深層学習　day1 Section4 勾配降下法

Q)該当するソースコードを探しみよう

$w^{(t+1)} = w^{(t)} - \varepsilon \Delta E n$

$\Delta E = \frac{\partial E}{\partial W} = [\frac{\partial E}{\partial W1} \cdot \cdot \cdot \frac{\partial E}{\partial Wn}]$

ε ：学習率

A)

```
network[key]  -= learning_rate * grad[key]
```

```
grad = backward(x, d, z1, y)
###呼び出している関数は下記
# 誤差逆伝播
def backward(x, d, z1, y):
    # print("¥n##### 誤差逆伝播開始 #####")

    grad = {}

    W1, W2 = network['W1'], network['W2']
    b1, b2 = network['b1'], network['b2']

    # 出力層でのデルタ
    delta2 = functions.d_mean_squared_error(d, y)
    # b2 の勾配
    grad['b2'] = np.sum(delta2, axis=0)
    # W2 の勾配
    grad['W2'] = np.dot(z1.T, delta2)
    # 中間層でのデルタ
    #delta1 = np.dot(delta2, W2.T) * functions.d_relu(z1)

    ## 試してみよう
    delta1 = np.dot(delta2, W2.T) * functions.d_sigmoid(z1)

    delta1 = delta1[np.newaxis, :]
```

```
    # b1 の勾配
    grad['b1'] = np.sum(delta1, axis=0)
    x = x[np.newaxis, :]
    # W1 の勾配
    grad['W1'] = np.dot(x.T, delta1)

    # print_vec("偏微分_重み 1", grad["W1"])
    # print_vec("偏微分_重み 2", grad["W2"])
    # print_vec("偏微分_バイアス 1", grad["b1"])
    # print_vec("偏微分_バイアス 2", grad["b2"])

    return grad
```

###繰り返している箇所は下記

```
# 勾配降下の繰り返し
for dataset in random_datasets:
    x, d = dataset['x'], dataset['d']
    z1, y = forward(network, x)
    grad = backward(x, d, z1, y)
    # パラメータに勾配適用
    for key in ('W1', 'W2', 'b1', 'b2'):
        network[key]  -= learning_rate * grad[key]

    # 誤差
    loss = functions.mean_squared_error(d, y)
    losses.append(loss)
```

Q）オンライン学習とは何かまとめよ

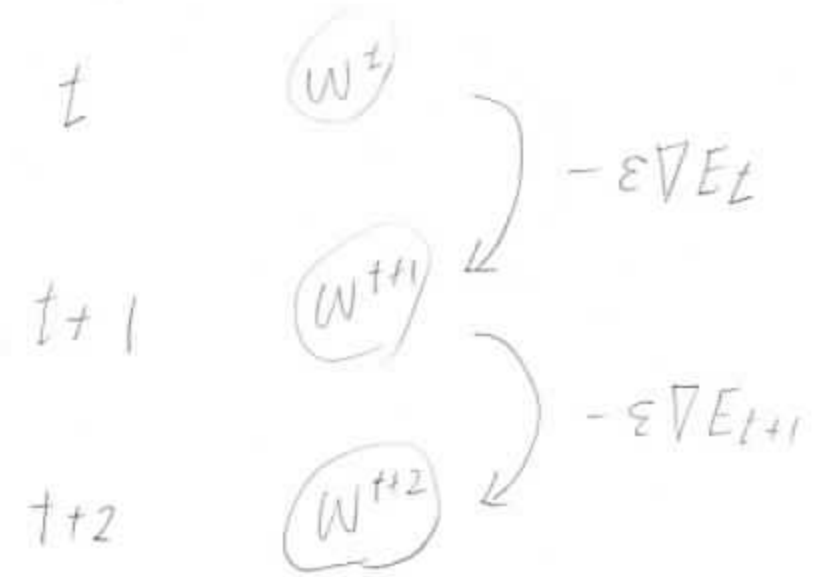
A)学習データが入ってくる度に都度パラメータ（重みw、バイアス値b）を更新し、都度学習を進めていく方法。一方、バッチ学習では一度にすべての学習データを使用してパラメータ更新を行う。

　最新の深層学習モデルでは、PC のメモリの制約回避等の理由でオンライン学習のほうが有力。

Q)この数式の意味を図に書いて説明せよ

$$w^{(t+1)} = w^{(t)} - \varepsilon \Delta E_n$$

A)


エポック
重み
t
W^t
$-\varepsilon\nabla E_t$
t+1
W^{t+1}
$-\varepsilon\nabla E_{t+1}$
t+2
W^{t+2}