

計算言語学₈

深層學習(基礎)

東京大学生産技術研究所

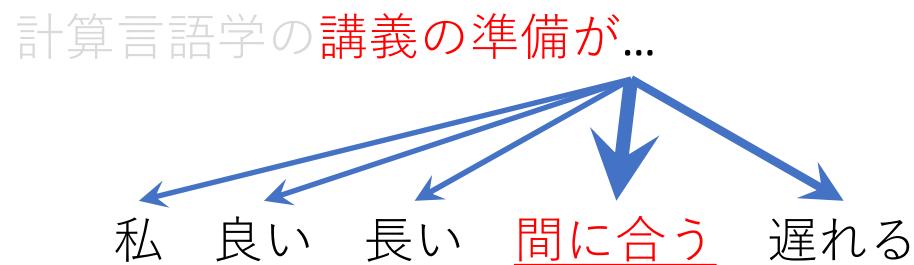
吉永 直樹

site: <http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/class/cl/>

素性工学とその難しさ

- (統計的)自然言語処理 = 部分的な言語構造に対するスコア付け

生成(言語モデル)



構造分類(品詞タグ付け)

Lucy in the sky with diamonds
NNP IN DT NN IN NNS

分類(評判分析)

欠点が少ない → 肯定的

野菜が少ない → 否定的

ラベンダーの香り → 肯定的

構造分類(依存構造解析)

Ringo Star has joined the Beatles

素性設計における基本的なアプローチ: 入力の抽象化と組み合わせ

- 素性 = 入力の抽象化 (高頻度化)
 - 入力の構造の分解・削除: 文 → BoW, 単語 → 接辞
 - 統語クラス: 品詞, 依存関係 (基礎解析の出力):
基礎解析の精度はそもそも100%でない
 - 意味クラス: クラスタ [Brown+ 1993], 埋め込み, シソーラス
文脈非依存かつ必ずしも当該タスクの性能に貢献しない
- 素性の組み合わせ (詳細化)
 - 例: n-gram (評判分析), 品詞の組み合わせ(依存構造解析)
 - 有効な組み合わせを列挙するには職人芸が必要

これらを所与の学習データに合わせて人手で調整 (至難)

(深層)ニューラルネットワーク

- 計算ユニットの連結に基づく計算グラフ
 - 入力から出力への任意の関数を表現・学習可能
- 利点:
 - 入力の抽象化: 入力からタスクに有効な素性を自動学習
 - 素性の組み合わせ: 活性化関数(非線形関数)によって, 素性の組み合わせを陰に考慮
- 欠点:
 - 入力を離散的な記号(単語)から連続的な数値に変換にするため, 計算コストが増加し, 処理の解釈性も失われる

ユニット： ニューラルネットワークの構成要素

- 入力(実数値ベクトル)から出力(実数値)への関数を表現
 - 入力の重み付き線形和に活性化関数(非線形関数)を適用

$$y = f(z)$$

活性化関数 重みベクトル バイアス項

$$z = \mathbf{w}^T \mathbf{x} + b = \left(\sum_i w_i x_i \right) + b$$

- 活性化関数(非線形関数)

$$f(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

シグモイド関数

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

双曲線正接関数

$$f(z) = \text{ReLU}(z) = \max(z, 0)$$

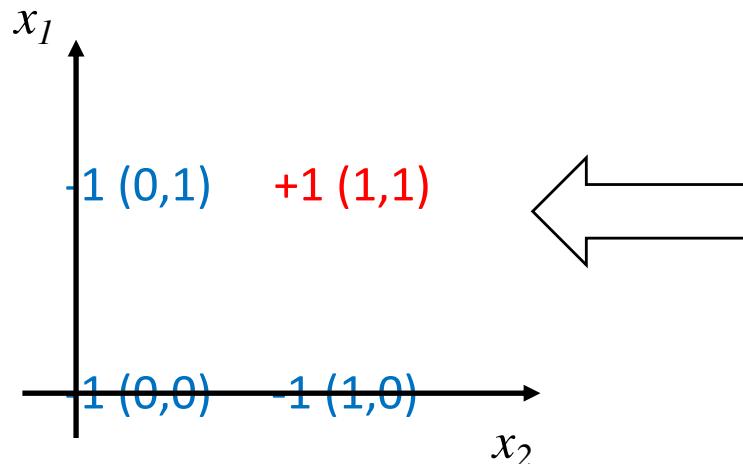
正規化線形関数

なぜ非線形関数を活性化関数に用いるのか？ (1/2)

