Ex-8 Lab Manual: Implement One-Hot Encoding of Words or Characters

---

1. Objective

To understand and implement **one-hot encoding**, a method used to represent categorical data (words or characters) in a numerical format suitable for machine learning and natural language processing (NLP).

2. Introduction

One-hot encoding is a technique used to convert words or characters into a binary vector representation. Each unique word or character is assigned a unique index, and its corresponding representation contains a **1** at the assigned index and **0s** elsewhere.

Example:

For a vocabulary of three words: ['cat', 'dog', 'fish'], their one-hot encoded vectors could be:

- cat → [1, 0, 0]
- dog → [0, 1, 0]
- fish → [0, 0, 1]

3. System Requirements

Hardware:

- Computer with **at least 4GB RAM** (8GB recommended)
- CPU/GPU support for faster computation (optional)

Software:

- Python (>=3.6)
- NumPy
- TensorFlow or Keras (for alternative implementation)

4. Step-by-Step Procedure

Step 1: Install Required Libraries

Ensure you have Python installed along with the necessary libraries. Install missing packages using:

```
pip install numpy tensorflow
```

Step 2: Import Required Libraries

```
import numpy as np

from tensorflow.keras.preprocessing.text import Tokenizer
```

Step 3: Define Sample Data

We define a small dataset of words or sentences for encoding:

```
sentences = [“I love machine learning”, “Deep learning is powerful”]
```

Step 4: Tokenization and Word Indexing

Use Tokenizer from Keras to assign unique integer indices to words:

```
tokenizer = Tokenizer()

tokenizer.fit_on_texts(sentences)

word_index = tokenizer.word_index

print(“Word Index:”, word_index)
```

Example Output:

```
{‘I’: 1, ‘love’: 2, ‘machine’: 3, ‘learning’: 4, ‘deep’: 5, ‘is’: 6, ‘powerful’: 7}
```

Step 5: Convert Words to One-Hot Encoding

Manually create one-hot encoded vectors using NumPy:

```
def one_hot_encode(word, word_index, vocab_size):

  encoding = np.zeros(vocab_size)

  encoding[word_index[word] – 1] = 1  # Subtract 1 for zero-based indexing

  return encoding


vocab_size = len(word_index)

encoded_words = {word: one_hot_encode(word, word_index, vocab_size) for word in word_index}


print(“One-Hot Encodings:”)

for word, encoding in encoded_words.items():

  print(f”{word}: {encoding}”)
```

Step 6: One-Hot Encoding Using Keras

Alternatively, Keras provides an automatic one-hot encoding function:

```
one_hot_results = tokenizer.texts_to_matrix(sentences, mode='binary')
print("One-Hot Encoded Matrix:")
print(one_hot_results)
```

5. Observations and Results

- Observe how each word is represented as a **binary vector**.
- Check how different words have **unique positions** in the encoded vector.

6. Troubleshooting

- **If you see all zeros in the output:** Ensure that words are correctly indexed.
- **If a word is missing:** Check that it is present in the word_index dictionary.
- **If dimensions mismatch:** Ensure vector size matches vocabulary size.

7. Additional Tasks

- Implement one-hot encoding for **characters instead of words**.
- Extend the vocabulary to include **new words dynamically**.
- Compare one-hot encoding with **word embeddings (e.g., Word2Vec, GloVe)**.

8. Conclusion

One-hot encoding is a foundational technique in NLP for representing text numerically. However, it suffers from **high-dimensionality** for large vocabularies, leading to the development of more efficient word embeddings.

End of Lab Manual

What is One-Hot Encoding: A Detailed Explanation

1. Introduction

One-hot encoding is a technique used to convert categorical data into a numerical format, making it suitable for machine learning and deep learning models. It is commonly used in **Natural Language Processing (NLP)** and **machine learning** tasks where algorithms require numerical input rather than raw text.

