深層学習 day3 section1

Q) 以下は再帰型NNにおいて構文木を入力として再帰的に文全体の表現ベクトルを得るPGである。ただし、NNの重みパラメータはグローバル変数として定義してあるものとし、_activation関数はなんらかの活性化関数であるとする。木構造は再帰的な辞書で定義してあり、rootが最も外側の辞書であると仮定する。(く)にあてはまるのは？

```
def traverse(node):
    if not isinstance(node, dict):
        v = node
    else:
        left = traverse(node['left'])
        right = traverse(node['right'])
        v = _activation(      (く)      )
    return v
```

A.) (2) W.dot(np.concatenate([left, right]))

Q) 下記の(お)にあてはまるのはどれか？

```
def bptt(xs, ys, W, U, V):
    # 順伝播
    hiddens, outputs = rnn.net(xs, W, U, V)
    # 損失関数のW, U, Vの偏微分
    dW = np.zeros_like(W)
    dU = np.zeros_like(U)
    dV = np.zeros_like(V)

    do = _calculate_do(outputs, ys)

    batch_size, n_seq = ys.shape[:2]
    # 時間を逆方向にたどりパラメータの偏微分値を計算
    for t in reversed(range(n_seq)):
        dV += np.dot(do[:, t].T, hiddens[:, t]) / batch_size
        delta_t = do[:, t].dot(V)
        # W, Uはさらに遡って計算
        for bptt_step in reversed(range(t+1)):
            dW += np.dot(delta_t.T, xs[:, bptt_step]) / batch_size
            dU += np.dot(delta_t.T, hiddens[:, bptt_step-1]) / batch_size
            delta_t = [ (お) ]

    return dW, dU, dV
```

損失関数を出力に関して偏微分した値を返す

batch_size

Ans) (2) delta_t.dot(U) ∵ 過去に遡るたびにUが掛けられる。