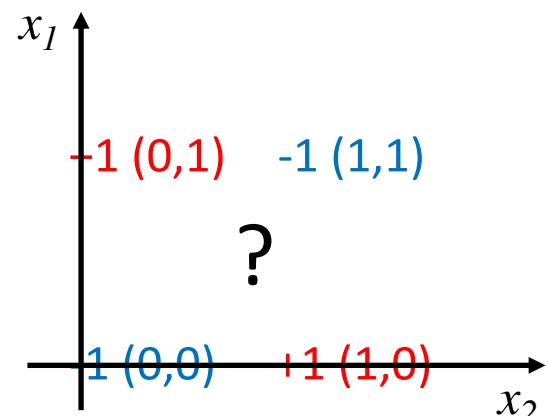
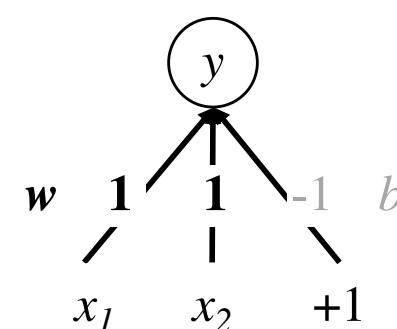
- 非線形関数を活性化関数に用いない場合、線形分離不可能な問題は解けない
 - 例: (単純)パーセプトロン

$$y = \text{sgn}(z) = \text{sgn}(\mathbf{w}^T \mathbf{x} + b)$$

AND 問題



XOR 問題 [Minsky & Papert 1969]



線形分離不可能

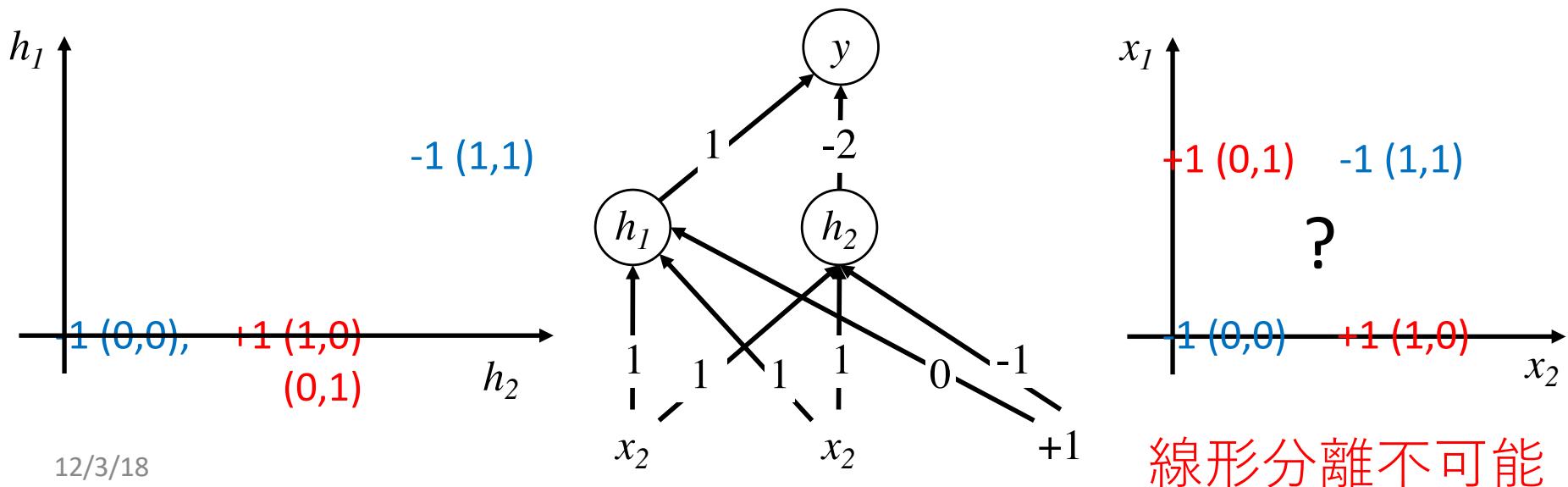
なぜ非線形関数を活性化関数に用いるのか？ (2/2)

- 非線形関数を活性化関数に用いることで、線形分離不可能な問題を解ける

$$y = \text{ReLU}(\mathbf{u}^T \mathbf{h})$$

$$\mathbf{h} = \text{ReLU}(\mathbf{W} \mathbf{x} + \mathbf{b}_h)$$

XOR 問題 [Minsky & Papert 1969]



順伝播型ニューラルネットワーク (1/2) (Feed-Forward Neural Network; FFNN)

- 再帰を含まない単純なニューラルネットワーク
 - 例: 2層の FFNN (隠れ層は1層)

