ニューラルネットワークでのデータ表現

19nm705x オウ　ヨウケイコ

テンソルとは、線形的な量または線形的な幾何概念を一般化したもので、基底を選べば、多次元の配列として表現できるようなものであります。しかし、テンソル自身は、特定の座標系によらないで定まる対象であります。個々のテンソルについて、対応する量を記述するのに必要な配列の添字の組の数は、そのテンソルの階数とよばれます。

数値を一つしか含んでいないテンソルは、スカラーと呼ばれます。Numpyでテンソルの軸の数を表示するには、ndim属性を使用します。スカラーテンソルの軸の数は0です(ndim=0)。

```
import numpy as np
x=np.array(12)
print(x)
```

```
12
```

```
print(x.ndim)
```

```
0
```

数値の配列はベクトルと呼ばれます。ベクトルは1次元テンソルです。1次元テンソルの軸はちょっど一つです。

```
x=np.array([12,3,6,14,7])
print(x)
```

```
[12  3  6 14  7]
```

```
print(x.ndim)
```

```
1
```

ベクトルの配列は行列です。行列は、行と列の二つの軸を持つため、2次元テンソルです。一つ目の軸の要素を行、二つ目の軸の要素を列と呼びます。

```
x=np.array([[5,66,23,9,0],
            [6,79,3,21,5],
             [56,7,32,44,12]])
print(x.ndim)
```

```
2
```

Numpyの３次元テンソルを見てみましょう。

```
x=np.array([[[5, 66, 23, 9, 0],
             [6, 79, 3, 21, 5],
             [56, 7, 32, 44, 12]],
            [[5, 66, 23, 9, 0],
             [6, 79, 3, 21, 5],
             [56, 7, 32, 44, 12]],
            [[5, 66, 23, 9, 0],
             [6, 79, 3, 21, 5],
             [56, 7, 32, 44, 12]]])
print(x.ndim)
```

```
3
```

テンソルは、次に示す3つの主な属性によって定義されます。

- 軸の数(階数)

　　例えば、3次元テンソルの軸は3つであり、行列の軸は二つです。Numpyなどの Pythonライブラリでは、軸の数をテンソルのndim属性とも呼びます。

- 形状

　　テンソルの各軸に沿った次元の数を表す整数のタプル。

- データ型

　　テンソルに含まれているデータの型。Pythonライブラリでは、通常はdtypeで表わされます。例えば、テンソルの型はfloat32、unit8、float64などになります。

これらの属性をより具体的に理解するために、MNISTの例で処理したデータをもう一度見てみましょう。まず、MNISTデータセットを読み込みます。

```
from keras.datasets import mnist
(train_images,train_labels),(test_images,test_labels)=mnist.load_data()
print(train_images.ndim)
```

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 14s 1us/step
3
```

```
print(train_images.shape)
```

```
(60000, 28, 28)
```

```
print(train_images.dtype)
```

```
uint8
```

テンソルの特定の要素を選択することをテンソル分解と呼びます。Numpy配列で実行可能なテンソルを調べてみましょう。
次の例では、10番目から100番目の手前までの数字を選択し、それらを(90,28,28)という形状の配列に配置します。

```
my_slice=train_images[10:100]
print(my_slice.shape)
```

```
(90, 28, 28)
```

```
my_slice=train_images[10:100,:,:,]
print(my_slice.shape)
```

```
(90, 28, 28)
```

コロン(:)は軸全体を選択することに相当します。

一般に、deeplearningでしようされるデータテンソルの最初の軸は、サンプル軸になります。サンプル軸はサンプル次元とも呼ばれます。

それに加えて、deeplearningのモデルは、データセット全体を一度に処理するのではなく、データを小さなバッチに分割します。具体的には、このMNISTデータセットのバッチサイズは128なので、一つ目のバッチは次のように定義されます。

```
batch=train_images[:128]
```

二つ目のバッチは次のように定義されます。

```
batch=train_images[128:256]
```

そしてn番目のバッチは次のようになります。

```
batch=train_images[128*n:128*(n+1)]
```

ベクトルデータは最も一般的なデータです。ベクトルデータの例を挙げておきます。

● 人の年齢、郵便番号と住所、収入をまとめた生命表データセット

　人をそれぞれ3つの値からなるベクトルとして特徴付けることができます。したがって、10万人分のデータセット全体を、形状が(100000,3)の2次元テンソルに格納できます。

データにおいて時間が重要となる場合は、常に、そうしたデータを明示的な時間軸をもつ3次元テンソルに格納するのが理にかなっています。
時間軸は常に二つ目の軸にするのが慣例となっています。次に、例を挙げておきます。

● 株価のデータセット

　現在の株価と1分間の最高値と最安値が1分おきに格納されます。したがって、1分間のデータは3次元ベクトルとしてエンコードされます。1日の取引時間を390分とすれば、1日分の株取引は形状が(390,3)の2次元テンソルとしてエンコードされます。250日分のデータは形状が(250,393,3)の3次元テンソルに格納できます。この場合、各サンプルは1日分のデータとなります。

一般に、画像は幅、高さ、色深度の3つの次元で表わされます。画像テンソルの形状には、Theanoが採用しているチャネルファースト方式とTensorflowが採用しているチャネルラスト方式の2つの規約があります。

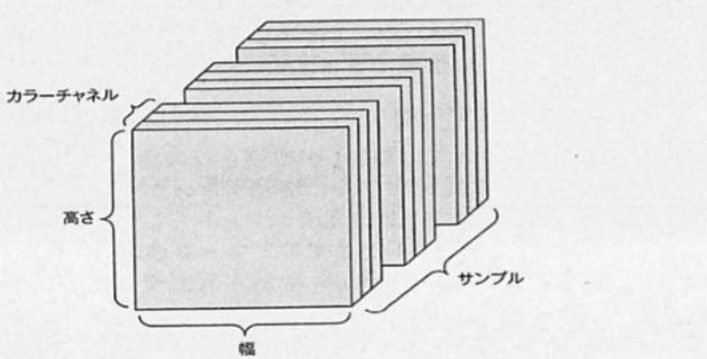

図 2-4：画像データの 4 次元テンソル（チャネルファースト方式）

5次元テンソルが必要となる現実のデータの種類は限られていますが、動画データはそのうちの1つです。各フレームをカラー画像とすれば、動画をフレームのシーケンスとして考えることができます。各フレームは形状が(height,width,color_depth)の3次元テンソル格納できるため、フレームのシーケンス形状が(frames,height,width,color_depth)の4次元テンソルに格納できます。したがって、さまざまな動画が含まれたバッチは、形状が(samples,frames,height,width,color_depth)の5次元テンソルに格納できます。