ML Concepts- Embeddings

Categorical Input Data

- Categorical data represents discrete items from a finite set of choices, such as movies a user has watched, words in a document, or a person's occupation.
- Sparse tensors are the most efficient way to represent categorical data, containing very few non-zero elements. For example, in a movie recommendation model, each user's movie-viewing history can be represented as a sparse tensor.
- Sparse vectors can be used to represent words, sentences, and documents, where each word in the vocabulary corresponds to a node in the vector.
- One way to represent a word as a vector of numbers is through one-hot encoding, where only one index has a non-zero value.
- A "bag of words" representation is another approach, where several nodes in the vector have non-zero values, representing the counts of words in a larger chunk of text.
- Sparse representations can lead to challenges in learning effectively due to the large size of input vectors and the lack of meaningful relations between vectors.
- Embeddings provide a solution by translating large sparse vectors into a lower-dimensional space that preserves semantic relationships, making it easier for models to learn effectively. Embeddings help address issues related to the size of the network and the lack of meaningful relations between vectors.

Translating to lower dimensional space

- Mapping high-dimensional data into a lower-dimensional space can help address the core problems of sparse input data.
- A small multi-dimensional space provides the ability to group semantically similar items together and maintain distance between dissimilar items, allowing for meaningful encoding of semantics.
- Embeddings can capture geometric relationships that represent semantic relations, as seen in examples like gender (man/woman, king/queen), verb tense (walking/walked, swimming/swam), and capital cities (Turkey/Ankara, Vietnam/Hanoi).
- Meaningful embedding spaces provide opportunities for machine learning systems to detect patterns that aid in the learning task.
- It's important to strike a balance in the number of dimensions in the embedding space. It should be large enough to encode rich semantic relations but small enough to allow for efficient training. A useful embedding may have on the order of hundreds of dimensions, which is much smaller than the size of the vocabulary for tasks like natural language processing.

Obtaining Embeddings

- Principal Component Analysis (PCA) is one of the mathematical techniques used for dimensionality reduction, which can be employed to create embeddings for a machine

learning system.

- Word2vec is an algorithm developed by Google for training word embeddings. It leverages the distributional hypothesis, which suggests that words with similar neighboring words tend to have semantic similarity.
- The distributional hypothesis is based on the idea that words appearing in proximity with similar neighboring words are semantically related.
- Word2vec uses a neural network to distinguish actual co-occurring groups of words from randomly grouped ones. It takes a sparse representation of a target word and one or more context words as input.
- Word2vec can create embeddings by either substituting a random noise word for the target word to create a negative example, or by pairing the true target word with randomly chosen context words.
- The main goal of Word2vec is to obtain an embedding, not necessarily to build a classifier. The weights connecting the input layer to the hidden layer are used to map sparse representations of words to smaller vectors.
- An alternative approach is to learn an embedding as part of the neural network for the target task. This results in a customized embedding for the specific system, but may take longer than training the embedding separately.
- An embedding layer can be created as a special type of hidden unit of size "d" in a neural network, and can be combined with other features and hidden layers.
- Learning a d-dimensional embedding maps each item to a point in a d-dimensional space, where similar items are located nearby. The edge weights between an input node and the nodes in the d-dimensional embedding layer correspond to the coordinate values for each of the d axes.