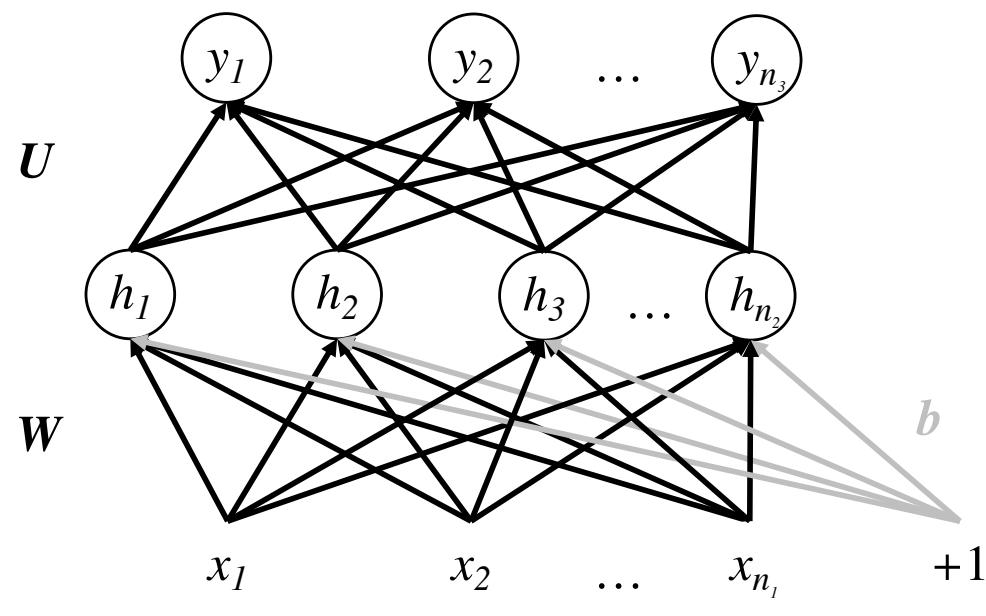
$$\begin{aligned} \mathbf{y} &= \text{softmax}(\mathbf{U}\mathbf{h}) \\ \mathbf{h} &= \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}_h) \end{aligned}$$

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^d e^{z_j}} \quad (1 \leq i \leq d)$$

output layer
(出力層)

hidden layer
(隠れ層)

input layer
(入力)



順伝播型ニューラルネットワーク (2/2) (Feed-Forward Neural Network; FFNN)

- 再帰を含まない単純なニューラルネットワーク
 - 例: 2層の FFNN (隠れ層は1層)

$$\begin{aligned} \mathbf{y} &= \text{softmax}(\mathbf{U}\mathbf{h}) \\ \mathbf{h} &= \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}_h) \end{aligned}$$

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^d e^{z_j}} \quad (1 \leq i \leq d)$$

- 例: n 層の FFNN (隠れ層は $n-1$ 層)

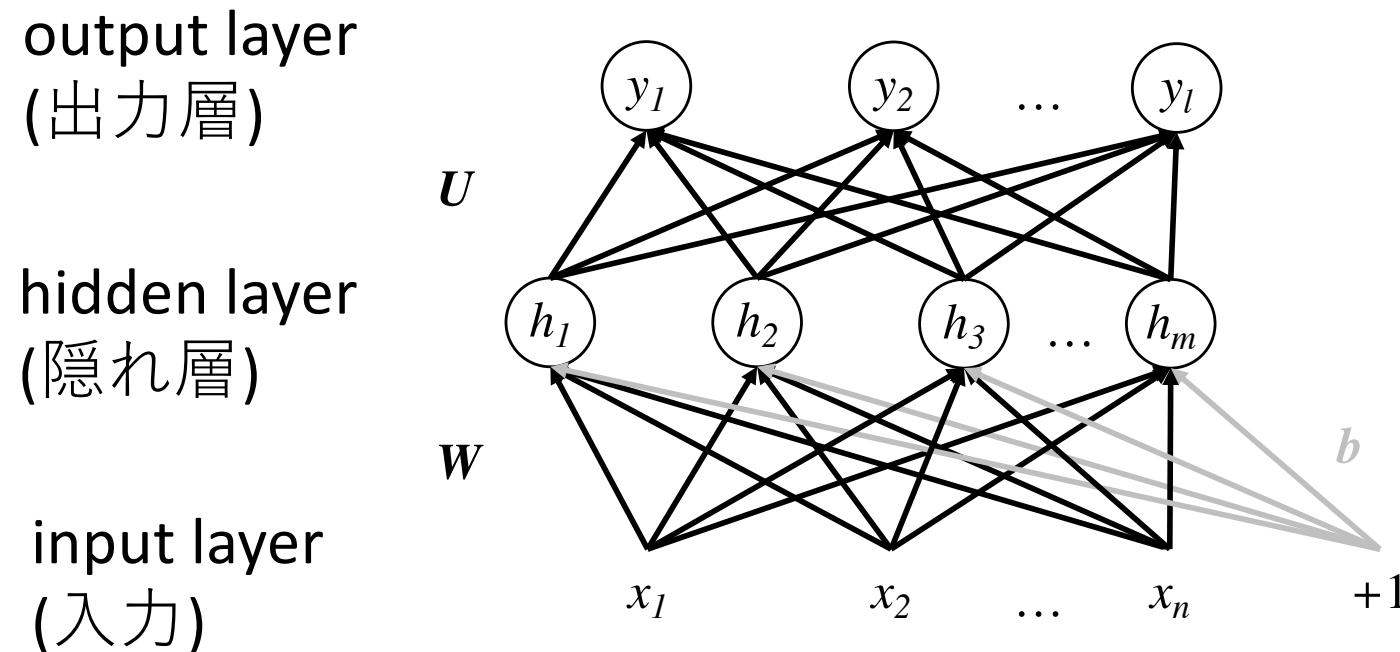
$$\mathbf{y} = \text{softmax}(\mathbf{U}\mathbf{h}^{[n-1]})$$

