3.2 線形代数

- 線形代数を用いれば，多くの数値に対する処理を，簡潔な数式で記述することができる.
- 線形代数における数値のまとめ方の概念には，スカラー，ベクトル，行列，テンソルがある.

3.2.1 スカラー

- スカラーは，1，5，12，-7などの通常の数値のこと.

3.2.2 ベクトル

- ベクトルは，スカラーを直線上に並べたもの.
- ベクトルは，NumPyの１次元配列を用いて，以下のように表すことができる：

```
import numpy as np

a = np.array([1, 2, 3])
b = np.array([-2.3, 0.25, -1.2, 1.8, 0.41])
```

3.2.3 行列

- 行列は，スカラーを格子状に並べたもので，例えば以下のように表記する：

$$\begin{pmatrix} 0.12 & -0.34 & 1.3 & 0.81 \\ -1.4 & 0.25 & 0.69 & -0.41 \\ 0.25 & -1.5 & -0.15 & 1.1 \end{pmatrix}.$$

- 行列において，水平方向のスカラーの並びを行，垂直方向のスカラーの並びを列という.

- NumPyの２次元配列を用いると，例えば以下のように行列を表現することができる：

```
import numpy as np

# 2×3の行列
a = np.array([[1, 2, 3],
              [4, 5, 6]])

# 3×2の行列
b = np.array([[0.21, 0.14],
              [-1.3, 0.81],
              [0.12, -2.1]])
```

- ディープラーニングで行われる演算は，大部分が行列同士の演算.

3.2.4 テンソル

- テンソルはスカラーを複数の次元に並べたもので，スカラー，ベクトル，行列を含む．
- 各要素につく添字の数を，そのテンソルの階数という.
- スカラーには添字がないので0階のテンソル，ベクトルは添字が1つなので1階のテンソル，行列は添字が2つなので2階のテンソルとなる．
- より高次元なものは，3階のテンソル，4階のテンソル...となる.

- NumPyの多次元配列を用いると，例えば次のように3階のテンソルを表現することができる：

```
import numpy as np

# (2, 3, 4)の3階のテンソル
a = np.array([[[0, 1, 2, 3],
               [2, 3, 4, 5],
               [4, 5, 6, 7]],
              [[1, 2, 3, 4],
               [3, 4, 5, 6],
               [5, 6, 7, 8]]])

print(a.shape)
# (2, 3, 4)
```

- このようなテンソルの形状は， `reshape` メソッドで自由に変換することができる.
- 以下の例では， `reshape` の引数にテンソルの形状を渡して，形状の変換をしている：

```
# 1階のテンソル(ベクトル)
b = np.array([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24])
print(b.shape)
# (24,)


# 2階のテンソル(行列)
b = b.reshape(4, 6)
print(b)
# [[ 1  2  3  4  5  6]
#  [ 7  8  9 10 11 12]
#  [13 14 15 16 17 18]
#  [19 20 21 22 23 24]]

print(b.shape)
# (4, 6)


# 3階のテンソル
b= b.reshape(2, 3, 4)
print(b)
# [[[ 1  2  3  4]
#   [ 5  6  7  8]
#   [ 9 10 11 12]]

#  [[13 14 15 16]
#   [17 18 19 20]
#   [21 22 23 24]]]

print(b.shape)
# (2, 3, 4)


# 4階のテンソル
b = b.reshape(2, 2, 3, 2)
print(b)
# [[[[ 1  2]
#    [ 3  4]
#    [ 5  6]]

#   [[ 7  8]
#    [ 9 10]
#    [11 12]]]


#  [[[13 14]
#    [15 16]
#    [17 18]]

#   [[19 20]
#    [21 22]
#    [23 24]]]]
```

```
print(b.shape)
# (2, 2, 3, 2)
```

- このように，要素数さえ一致していれば，`reshape` により自由にテンソルの形状を変換することができる．

3.2.5 スカラーと行列の積

3.2.6 要素ごとの積

3.2.7 行列積

3.2.8 行列の転置