# １．　section1_再起型ニューラルネットワークの概念

## 演習チャレンジ

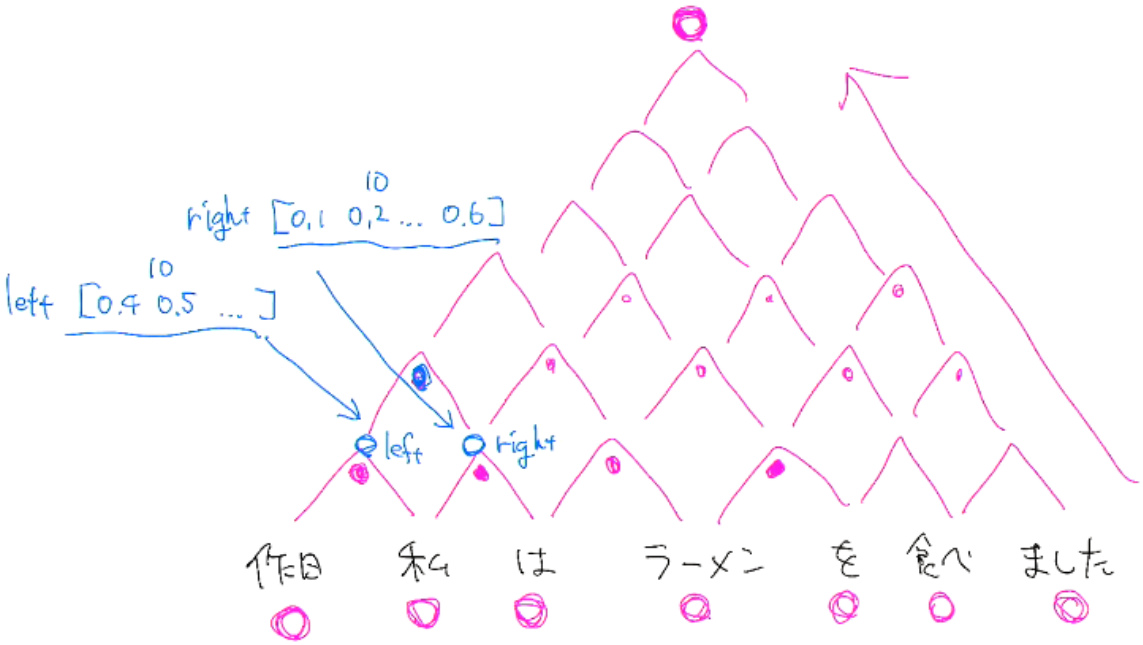
以下は再帰型ニューラルネットワークにおいて構文木を入力として再帰的に文全体の表現ベクトルを得るプログラムである。ただし、ニューラルネットワークの重みパラメータはグローバル変数として定義してあるものとし、_activation関数はなんらかの活性化関数であるとする。木構造は再帰的な辞書で定義してあり、rootが最も外側の辞書であると仮定する。
（<）にあてはまるのはどれか。

```
def traverse(node):
    """
    node: tree node, recursive dict, {left: node', right: node''}
          if leaf, word embeded vector, (embed_size,)

    W: weights, global variable, (embed_size, 2*embed_size)
    b: bias, global variable, (embed_size,)
    """
    if not isinstance(node, dict):
        v = node
    else:
        left = traverse(node['left'])
        right = traverse(node['right'])
        v = _activation(                    (<)                    )
    return v
```

