

# データサイエンス 『実践コース』

## 数理工学PBL

---

Day 2-1：分散表現

一関高専 小池 敦

# 日程（1日目）

2/18(土)10:00-12:00, 13:00-16:00

---

- 自然言語処理と深層学習の基礎
  - Python速習, 基本的な文字列処理
  - Webからのテキスト取得と前処理
  - 日本語に対する自然言語処理
  - 深層学習の基礎（仕組みと進展, Kerasの基礎）

# 日程 (2日目)

2/19(日) 10:00-12:00, 13:00-16:00

---

- 深層学習による自然言語処理
  - 分散表現
  - アテンション
  - 大規模言語モデル (応用と基礎技術)
  - Hugging Faceライブラリによる高度自然言語処理

# 分散表現・アテンション (1日目前半)

---

- 分散表現（講義）
  - 分散表現とは？
  - 基本的な数学（線形結合）
  - 分散表現の求め方
- 実習
  - 分散表現の簡易計算
- アテンション（講義）
  - 再帰型ニューラルネットワークとアテンション
  - セルフアテンション
  - Transformer

# 大規模言語モデル（1日目後半）

---

- 大規模言語モデル（講義）
  - 大規模言語モデルの応用
  - 大規模言語モデルの基礎技術
    - 基本モデル, Text-to-Text, アライメント
- 実習
  - Hugging Face Transformersの基本的な使い方

# 分散表現

---

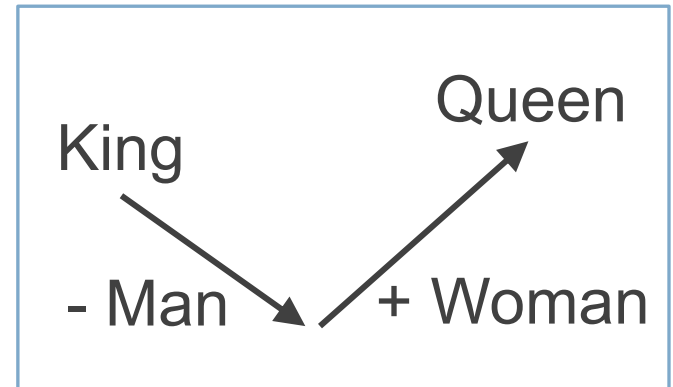
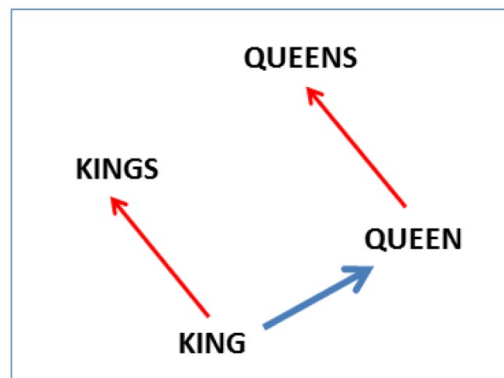
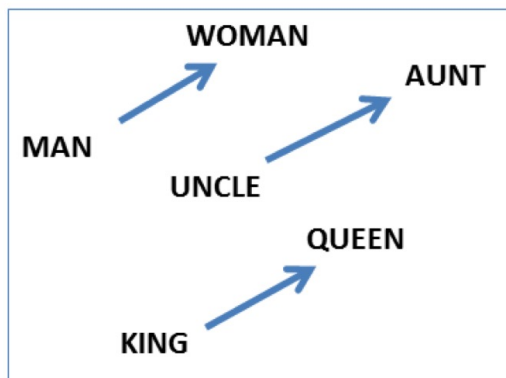
# 単語の分散表現とは？

---

- 単語を低次元の実数値ベクトルで表したもの
  - 数10次元から数100次元
  - One-hotベクトルは語彙サイズ分の次元が必要だった
- 意味が似ている単語が近くに配置される
- 深層学習において標準的に使用されている
  - One-hot ⇒ 分散表現 の変換はword2vecと呼ばれる

# 分散表現の特徴

- ベクトルの足し算に意味づけができる
  - 概ね以下のような関係が成り立つ





# 線形結合

---

# 線形結合 (行ベクトル版)

---

$$[\alpha_1 \quad \alpha_2 \quad \cdots \quad \alpha_n] \begin{bmatrix} \boxed{v_1} \\ \boxed{v_2} \\ \vdots \\ \boxed{v_n} \end{bmatrix} = \begin{aligned} &\alpha_1 \boxed{v_1} \\ &+ \alpha_2 \boxed{v_2} \\ &\vdots \\ &+ \alpha_n \boxed{v_n} \end{aligned}$$
$$= \sum_{i=1}^n \alpha_i v_i$$

