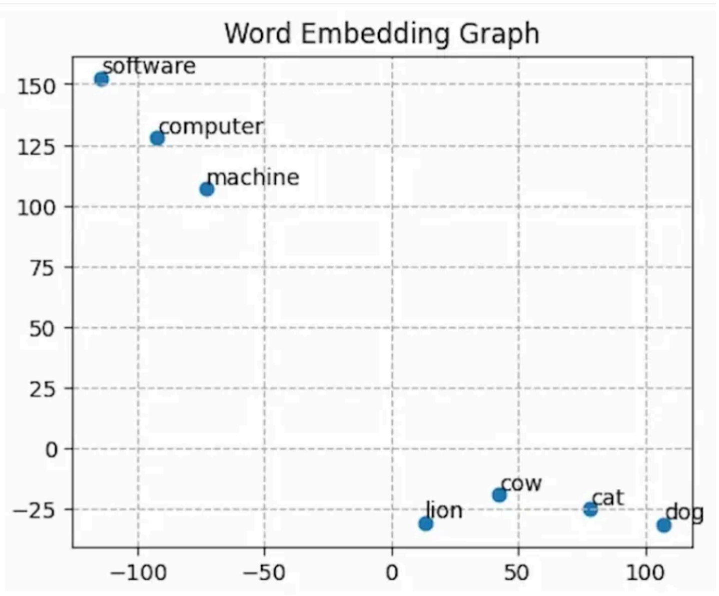
What are Embedding in Machine Learning?

Last Updated : 17 Sep, 2025

In machine learning, embeddings are a way of representing data as numerical vectors in a continuous space. They capture the meaning or relationship between data points, so that similar items are placed closer together while dissimilar ones are farther apart. This makes it easier for algorithms to work with complex data such as words, images or audios in a recommendation system.

- They convert categorical or high-dimensional data into dense vectors.
- They help machine learning models work with different types of data.
- These vectors help show what the objects mean and how they relate to each other.
- They are widely used in natural language processing, recommender systems and computer vision.



Word

In the above graph, we observe distinct clusters of related words.

- For instance "computer", "software" and "machine" are clustered together, indicating their semantic similarity.
- Similarly "lion", "cow" ,"cat" and "dog" form another cluster, representing their shared attributes.
- There exists a significant gap between these clusters highlighting their dissimilarity in meaning or context.

Key terms used for Embedding

1. Vector

- A vector is a list of numbers that describes a size (magnitude) and a direction. In machine learning, it usually means a set of numbers that shows features or characteristics of something.
- **Example**: In 2D, the vector points 3 steps along the x-axis and 4 steps along the y-axis. Its total length (magnitude) is 5.

2. Dense Vector

- A dense vector is a type of vector where most numbers are not zero. In machine learning, dense vectors are often used to describe things like words, images or data points because they capture a lot of details.
- **Example**: [2000, 3, 5, 9.8] could describe a house, showing size, number of bedrooms, bathrooms and age.

3. Vector space

- A vector space or linear space is a mathematical structure consisting of a set of vectors that can be added together and multiplied by scalars, satisfying certain properties. It satisfy the certain properties like Closure under addition and Scalar multiplication.
- **Example**: The set of all 3D vectors with real-number coordinates forms a vector space like the vectors [1, 0, 0], [0, 1, 0] and [0, 0, 1] constitute a basis for the 3D vector space.

4. Continuous Vector space

- A continuous vector space is a special kind of vector space where each value can be any real number (not just whole numbers). In embeddings, it means every object can be described with numbers that can smoothly change.
- **Example**: The color [0.9, 0.3, 0.1] in RGB shows a shade of red, where each number can be any value between 0 and 1.

How do Embeddings Work?

Let's see how embeddings work:

1. Define similarity signal:

First, decide what we want the model to treat as “similar”.

- **Text:** Words or sentences that appear in similar contexts.
- **Images:** Pictures of the same object or scene.
- **Graphs:** Nodes that are connected or related.

2. Choose dimensionality:

Select how many numbers (dimensions) will describe each item, it could be 64, 384, 768 or more.

- **More dimensions:** more detail but slower and uses more memory.
- **Fewer dimensions:** faster but may lose detail.

3. Build the encoder

This is the model that turns our data into a list of numbers (vector):

- **Text:** Language models like BERT.
- **Images:** Vision models like CNN or ViT.
- **Audio:** Models that process sound (e.g., turning it into spectrograms first).
- **Graphs:** Methods like Node2Vec or graph neural networks.
- **Tabular data:** Models that compress features into embeddings.