$$\mathbf{h}^{[0]} = \mathbf{x}$$

$$\mathbf{h}^{[i]} = \sigma(\mathbf{W}\mathbf{h}^{[i-1]} + \mathbf{b}_h^{[i]})(1 \leq i \leq n)$$

ニューラルネットワークの学習

- 損失関数: クロスエントロピー損失
- 最適化: (確率的) 勾配降下法
 - 勾配の計算: 誤差逆伝播法 (backpropagation; backprop)



ニューラルネットワークの学習: 損失関数

- 学習データに対するクロスエントロピー損失
 - 二値分類の場合

$$L_{CE}(\hat{y}, y) = -\log p(y|x) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$$

- 多クラス分類の場合

$$\begin{aligned} L_{CE}(\hat{y}, y) &= -\sum_{i=1}^C y_i \log \hat{y}_i \\ &= -\log \hat{y}_i \quad \boxed{y=y_i} \quad \textit{negative log-likelihood loss} \\ &= -\log \frac{e^{z_i}}{\sum_{j=1}^C e^{z_j}} \end{aligned}$$

ニューラルネットワークの学習: 勾配の計算 (1/3)

- 復習: ロジスティック回帰(隠れ層なし, 二値分類)

$$\begin{aligned}\frac{\partial}{\partial w_j} L_{\text{CE}}(\hat{y}, y) &= \frac{\partial}{\partial w_j} - [y \log \sigma(\mathbf{w}^T \mathbf{x} + b) + (1 - y)(1 - \sigma(\mathbf{w}^T \mathbf{x} + b))] \\ &= - \left[\frac{\partial}{\partial w_j} y \log \sigma(\mathbf{w}^T \mathbf{x} + b) + \frac{\partial}{\partial w_j} (1 - y)(1 - \sigma(\mathbf{w}^T \mathbf{x} + b)) \right]\end{aligned}$$

$$\frac{d \ln(x)}{d} = \frac{1}{x}$$

$$\begin{aligned}\frac{d\sigma(z)}{dz} &= \frac{d}{dz} \frac{1}{1 + e^{-z}} \\ &= -\frac{1}{(1 + e^{-z})^2} \cdot -1 \cdot e^{-z} \\ &= \frac{1}{1 + e^{-z}} \cdot \frac{e^{-z}}{1 + e^{-z}} \\ &= \sigma(z)(1 - \sigma(z))\end{aligned}$$

$$\begin{aligned}&= - \left[\frac{y}{\sigma(\mathbf{w}^T \mathbf{x} + b)} - \frac{1 - y}{1 - \sigma(\mathbf{w}^T \mathbf{x} + b)} \right] \frac{\partial}{\partial w_j} \sigma(\mathbf{w}^T \mathbf{x} + b) \\ &= - \left[\frac{y - \sigma(\mathbf{w}^T \mathbf{x} + b)}{\sigma(\mathbf{w}^T \mathbf{x} + b)(1 - \sigma(\mathbf{w}^T \mathbf{x} + b))} \right] \sigma(\mathbf{w}^T \mathbf{x} + b)(1 - \sigma(\mathbf{w}^T \mathbf{x} + b)) \frac{\partial(\mathbf{w}^T \mathbf{x} + b)}{\partial w_j} \\ &= - \left[\frac{y - \sigma(\mathbf{w}^T \mathbf{x} + b)}{\sigma(\mathbf{w}^T \mathbf{x} + b)(1 - \sigma(\mathbf{w}^T \mathbf{x} + b))} \right] \underbrace{\sigma(\mathbf{w}^T \mathbf{x} + b)(1 - \sigma(\mathbf{w}^T \mathbf{x} + b))}_{\sigma(\mathbf{w}^T \mathbf{x} + b) - y} x_j \\ &= -[y - \sigma(\mathbf{w}^T \mathbf{x} + b)]x_j \\ &= [\sigma(\mathbf{w}^T \mathbf{x} + b) - y]x_j\end{aligned}$$