## 参考：線形結合（列ベクトル版）

---

$$\begin{bmatrix} \boxed{v_1} & \boxed{v_2} & \cdots & \boxed{v_n} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} = \alpha_1 \boxed{\begin{bmatrix} v_1 \end{bmatrix}} + \alpha_2 \boxed{\begin{bmatrix} v_2 \end{bmatrix}} + \cdots + \alpha_n \boxed{\begin{bmatrix} v_n \end{bmatrix}}$$
$$= \sum_{i=1}^n \alpha_i v_i$$

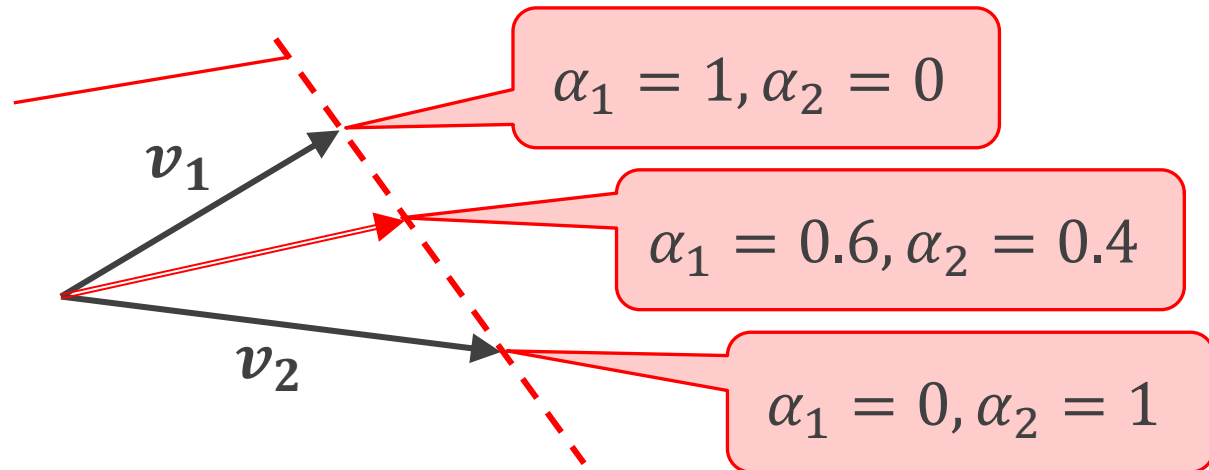
# アフィン結合

- 線形結合  $\sum_{i=1}^n \alpha_i v_i$  において  $\sum_{i=1}^n \alpha_i = 1$  の場合

をアフィン結合と言う

ふたつのベクトルのアフィン結合

アフィン結合  
 $\alpha_1 v_1 + \alpha_2 v_2$  は  
必ずこの直線上  
になる

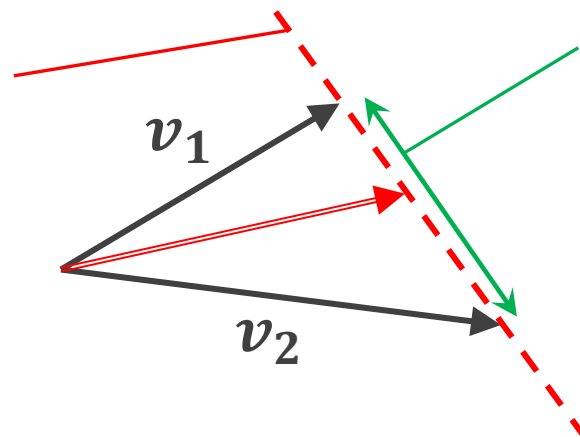


# 凸結合

- アフィン結合において  $0 \leq \alpha_i \leq 1$  ( $1 \leq i \leq n$ ) の場合を凸結合と言う

## ふたつのベクトルの凸結合

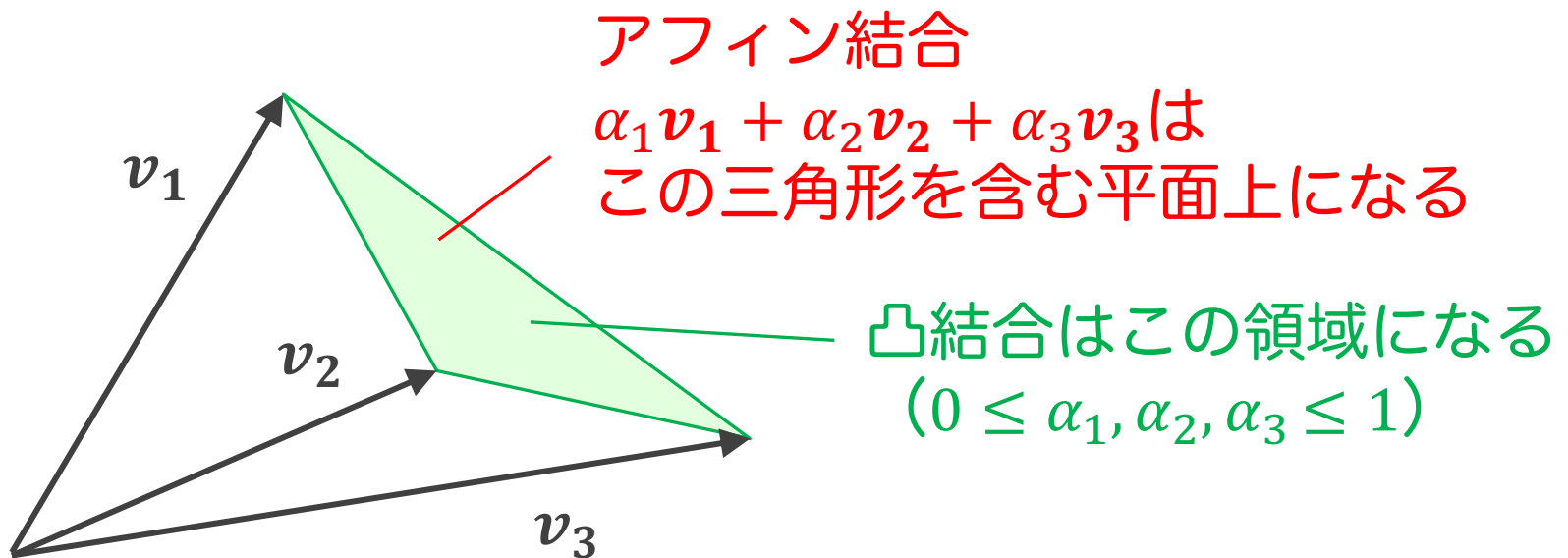
アフィン結合  
 $\alpha_1 v_1 + \alpha_2 v_2$  は  