4. Train with a metric-learning objective:

- Show the model examples of things that are "similar" and "different."
- Teach it to place similar ones close together and different ones far apart.
- This process is called metric learning.

5. Negative sampling and batching:

Give the model tricky "hard negative" examples, things that seem alike but aren't so it learns to tell them apart better.

6. Validate and Tune

Test how well our embeddings work by checking:

- How accurate search results are.
- How well items group into the right categories.
- How good automatic clustering is.

If the results aren't good, adjust vector size, training method or data.

7. Index for Fast Retrieval

Store our vectors in a special database like Qdrant or FAISS to quickly find the closest matches, even from millions of items.

8. Use the embeddings

Once ready, embeddings can be used for:

- **Semantic search:** finding by meaning, not exact words.
- **RAG (Retrieval-Augmented Generation):** feeding relevant facts to an AI model.
- **Classification:** predicting the correct label or category.
- **Clustering:** grouping similar items together.
- **Recommendations:** suggesting similar products, content or users.
- **Monitoring:** spotting unusual changes or patterns over time.

Importance of Embedding

Embeddings are used across various domains and tasks for several reasons:

- **Semantic Representation:** Embeddings capture semantic relationships between entities in the data. For example, in word embeddings, words with similar meanings are mapped to nearby points in the vector space.
- **Dimensionality Reduction:** Embeddings reduce the dimensionality of data while preserving important features and relationships.
- **Transfer Learning:** Embeddings learned from one task or domain can be transferred and fine-tuned for use in related tasks or domains.
- **Feature Engineering:** Embeddings automatically extract meaningful features from raw data, reducing the need for manual feature engineering.
- **Interpretability:** Embeddings provide interpretable representations of data. For example, in word embeddings, the direction and distance between word vectors can correspond to meaningful relationships such as gender, tense or sentiment.

Objects that can be Embedded

From textual data to images and beyond, embeddings offer a versatile approach to encoding

1. Words

Word embeddings are numeric vectors that represent words in a continuous space, where similar words are placed near each other. These vectors are learned from large text datasets and capture the meanings and relationships between words making it easier for computers to understand and process language in tasks like sentiment analysis and translation.

Some of the Popular word embeddings include:

- Word2Vec
- GloVe (Global Vectors for Word Representation)
- FastText
- BERT (Bidirectional Encoder Representations from Transformers)
- GPT

2. Complete Text Document

Text embeddings or document embeddings represent entire sentences, paragraphs or documents as numeric vectors in a continuous space. Unlike word embeddings that focus on single words, text embeddings capture the meaning and context of longer text segments. This allows for easier comparison and analysis of complete pieces of text in NLP tasks like sentiment analysis, translation or document classification.

Some of the Popular text embedding models include:

- Doc2Vec

- Universal Sentence Encoder (USE)
- BERT
- ELMO

3. Audio Data

Audio data includes individual sound samples, audio clips and entire audio recordings. By representing audio as dense vectors in a continuous vector space, embedding techniques effectively capture acoustic features and relationships. This enables a wide range of audio processing tasks such as speech recognition, speaker identification, emotion detection and music genre classification.

Some of the popular Audio embedding techniques may include Wav2Vec

4. Image Data

Image embeddings are numerical representations of images in a continuous vector space, extracted by processing images through convolutional neural networks (CNNs). These embeddings encode the visual content, features and semantics of images, facilitating efficient understanding and processing of visual information by machines.

Some of the popular CNNs based Image embedding techniques include:

- VGG
- ResNet
- Inception
- EfficientNet

5. Graph Data

Graph embeddings convert a graph’s nodes and edges into numeric vectors, capturing the graph’s structure and relationships. This representation makes complex graph data easier for machine learning models to use enabling tasks like node classification, link prediction and clustering.

Some popular graph embedding techniques include:

- Node2Vec
- DeepWalk
- Graph Convolutional Networks

6. Structured Data

Structured data such as feature vectors and tables can be embedded to help machine learning models capture underlying patterns. Common techniques include Autoencoders

Visualization of Word Embeddings using t-SNE

Visualizing word embeddings can provide insights into how words are positioned relative to each other in a high-dimensional space. In this code, we demonstrate how to visualize word embeddings using t-SNE (t-distributed Stochastic Neighbor Embedding) a technique for dimensionality reduction after training a Word2Vec model on the 'text8' corpus.