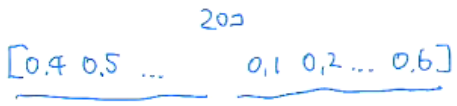
（解答）２

構文木



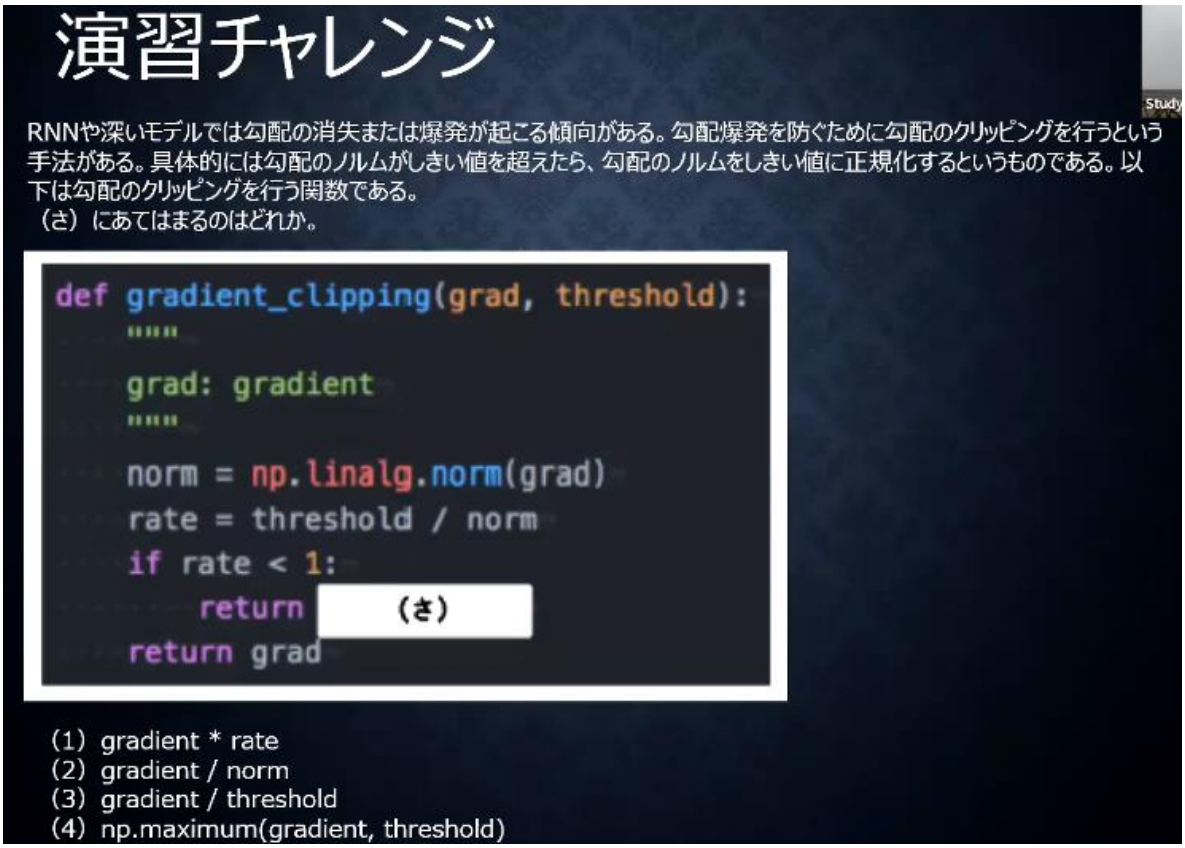

単語同士の特徴量

Concatenate




１０＋１０＝２０個になる－＞重みを掛ける。－＞１０個にする。

## ２．　section2_LSTM



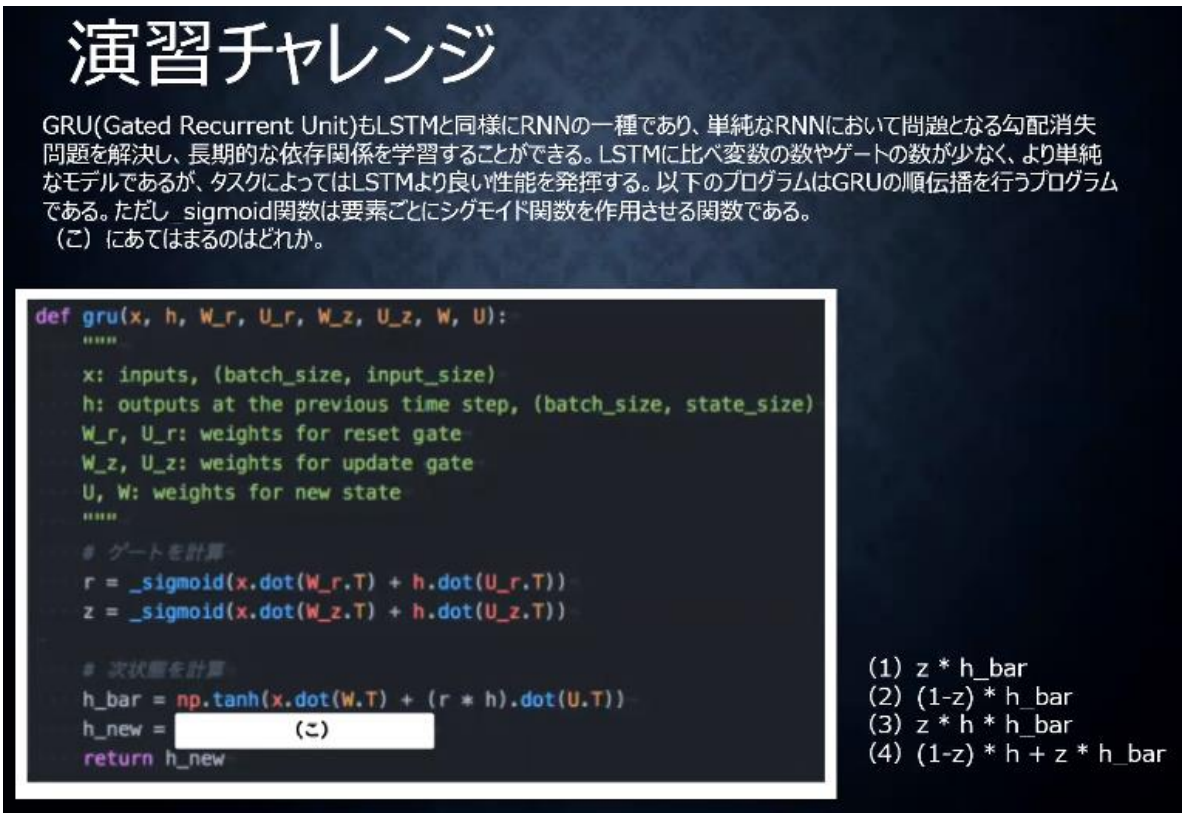

（解答）（１）

勾配のクリッピングを行うので rate を掛ける。


ＣＥＣに覚えさせる情報を説明している。

（解答）（３）

３．　section3_GRU




（解答）（４）

● LSTM と GRU の違いを簡潔に述べよ

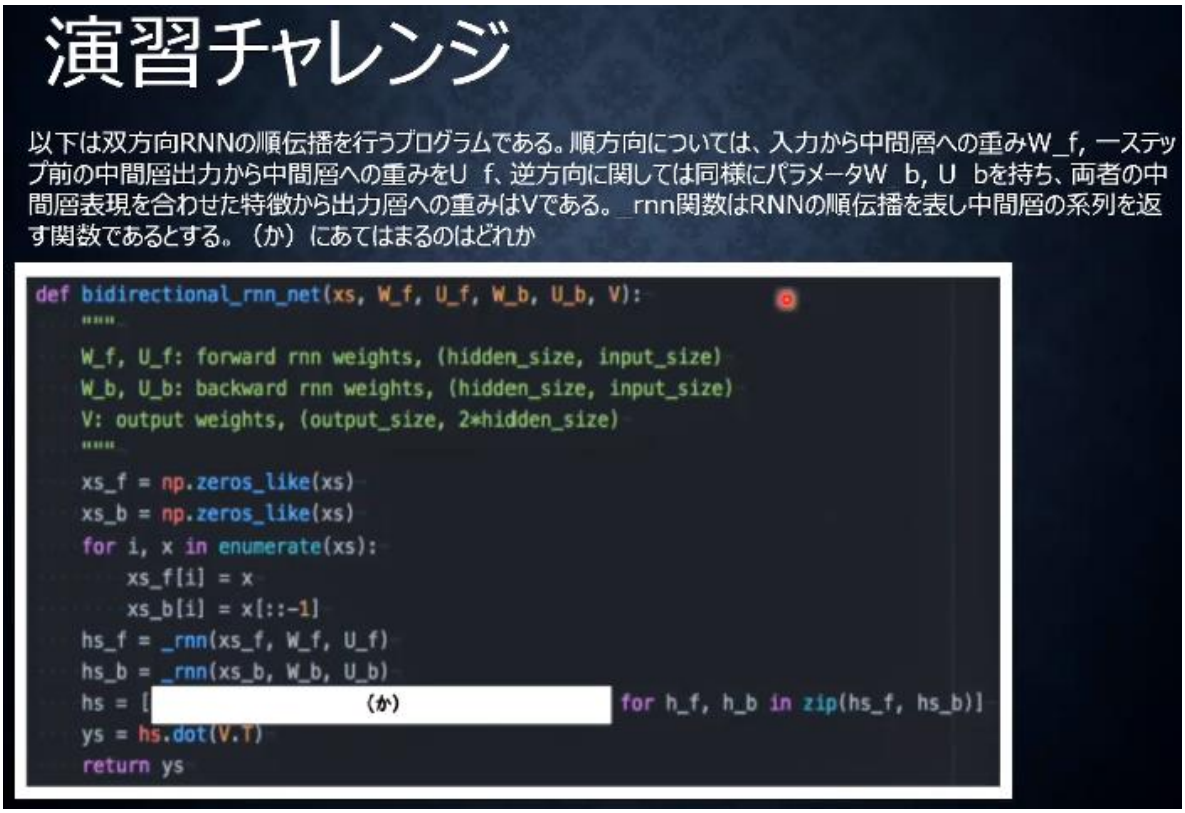
（１）ＬＳＴＭ

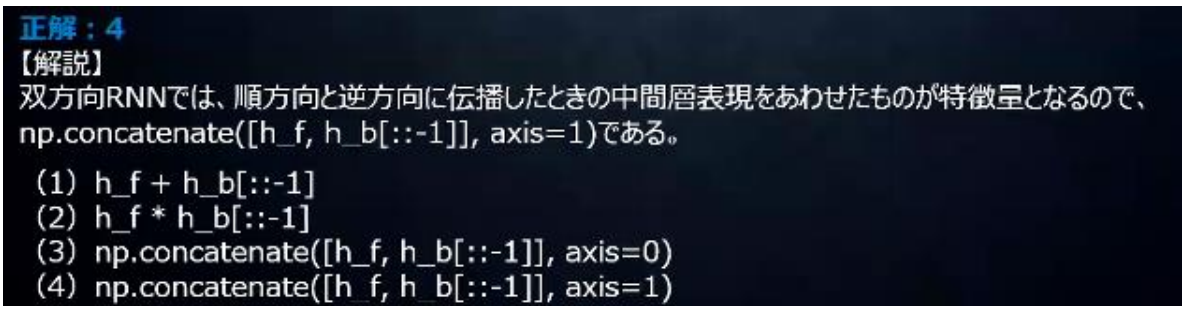
入力ゲート、出力ゲート、忘却ゲート、ＣＥＣはパラメータが多い

（２）ＧＲＵ

リセットゲート、更新ゲート

ＧＲＵの郷がパラメータが少なく、計算時間が早い


演習チャレンジ
以下は双方向RNNの順伝播を行うプログラムである。順方向については、入力から中間層への重みW_f, 一ステップ前の中間層出力から中間層への重みをU f、逆方向に関しては同様にパラメータW b, U bを持ち、両者の中間層表現を合わせた特徴から出力層への重みはVである。_rnn関数はRNNの順伝播を表し中間層の系列を返す関数であるとする。（か）にあてはまるのはどれか
def bidirectional_rnn_net(xs, W_f, U_f, W_b, U_b, V):
"""
