Markdownの数式の書き方

輪読 レポート課題

入力 X，重み W とする全結合層について，順伝播の出力 Y を以下のように定義する.

$$
\begin{aligned}
X &= \begin{pmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \end{pmatrix} \\
W &= \begin{pmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \end{pmatrix} \\
Y &= XW
\end{aligned}
$$

問1

X と W の要素を用いて，Y の全要素を求めよ.

$$
Y = \begin{pmatrix} ? & ? & ? \\ ? & ? & ? \end{pmatrix}
$$

問2

損失関数 L とすると，Yの各要素についての勾配を書け．ただし，L はスカラーとする.

$$
\frac{\partial L}{\partial Y} = \begin{pmatrix} ? & ? & ? \\ ? & ? & ? \end{pmatrix}
$$

問3

L を $x_{1,1}$ について微分せよ.

$$
\begin{aligned}
\frac{\partial L}{\partial x_{1,1}} &= \sum_{j=1}^{2}\sum_{i=1}^{3}\frac{\partial L}{\partial y_{j,i}}\frac{\partial y_{j,i}}{\partial x_{1,1}} \\
&= \cdots
\end{aligned}
$$

問4

L を それぞれ $x_{1,2}$， $x_{2,1}$， $x_{2,2}$ について微分せよ.

問5

$\frac{\partial L}{\partial X}$ を求めよ.

$$
\begin{aligned}
\frac{\partial L}{\partial X} &= \begin{pmatrix} \frac{\partial L}{\partial x_{1,1}} & \frac{\partial L}{\partial x_{1,2}} \\ \frac{\partial L}{\partial x_{2,1}} & \frac{\partial L}{\partial x_{2,2}} \end{pmatrix} \\
&= \cdots (\text{ここを書け}) \\
&= \frac{\partial L}{\partial Y} W^T
\end{aligned}
$$

問6

$\frac{\partial L}{\partial W} = X^T \frac{\partial L}{\partial Y}$ を示せ.

問7

参考ソースコードの空欄を埋めて，二つの全結合層とReLUからなるネットワークについて,

1. 順伝播
2. 逆伝播
3. 勾配を用いた重みの更新

を実装せよ.

- 入力 $X \in \mathbb{R}^{64,1000}$（$64 \times 1000$ の行列）．64個のデータがあり，それぞれは

1000 次元
- 真値 $T \in \mathbb{R}^{64\times 10}$．あるひとつのデータにの真値は 10 次元である.
- 重み $W_1 \in \mathbb{R}^{1000\times 100}$，$W_2 \in \mathbb{R}^{100\times 10}$ とする.
- 損失関数は自乗誤差，順伝播は下記とする.

$$
\begin{aligned}
Y_1 &= XW_1 \\
Y_2 &= \mathrm{ReLU}(Y_1) \\
Y_{\mathrm{pred}} &= Y_2 W_2 \\
loss &= \sum (y_{\mathrm{pred}} - t)^2
\end{aligned}
$$

参考ソースコード(1)：

```
import numpy as np

# For reproducibility
rng = np.random.default_rng(0)

# N is batch size; D_in is input dimension;
# H is hidden dimension; D_out is output dimension.
N, D_in, H, D_out = 64, 1000, 100, 10
learning_rate = 1e-4

# Create random input data and ground truth
x = rng.standard_normal( (N,D_in) )
t = rng.standard_normal( (N, D_out) )

# Randomly initialize weights
w1 = ...
w2 = ...

for i in range(500):
    # Forward pass: compute predicted y
    y1 = ...
    y2 = ...
    y_pred = ...

    # Compute and print loss
    loss = ...
    print(i, loss)

    # Backprop to compute gradients of w1 and w2 with respect to loss
    grad_y_pred = ...
    grad_w2 = ...
    grad_y2 = ...
```

```
    grad_y1 = ...
    grad_w1 = ...

    # Update weights
    w1 = ...
    w2 = ...
```

出力例：

```
0 26365524.19345431
1 18539004.956771437
2 14019258.774463898
3 10761426.115604788
4 8169072.8958810605
5 6099576.78509928
6 4499933.628590755
7 3309552.8580611832
8 2448877.062830101
9 1836295.6953177932
...
490 9.853059833781279e-05
491 9.47769895507549e-05
492 9.116623972396843e-05
493 8.769287584912006e-05
494 8.435302599117529e-05
495 8.113959632491307e-05
496 7.804999208049452e-05
497 7.507747826896027e-05
498 7.22182049129895e-05
499 6.947019049376511e-05
```

問8

Cross Entropy損失を用いて，ニューラルネットを訓練せよ.

ネットワークの出力 $Y \in \mathbb{R}^{N \times 10}$，正解クラスのインデックス $i \in [0, 9]$ とすると，Cross Entropyは下記のように計算する.

$$loss = \sum_{n=1}^{N} -\log(\mathrm{softmax}(y_{n,i}))$$

$$\mathrm{softmax}(y) = \frac{e^{y}}{\sum_{j}^{10} e^{y_j}}$$

$$\frac{\partial}{\partial Y} loss = \begin{cases} \mathrm{softmax}(y_{n,j}) - 1 & \text{if} \quad j = i \\ \mathrm{softmax}(y_{n,j}) & \text{otherwise} \end{cases}$$

参考ソースコード(2)

```
import numpy as np

# For reproducibility
rng = np.random.default_rng(0)

# N is batch size; D_in is input dimension;
# H is hidden dimension; D_out is output dimension.
N, D_in, H, D_out = 64, 1000, 100, 10
learning_rate = 1e-4

# Create random input data
x = rng.standard_normal(N, D_in)


# Create discrete ground truth
class_idx = np.random.randint( 0, D_out - 1, N )  # ランダムに正解クラスを生成
t = np.zeros((N,D_out))  # 他クラスは0
t[range(N),class_idx] = 1  # 正解クラスは1

# Randomly initialize weights
w1 = ...
w2 = ...

for i in range(500):
    # Forward pass: compute predicted y
    y1 = ...
    y2 = ...
    y_pred = ...

    # Calculate cross entropy loss
    ## Apply softmax
    y_pred_softmax = ...

    ## Negative log likelihood
    loss = ...
```

```
    print(i, loss)

    # Backprop to compute gradients of w1 and w2 with respect to loss
    grad_y_pred = ...
    grad_w2 = ...
    grad_y2 = ...
    grad_y1 = ...
    grad_w1 = ...

    # Update weights
    w1 -= ...
    w2 -= ...
```

出力例

```
0 18577.325696960957
1 18569.038169633015
2 18560.815386999682
3 18552.63451574695
4 18544.349422965068
5 18536.130059624775
6 18527.911941126353
7 18519.703155128533
8 18511.481262949834
9 18503.262100480468
...
490 16507.88223835033
491 16500.883613169462
492 16493.892156589183
493 16486.90721282158
494 16479.930409234268
495 16472.96477376828
496 16466.009036097814
497 16459.0630991756
498 16452.1283421231
499 16445.20432192529
```