2.2ニューラルネットワークでのデータ表現

テンソルとは

・任意の次元に対し、行列を一般化したもので
　データ（数値）の入れ物

例：Numpy配列に格納されたデータ

任意の次元に対し、行列を一般化したもの
…?

テンソルとは、ｎ次元のデータの入れ物

テンソルとは

語彙
・テンソルの軸とは
　→テンソルの次元、軸０、軸１と数える

・テンソルの階数（rank）とは
　→軸の数

テンソルの具体例

・０次元テンソル（スカラー）
・１次元テンソル
・２次元テンソル
について説明する

テンソルの具体例
・０次元テンソル（スカラー）

・軸とは
→テンソルの次元
・階数（rank）とは
→軸の数

点（０次元）

```
import numpy as np
x=np.array(5) #これが0次元テンソル
print(x)
print(x.ndim) #階数を表示
```

実行結果
5
0

つまり、ｘは０次元テンソルで、階数０のデータの入れ物

テンソルの具体例
・１次元テンソル

・軸とは
→テンソルの次元
・階数（rank）とは
→軸の数

線（１次元）

```
import numpy as np
x=np.array([3,4]) #これが1次元テンソル
print(x)
print(x.ndim) #階数を表示
```

実行結果

[3 4]

1

つまり、ｘは１次元テンソルで、階数１のデータの入れ物

テンソルの具体例
・２次元テンソル

・軸とは
→テンソルの次元
・階数（rank）とは
→軸の数

平面（２次元）

```
import numpy as np
x=np.array([[2,3,4,5],
            [5,8,7,9],
            [2,4,1,7]]) #これが2次元テンソル
print(x)
print(x.ndim) #階数を表示
```

実行結果

```
[[2 3 4 5]
 [5 8 7 9]
 [2 4 1 7]]
2
```

つまり、ｘは２次元テンソルで階数2のデータの入れ物

テンソルの呼び方

・階数**n**のテンソル
・ n 次元テンソルといった呼び方がある

n 次元テンソルが一般的

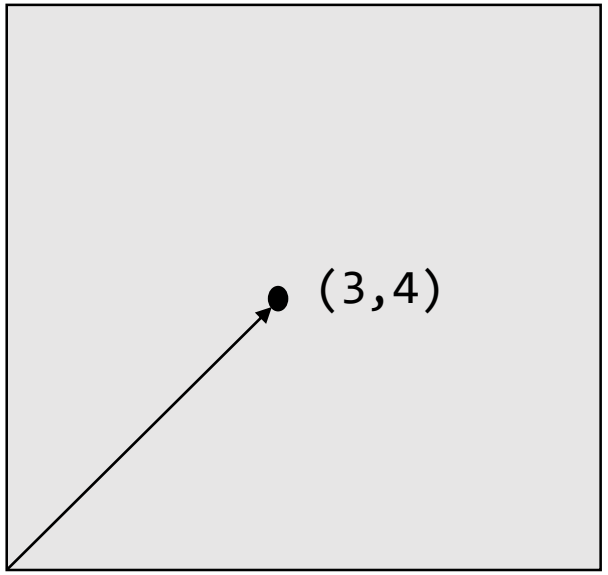
テンソルとは、 n 次元のデータの入れ物

テンソルの注意

※テンソルとベクトルは異なる

`x=np.array([3,4])`

は１次元テンソルで２次元ベクトルである



テンソルの重要な属性

・階数（rank）
・形状
・データ型
の３つがある

テンソルの重要な属性

- 階数

・軸とは
→テンソルの次元
・階数（rank）とは
→軸の数

何次元か

先ほどの例

```
import numpy as np
x=np.array([3,4]) #これが1次元テンソル
print(x)
print(x.ndim) #階数を表示
```

で、x.ndimによって階数を求めたように
PythonのNumpyなどではndim属性という

テンソルの重要な属性

・形状
　→次元の数を表すタプル

タプル
→データ型の１種

・行列
（[[2,3,4,5,6],
　[5,8,7,9,7],
　[2,4,1,7,8]]）
のテンソルの形状は(3,5)

・ベクトル
（[2,3,4,5,6]）
のテンソルの形状は(5,)