ニューラルネットワークの学習: 勾配の計算 (2/3)

- 誤差逆伝播法 (error backpropagation; backprop)
 - 合成関数の微分の連鎖律を用い各パラメタの勾配を計算

$$f(x) = u(v(w(x)))$$

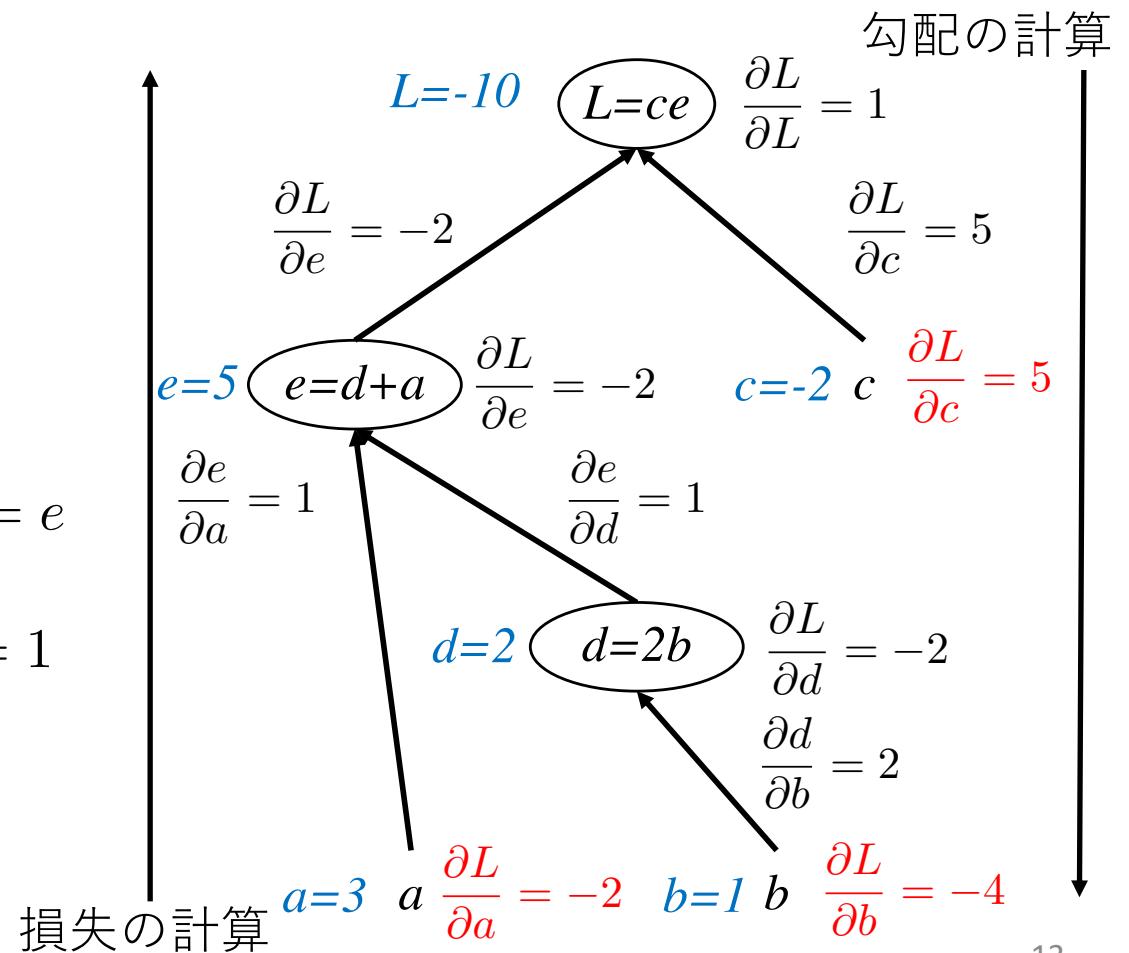
$$\frac{df}{dx} = \frac{du}{dv} \cdot \frac{dv}{dw} \cdot \frac{dw}{dx}$$

例) $L = c(a + 2b)$

$$L = ce : \quad \frac{\partial L}{\partial e} = c, \quad \frac{\partial L}{\partial c} = e$$

$$e = a + d : \quad \frac{\partial e}{\partial a} = 1, \quad \frac{\partial e}{\partial d} = 1$$

$$d = 2b : \quad \frac{\partial d}{\partial b} = 2$$



ニューラルネットワークの学習: 勾配の計算 (3/3)