必ずこの直線上  
になる



凸結合はこの線分上になる  
( $0 \leq \alpha_1, \alpha_2 \leq 1$ )

# アフィン結合と凸結合

## 3つのベクトルのアフィン結合



凸結合  $\Rightarrow$  ベクトルのブレンド

# 線形結合 $\Rightarrow$ 凸結合

- 線形結合の係数をソフトマックスで変換すると、凸結合になる（全係数が0～1で和が1になる）

線形結合



$$[\alpha_1 \quad \alpha_2 \quad \cdots \quad \alpha_n] \begin{bmatrix} \boxed{v_1} \\ \boxed{v_2} \\ \vdots \\ \boxed{v_n} \end{bmatrix}$$

凸結合

$$\text{softmax}([\alpha_1 \quad \alpha_2 \quad \cdots \quad \alpha_n]) \begin{bmatrix} \boxed{v_1} \\ \boxed{v_2} \\ \vdots \\ \boxed{v_n} \end{bmatrix}$$

# One-hotベクトルによる 凸結合

---

$$\begin{bmatrix} \alpha_1 & \cdots & \alpha_k & \cdots & \alpha_n \\ 0 & \cdots & 1 & \cdots & 0 \end{bmatrix} \begin{bmatrix} \boxed{v_1} \\ \vdots \\ \boxed{v_k} \\ \vdots \\ \boxed{v_n} \end{bmatrix} = \boxed{v_k}$$