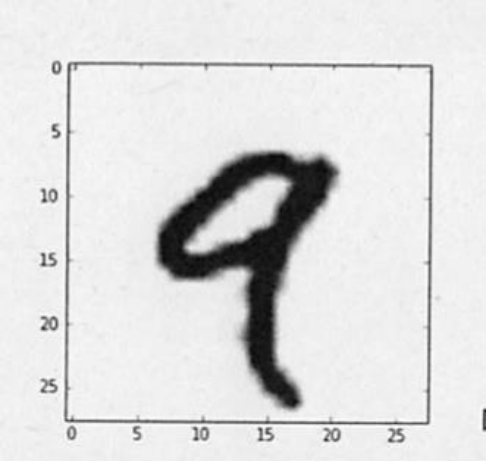
・スカラー
（5）
のテンソルの形状は()

テンソルの重要な属性

・データ型
　float32,unit8,float64といったデータの型のこと

　char型もあるがNumpyにはこの型のテンソルは無い

MNISTを使ったテンソルの実践



・MNIST
左図のようなサイズが28*28の手書きの数字のデータが６万個入ったデータセット（60000*28*28）

```
>>> print(train_images.ndim)
3
```

このテンソルの形状は次のとおりです。

```
>>> print(train_images.shape)
(60000, 28, 28)
```

このテンソルのデータ型（dtype 属性の値）は次のとおりです。

```
>>> print(train_images.dtype)
uint8
```

・階数（rank）
・形状
・データ型
を求めている

Numpyでのテンソルの操作

・MNISTの6万個のデータのうち10から100番目の手前までデータを取り出したい

```
>>> my_slice = train_images[10:100]
>>> my_slice.shape
(90, 28, 28)
```

形状が（90*28*28で表現される）ことより、取り出せていることが分かる。

この例のように、もとのデータは３次元なのに軸０しか指定しないと、他の軸はすべて取り出される。

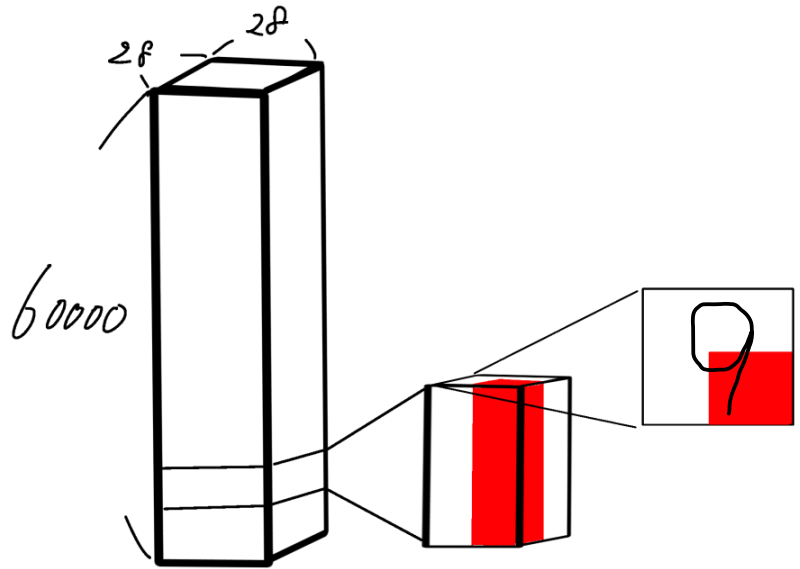
Numpyでのテンソルの操作

ルール

・[以上:より小さい]のように範囲を指定
・[10:20,7:14,7:14]のように各軸の範囲を指定
・さっきの例のように、指定する軸がテンソルの次元より
　少ないと指定より大きい次元はすべて取り出される。
・[:]のように範囲を指定しないとその軸すべてが範囲
・[:14]だと14より小さいもの　[14:]だと14以上すべてを指定

```
>>> # 先の例と同じ
>>> my_slice = train_images[10:100, :, :]
>>> my_slice.shape
(90, 28, 28)

>>> # これも先の例と同じ
>>> my_slice = train_images[10:100, 0:28, 0:28]
>>> my_slice.shape
(90, 28, 28)
```