- 誤差逆伝播法 (error backpropagation; backprop)
 - 合成関数の微分の連鎖律を用い各パラメタの勾配を計算
 - 活性化関数は(劣)微分可能でなければならない

$$\frac{d\sigma(z)}{dz} = (1 - \sigma(z))\sigma(z)$$

$$\frac{d \tanh(z)}{dz} = 1 - \tanh^2(z)$$

$$\frac{d \text{ReLU}(z)}{dz} = \begin{cases} 0 & \text{for } z < 0 \\ 1 & \text{for } z = 0 \\ 1 & \text{for } z > 0 \end{cases}$$

劣勾配 (subgradient)

ニューラルネットワーク学習の工夫

- 問題:

- 損失関数が非凸となる(ことが多い)ため局所解に嵌まる
- 過適合 (over-fitting) しやすい
- 行列計算が重い

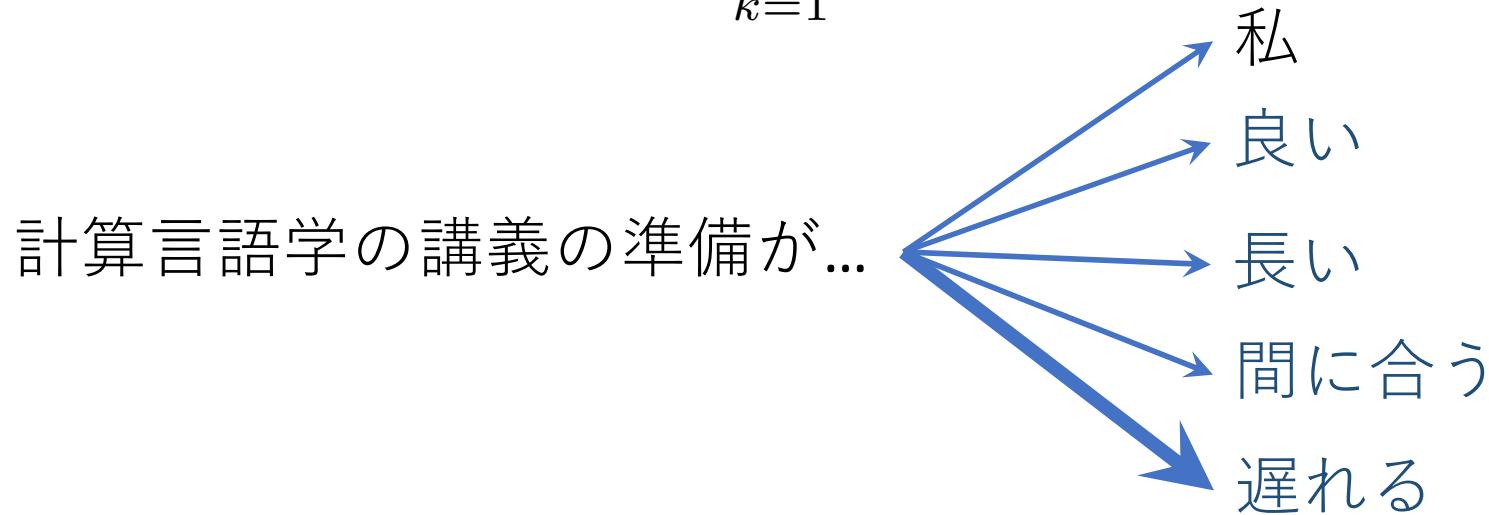
- 対策:

- 亂数で初期化し, 正則化やドロップアウト [Hinton+ 2012] (学習時に隠れ層のユニットをランダムに削除) を用いる
- モデルのハイパーパラメタ(学習率, バッチサイズ, 隠れ層数・次元数, 活性化関数)のチューニング
- 開発データで early stopping
- SGD の亜種 (Adam [Kingma+ 2015] など) で最適化する
- GPU を用いる

ニューラル言語モデル [Bengio+ 2003] (1/3)

