DEEP LEARNING

BY M.J.PASSLAR

COMPUTERONIC – TEHRAN – IRAN

2022

CHAPTER 8 : TEXT AND SENTENCES IN DEEP LEARNING

What's data in this case ?

"DNN is a type of machine learning that mimics the way the brain learns"

Sentence

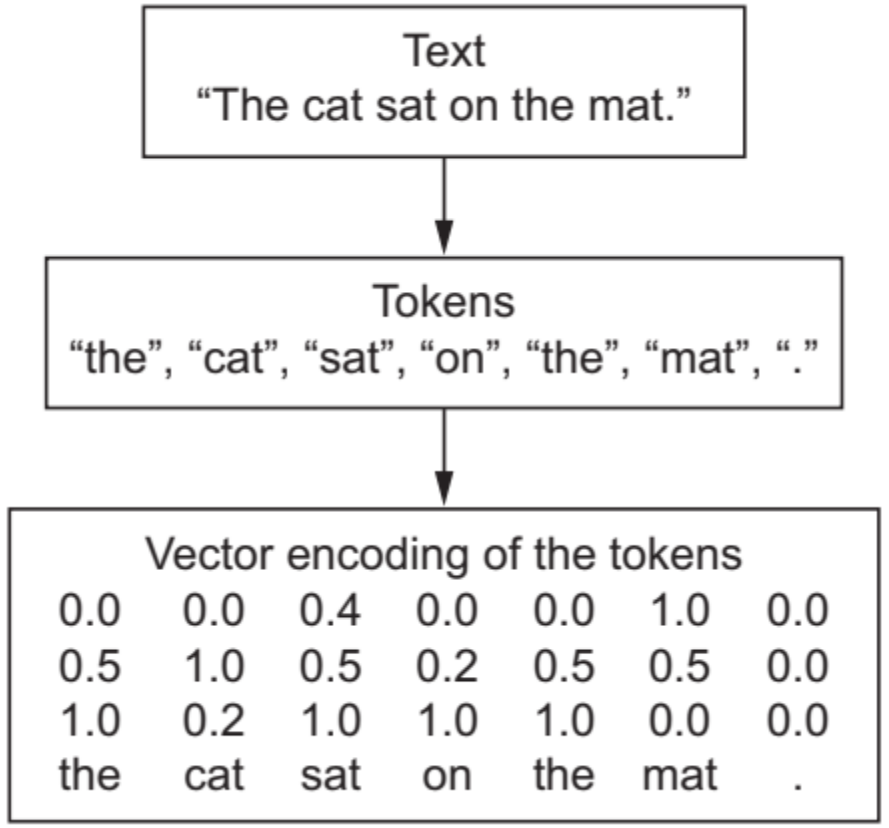
['DNN', 'is', 'a', 'type', 'of', 'machine', 'learning', 'that', 'mimics', 'the', 'way', 'the', 'brain', 'learns']

Words

['D','N','N'],['i','s'],['a'],['t','y','p','e'] , ...

Chars

Some definition ...



Row data

Tokens : spilite sentence to words/chars

Vectorizing(encoding) : assign floats number to words/chars

Encoding methods (One-Hot encoding)

- One-hot encoding
  - The most basic method
  - The most common method
  - Word level / character level

Assign index to words → Make array of size N → Make data set of N * data

One-hot word vectors:
- Sparse
- High-dimensional
- Hardcoded

Word-level one-hot encoding

```
import numpy as np
samples = ['The cat sat on the mat.', 'The dog ate my homework.']
token_index = {}
for sample in samples:
    for word in sample.split():
        if word not in token_index:
            token_index[word] = len(token_index) + 1

max_length = 10
results = np.zeros(shape=(len(samples),
max_length,
max(token_index.values()) + 1))

for i, sample in enumerate(samples):
    print(i,sample)
    for j, word in list(enumerate(sample.split()))[:max_length]:
        print(j,word)
        index = token_index.get(word)
        results[i, j, index] = 1
```

Character-level one-hot encoding

```
import string
import numpy as np
samples = ['The cat sat on the mat.', 'The dog ate my homework.']
characters = string.printable
token_index = dict(zip(range(1, len(characters) + 1), characters))
max_length = 50
results = np.zeros((len(samples), max_length, max(token_index.keys()) + 1))
for i, sample in enumerate(samples):
    print(i,sample)
    for j, character in enumerate(sample):
        print(j,character)
        index = token_index.get(character)
        results[i, j, index] = 1
```

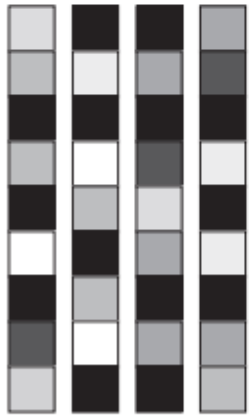
Using keras encoding tools

```
from keras.preprocessing.text import Tokenizer
samples = ['The cat sat on the mat.', 'The dog ate my homework.']
tokenizer = Tokenizer(num_words=1000)
tokenizer.fit_on_texts(samples)
sequences = tokenizer.texts_to_sequences(samples)
one_hot_results = tokenizer.texts_to_matrix(samples, mode='binary')
word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))
```

Encoding method(Word-embedding encoding)

- Word-embedding encoding
  - Smaller vector
  - Use float number instead binary
  - Word level / character level

Assign index to words → Make array of size N → Make data set of N * data



Word embeddings:
- Dense
- Lower-dimensional
- Learned from data

IMDB classification – preparing data

```
from keras.datasets import imdb
from keras import preprocessing
from tensorflow.keras.preprocessing.sequence import pad_sequences

max_features = 10000
maxlen = 20
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
x_train = pad_sequences(x_train, maxlen=maxlen)
x_test = pad_sequences(x_test, maxlen=maxlen)
```

IMDB classification – make model

```
model = Sequential()
model.add(Embedding(10000, 8, input_length=maxlen))
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
model.summary()
history = model.fit(x_train, y_train,
                    epochs=10,
                    batch_size=32,
                    validation_split=0.2)
```

IMDB classification – plot the result

```
import matplotlib.pyplot as plt
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```