---

2. What is One-Hot Encoding?

One-hot encoding represents each category (word, character, or label) as a unique **binary vector** where:

- **Only one position has a value of 1** (indicating the presence of that category).
- **All other positions have a value of 0**.

This method ensures that categorical data can be efficiently processed by algorithms.

---

3. Example of One-Hot Encoding

3.1 One-Hot Encoding of Words

Suppose we have a vocabulary of three words:

["cat", "dog", "fish"]

Their corresponding one-hot vectors could be:

Word	One-Hot Encoding
cat	[1, 0, 0]
dog	[0, 1, 0]
fish	[0, 0, 1]

Each word is uniquely represented as a vector of the same length as the vocabulary size.

3.2 One-Hot Encoding of Characters

Consider encoding characters in the word "HELLO":

Character	One-Hot Encoding
H	[1, 0, 0, 0, 0]
E	[0, 1, 0, 0, 0]
L	[0, 0, 1, 0, 0]
O	[0, 0, 0, 0, 1]

Here, each unique character in "HELLO" is assigned a binary vector.

3.3 One-Hot Encoding of Labels (Categorical Data)

In classification tasks, labels can also be one-hot encoded.
For example, if we have three classes: ['Apple', 'Banana', 'Orange'], their one-hot representations would be:

Class	One-Hot Encoding
Apple	[1, 0, 0]
Banana	[0, 1, 0]
Orange	[0, 0, 1]

---

4. Why Use One-Hot Encoding?

One-hot encoding is used because **machine learning models require numerical input** rather than categorical data.

5. Advantages of One-Hot Encoding

✅ **Easy to interpret** – The binary vectors are simple and clear.
✅ **Compatible with most ML algorithms** – Many machine learning models require numerical input.
✅ **Eliminates Ordinal Relationships** – Unlike label encoding, one-hot encoding does not imply any ranking between categories.

---

6. Disadvantages of One-Hot Encoding

❌ **High Dimensionality** – If the dataset has a large number of unique words or categories, the vector size becomes **very large** (sparse representation).
❌ **Increased Memory Usage** – More dimensions mean more storage and computational power is required.

---

7. Implementing One-Hot Encoding in Python

We can implement one-hot encoding using **NumPy** or **Keras**.

7.1 Using NumPy

```
import numpy as np

# Define a vocabulary

vocab = ["cat", "dog", "fish"]

# Create a mapping

word_to_index = {word: i for i, word in enumerate(vocab)}

# Function for one-hot encoding

def one_hot_encode(word, vocab_size):

    vector = np.zeros(vocab_size)

    vector[word_to_index[word]] = 1

    return vector

# Encode 'cat'

print(one_hot_encode("cat", len(vocab)))
```

Output:

csharp

CopyEdit

[1. 0. 0.]

7.2 Using TensorFlow/Keras

```
from tensorflow.keras.preprocessing.text import Tokenizer


# Define sample sentences

sentences = ["I love coding", "Machine learning is amazing"]


# Tokenize and encode

tokenizer = Tokenizer()

tokenizer.fit_on_texts(sentences)

word_index = tokenizer.word_index


# Convert text to one-hot encoding

one_hot_results = tokenizer.texts_to_matrix(sentences, mode='binary')

print(one_hot_results)
```

---

8. Alternatives to One-Hot Encoding

One-hot encoding is **not always the best approach** due to its high dimensionality. Some alternatives include:

- **Label Encoding** – Assigns a unique integer to each category but introduces ordinal relationships.
- **Word Embeddings (e.g., Word2Vec, GloVe, BERT)** – Captures semantic relationships between words.
- **TF-IDF (Term Frequency-Inverse Document Frequency)** – Assigns importance to words based on frequency.

---

9. In short Answer/Conclusion

One-hot encoding is a fundamental technique for representing categorical data numerically. While it is simple and effective, it suffers from high dimensionality. For larger datasets, alternative methods like **word embeddings** are preferred.