W_f, U_f: forward rnn weights, (hidden_size, input_size)
W_b, U_b: backward rnn weights, (hidden_size, input_size)
V: output weights, (output_size, 2*hidden_size)
"""
xs_f = np.zeros_like(xs)
xs_b = np.zeros_like(xs)
for i, x in enumerate(xs):
xs_f[i] = x
xs_b[i] = x[::-1]
hs_f = _rnn(xs_f, W_f, U_f)
hs_b = _rnn(xs_b, W_b, U_b)
hs = [
（か）
for h_f, h_b in zip(hs_f, hs_b)]
ys = hs.dot(V.T)
return ys


正解：4
【解説】
双方向RNNでは、順方向と逆方向に伝播したときの中間層表現をあわせたものが特徴量となるので、
np.concatenate([h_f, h_b[::-1]], axis=1)である。
(1) h_f + h_b[::-1]
(2) h_f * h_b[::-1]
(3) np.concatenate([h_f, h_b[::-1]], axis=0)
(4) np.concatenate([h_f, h_b[::-1]], axis=1)

（解答）（４）

５．　section5_Seq2Seq


（解答）（１）

## ６．　section6_Word2Vec

Word2Vec は、単語をベクトルとして表現し、意味の近さや、次に示す「計算のアナロジー」を行えるようにするための仕組みである。２０１３年に Google 社によって開発された。