Numpyでのテンソルの操作

・負のインデックスについて
Pythonのリストではマイナスのインデックスがあり最後尾からの順番を意味する

リスト
a=[3,12,45,56,29]
の-1番目は29

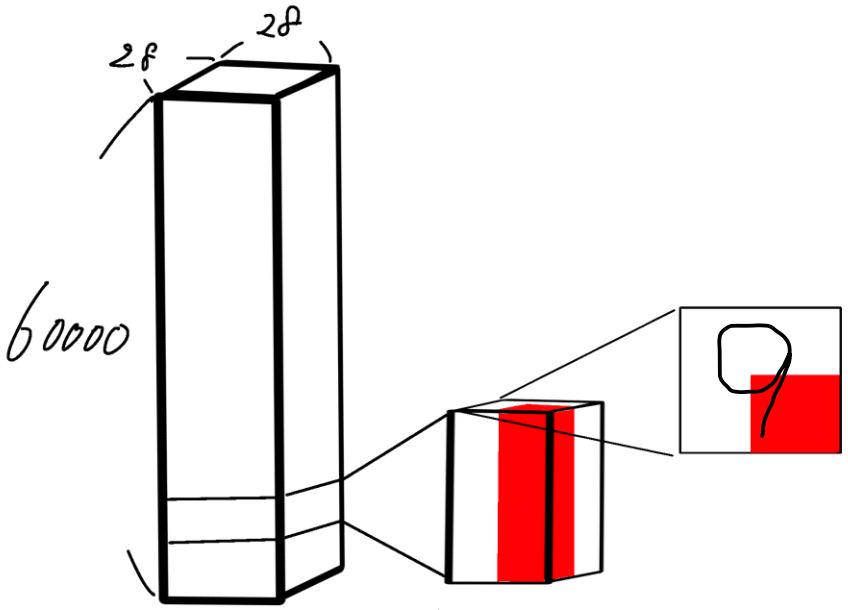
Numpyでのテンソルの操作でも同様の指定が可能

データバッチ

語彙

・サンプル軸

データテンソルの最初の軸（60000*28*28）の60000のこと



・バッチ

→一群、１まとまり

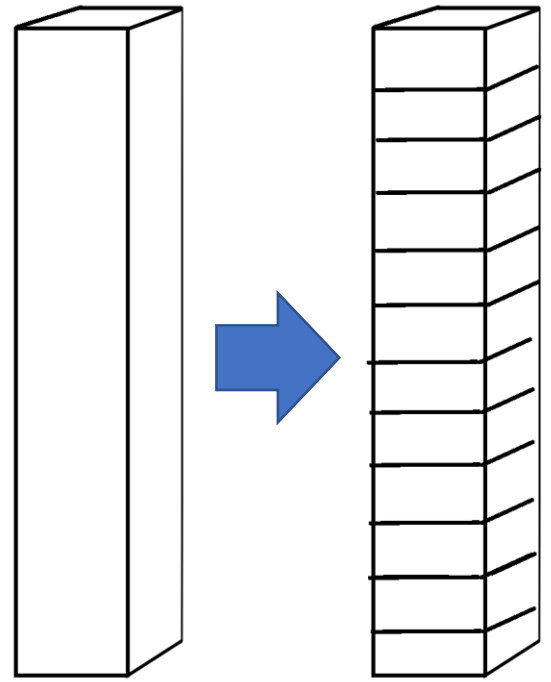
データバッチ

ディープラーニングではデータをバッチに分けて処理

batch=train_images[:128]
batch=train_images[128:256]
～
batch= train_images[128n:128(n+1)]

といったように分割

それぞれの最初の軸（軸０）をバッチ軸という



データテンソルの現実的な例

・ベクトルデータ
　（２次元テンソル）
・時系列データまたはシーケンスデータ
　（３次元テンソル）
・画像データ
　（４次元テンソル）
・動画データ
　（５次元テンソル）
の４つを紹介します。