値が1のindexに対応するベクトルが  
取り出される



# 分散表現の求め方

---

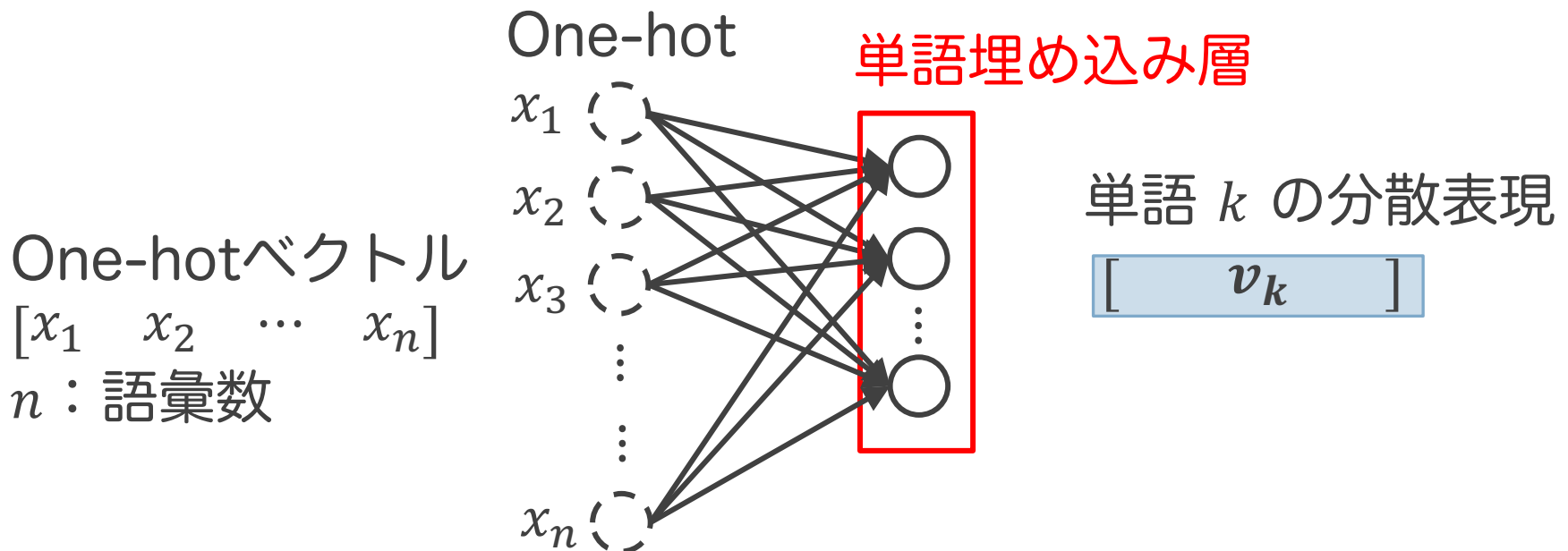
# 分散表現学習の方針

---

- ニューラルネットワークでOne-hotベクトルから分散表現への変換方法を学習させる
- 分散表現を用いて別の問題を解かせるようなネットワークにする
  - 文中のある単語を隠して周りの単語から予測させる (CBOW)
  - 文中のある単語から、周りの単語を予測させる (skip-gram)

# 単語埋め込み (embedded) 層

- 単語IDに対応するOne-hotベクトルを分散表現に変換する層
- 活性化関数のない全結合層となっている(ただし入力単語ID)
- 複数の単語IDを入力すると複数の分散表現が得られる



# 単語埋め込み (embedded) 層 の行列表現

One-hotベクトル

$$\begin{bmatrix} x_1 & \cdots & x_k & \cdots & x_n \\ 0 & & 1 & & 0 \end{bmatrix}$$

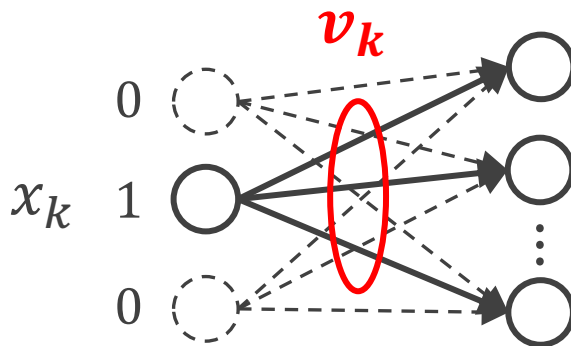
辺の重み $W$

$$\begin{bmatrix} \boxed{v_1} \\ \vdots \\ \boxed{v_k} \\ \vdots \\ \boxed{v_n} \end{bmatrix}$$

=

分散表現

$$\boxed{v_k}$$



単語  $k$  の分散表現は、  
辺の重み行列  $W$  の  
 $k$  行目の行ベクトルとなる

# CBOW (Continuous Bag-of-Words)

- 文中の単語を前後  $c$  単語ずつから予測する
  - 自己教師あり学習となっている  
(教師あり学習だがラベルを用意する必要がない)
  - 予測の正解率は100%にならないが問題ない  
(学習の過程で分散表現が良くなれば十分)