Step 1: Import Libraries

- NumPy: Handles numerical data and array manipulation.
- Matplotlib: Creates plots and visualizations.
- scikit-learn: Reduces high-dimensional vectors to two dimensions for easy visualization.
- Gensim: Downloads text datasets and trains word embedding models.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
import gensim.downloader as api
from gensim.models import Word2Vec
```

Step 2: Load Data and Train Word2Vec Model

Loads a sample text dataset and uses it to train a Word2Vec model which creates word vectors.

```
corpus = api.load('text8')
model = Word2Vec(corpus)
```

Step 3: Select Words and Get Their Embeddings

- Chooses a list of sample words.
- Extracts their vector representations from the model as NumPy arrays.

```
words = ['cat', 'dog', 'elephant', 'lion', 'bird', 'rat', 'wolf', 'cow',
         'goat', 'snake', 'rabbit', 'human', 'parrot', 'fox', 'peacock',
         'lotus', 'roses', 'marigold', 'jasmine', 'computer', 'robot',
         'software', 'vocabulary', 'machine', 'eye', 'vision',
         'grammar', 'words', 'sentences', 'language', 'verbs', 'noun',
         'transformer', 'embedding', 'neural', 'network', 'optimization']
words = [word for word in words if word in model.wv.key_to_index]
word_embeddings = [model.wv[word] for word in words]
embeddings = np.array(word_embeddings)
```

Step 4: Reduce Dimensionality with t-SNE

Uses t-SNE from scikit-learn to shrink high-dimensional word vectors into two dimensions for visualization.

```
tsne = TSNE(n_components=2, perplexity=2)
embeddings_2d = tsne.fit_transform(embeddings)
```

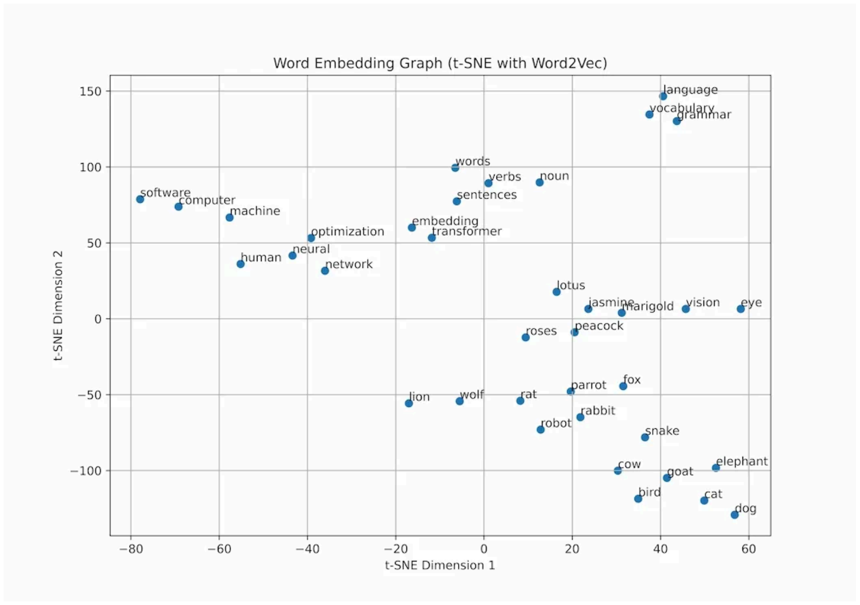
Step 5: Plot Embedding

Displays a scatter plot of the words in 2D space, labels each point with its word and displays the plot.

```
plt.figure(figsize=(10, 7), dpi=1000)
plt.scatter(embeddings_2d[:, 0], embeddings_2d[:, 1], marker='o')
for i, word in enumerate(words):
    plt.text(embeddings_2d[i, 0], embeddings_2d[i, 1],
             word, fontsize=10, ha='left', va='bottom')
plt.xlabel('t-SNE Dimension 1')
plt.ylabel('t-SNE Dimension 2')
plt.title('Word Embedding Graph (t-SNE with Word2Vec)')
plt.grid(True)
plt.savefig('embedding.png')
plt.show()
```

Output:

> *Original embedding vector shape (37, 100)*
>
> *After applying t-SNE embedding vector shape (37, 2)*



Output

Here we can see snake, cow, birds, etc are grouped together nearby showing similarity (all animals) whereas computer and machines are far away from animal cluster showing disimilarity.

Suggested Quiz

4 Questions

In machine learning, embeddings primarily help in:

- (A) Representing categorical/high-dimensional data as dense vectors
- (B) Storing raw text in a database
- (C) Converting numerical data back to text
- (D) Visualizing only images

Login to View Explanation

1/4

< Previous Next >

Comment

A anura... + Follow 1