データテンソルの現実的な例

・ベクトルデータ（２次元テンソル）
サンプル軸と特徴軸からなる

１０万人の年齢、郵便番号と住所、収入をまとめた生命表データセット
→形状が（100000,3）の２次元テンソルに格納

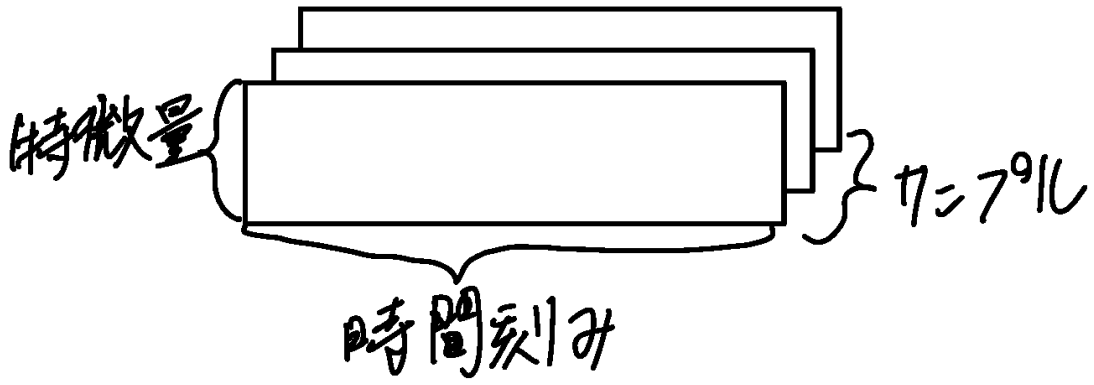
・テンソルの特性
階数→ｎ次元
形状→（3,5）とか
データ型→float32とか
・データバッチ
→分割されたデータの１まとまり

データテンソルの現実的な例

・テンソルの特性
階数→ｎ次元
形状→（3,5）とか
データ型→float32
とか
・データバッチ
→分割されたデータの１まとまり

・時系列データ（３次元テンソル）
サンプル軸、特徴量、時間刻みといったものからなる

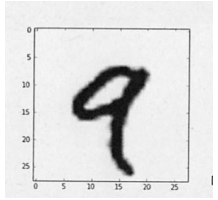
株価のデータセット
・３次元ベクトルからなる１分間のデータ
・ 390分が１日の取引時間
・250日間分のサンプル
→形状が（250,3,390）の
３次元テンソルに格納



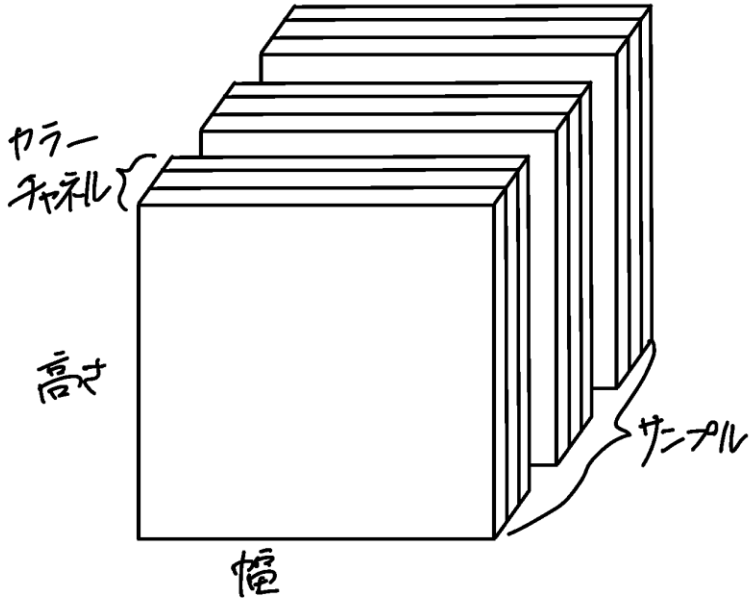
データテンソルの現実的な例

・画像データ（４次元テンソル）
サンプル軸、幅、高さ、色深度からなる

128個のサイズが256*256のカラー画像は形状が(128,256,256,3)の４次元テンソルに格納

MNIST  のようなグレースケール画像は３次元テンソルで表せるが慣習として（128,256,256,1）
といったようなテンソルに格納

・テンソルの特性
階数→ｎ次元
形状→（3,5）とか
データ型→float32
とか
・データバッチ
→分割されたデータの１まとまり



データテンソルの現実的な例

チャネルラストとチャネルファースト

画像テンソルの形状には
（samples,width,height,color_depth）
のチャネルラストと
（samples, color_depth ,width,height）
のチャネルファースト
二種類がある

データテンソルの現実的な例

・動画データ（５次元テンソル）
動画は少しずつ変化する沢山の画像からなり、
１枚１枚をフレームと呼ぶ
動画データは
サンプル軸、フレーム、幅、高さ、色深度からなる

60秒間の144*256のビデオクリップを４fps（flames per second）でサンプリングするとフレーム数は240になり、このようなビデオクリップ４つのバッチは、形状が（4,240,144,256,3）の５次元テンソルに格納

・テンソルの特性
階数→ｎ次元
形状→（3,5）とか
データ型→float32とか
・データバッチ
→分割されたデータの１まとまり