$c = 2$  の時

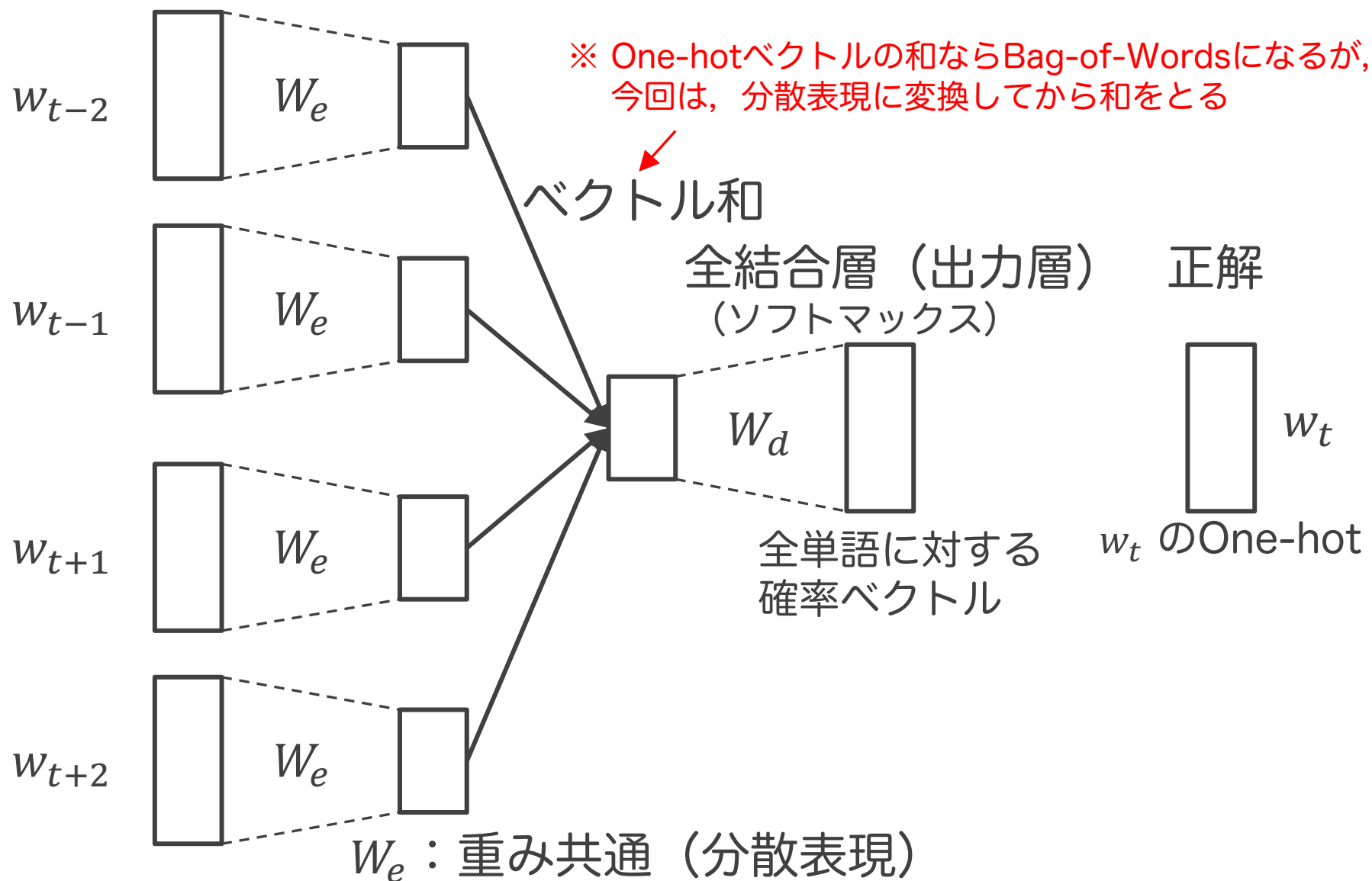
前後2語ずつから予測

必要な / もの / は / 紙 /  / ペン / だけ / だ

$w_{t-2}$   $w_{t-1}$   $w_{t+1}$   $w_{t+2}$

と

# One-hot 単語埋め込み層



# 実習で作成するモデル

---

- 映画のレビュー文から，内容がポジティブかネガティブかを推定する
  - 入力：文の各単語（最大100単語）
  - 出力：論理値（ポジティブならTrue）
- CBOWの出力を単語でなく論理値に変える
  - CBOWはソフトマックス計算が重く工夫が必要

単語ID    One-hot    単語埋め込み層