- N-gram 言語モデルの問題: ゼロ頻度問題
 - 履歴: N-1語の単語(シンボル)の系列

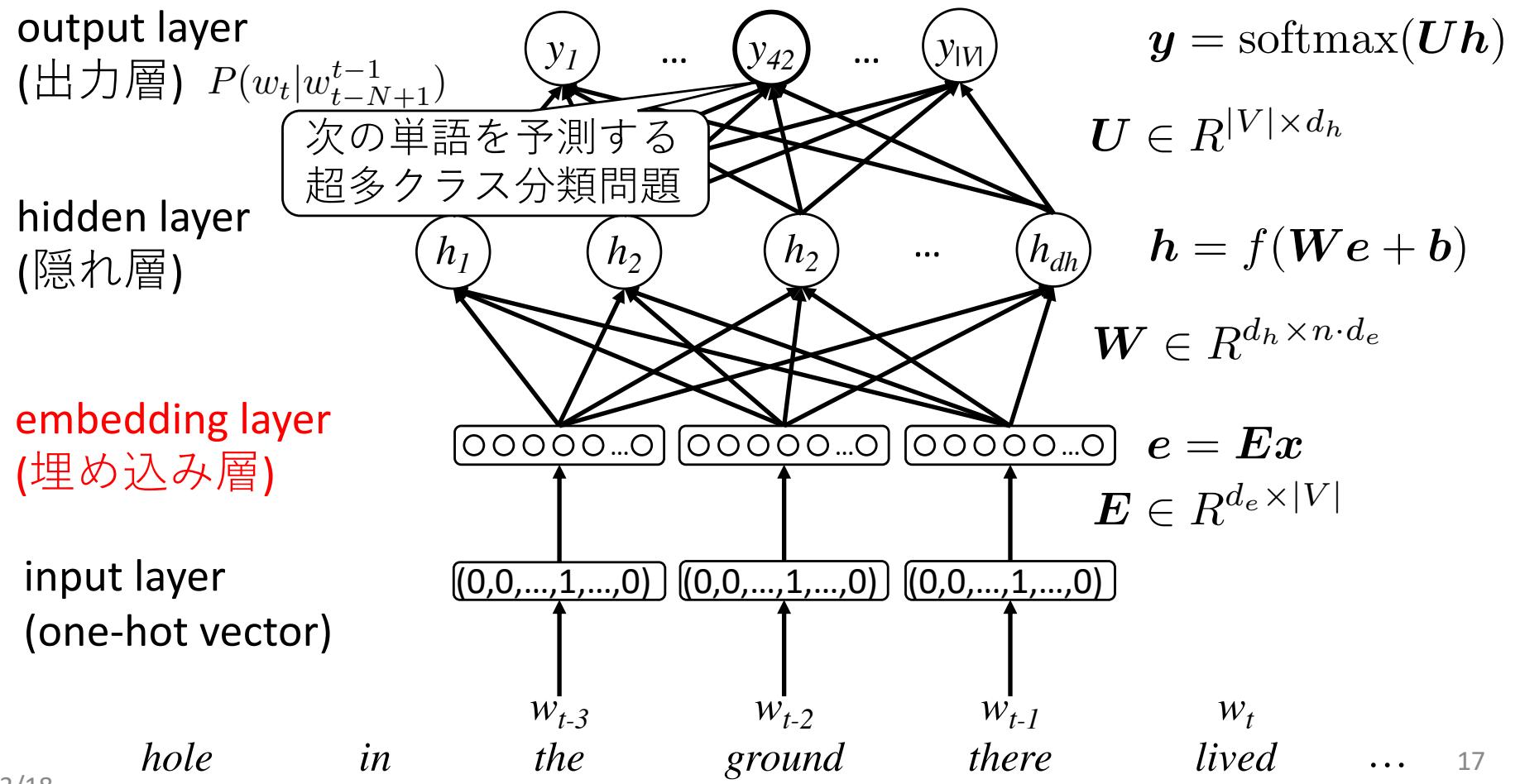
$$P(w_1 \dots w_n) \approx \prod_{k=1}^n P(w_k | w_{k-N+1}^{k-1})$$



$$P(\text{“遅れる”} | \text{“計算言語学の講義の準備が”}) = \frac{C(\text{“計算言語学の講義の準備が遅れる”})}{C(\text{“計算言語学の講義の準備が”})}$$
$$= 0$$

ニューラル言語モデル [Bengio+ 2003] (2/3)

- 順伝播型ニューラルネットワークの言語モデル
 - 履歴: N-1語の単語埋め込み(実数値ベクトル)の系列



ニューラル言語モデル [Bengio+ 2003] (3/3)

- 順伝播型ニューラルネットワークの言語モデル
 - 履歴: **N-1語の単語埋め込み(実数値ベクトル)の系列**
 - 埋め込み層(隠れ層)により **履歴の類似性をモデル化**
 - Skip-gram gram w/ negative sampling (SGNS) などの学習結果で初期化してもよい (pre-training)
 - N-gram モデルより優れた予測性能 (perplexity)
 - スムージング不要だが未知語の問題は依然残る
- 学習: 誤差逆伝播法

$$L = -\log p(w_t | w_{t-N+1}^{t-1}; \theta = \{\mathbf{E}, \mathbf{W}, \mathbf{U}, \mathbf{b}\})$$

$$\theta_{t+1} = \theta_t - \eta \frac{\partial -\log p(w_t | w_{t-N+1}^{t-1})}{\partial \theta}$$

ニューラル言語モデル [Bengio+ 2003] (再掲)