Word2Vec を用いることで、単語同士の意味関係をうまく捉え、扱うことができるようになる。

・王様・・・・・(0.4,0.1,0.9,0.4)
・女王・・・・・(0.5,0.2,0.3,0.4)
・男・・・・・・(0.1,0.0,0.8,0.2)
・女・・・・・・(0.2,0.1,0.2,0.3)
ここから、ベクトルが指す天童市の距離を計算し、
・「王様」と「女王」の距離は近い
・「王様」－「男」＋「女」≒「女王」

というような、単語の意味関係を捉えたり、足し引きしたりということをベクトル同士の演算によって行うことができる。

## 7． section7_Attention Mechanism

２０１７年、RNN や CNN を使わずに系列データを扱う Transformer というモデルが提案された。

Transformer は Seq2Seq と同じようにエンコーダ、デコーダからなる。エンコードとデコードはそれぞれ N 回繰り返される。

Transformaer の内部にはいくつものアテンション機構が配置されている。それらアテンション機構の基本モデルはスケール・ドットプロダクト・アテンションと呼ばれている。

（1）アテンション機構

Attention(Q,K,V)=softmax(QKT)・V

Q：クエリー

K：キー

V：バリュー

これは、質問文（Query）を入れると、参照すべき場所（Key）が決まり、その場所の回答候補（Value）が得られる。

Transformer はソフトアテンションを行っているため、参照すべき場所の重み（Key）が決まり、その重みに応じて平均化することで回答候補を得ている。

（2）スケール・ドットプロダクト・アテンション

Attention(Q,K,V)=softmax(QKT/√dk)・V