- 順伝播型ニューラルネットワークの言語モデル
 - 履歴: $N-1$ 語の単語埋め込み(実数値ベクトル)の系列

output layer

(出力層) $P(w_t | w_{t-N+1}^{t-1})$

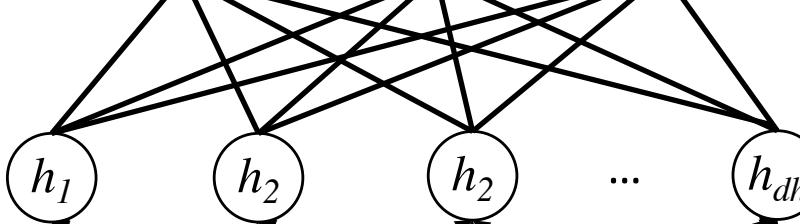


$$\mathbf{y} = \text{softmax}(\mathbf{U}\mathbf{h})$$

$$\mathbf{U} \in R^{|V| \times d_h}$$

hidden layer

(隠れ層)

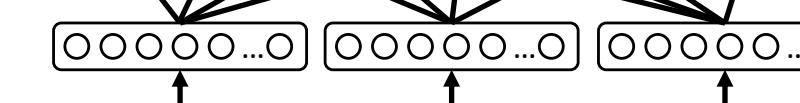


$$\mathbf{h} = f(\mathbf{W}\mathbf{e} + \mathbf{b})$$

$$\mathbf{W} \in R^{d_h \times n \cdot d_e}$$

embedding layer

(埋め込み層)



$$\mathbf{e} = \mathbf{E}\mathbf{x}$$

$$\mathbf{E} \in R^{d_e \times |V|}$$

input layer

(one-hot vector)



固定長の履歴しか扱うことができない

w_{t-3}

in

the

w_{t-2}
 $ground$

w_{t-1}
 $there$

w_t
 $lived$

\dots

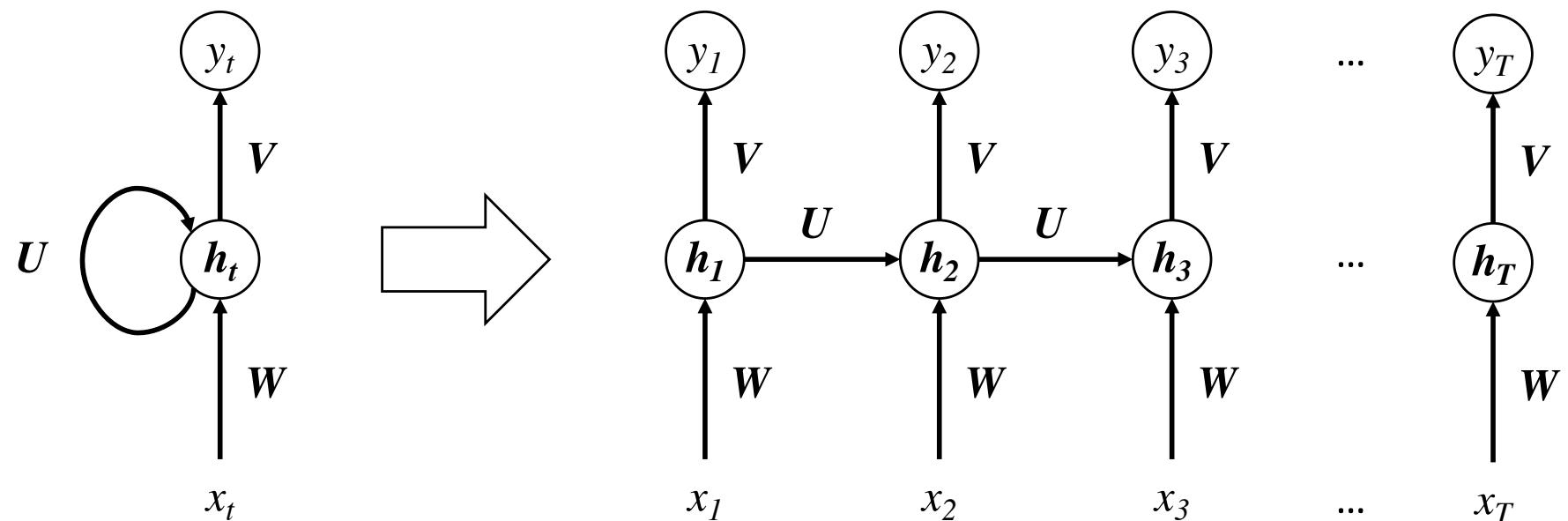
Elman Nets: 単純な再帰型ニューラルネットワーク (Recurrent Neural Network) [Elman 1990]

- ユニット間に再帰を含むニューラルネットワーク
 - 入力系列を各時刻で一つずつ読み、隠れ層を逐次計算

$$y_t = f(Vh_t)$$

$$h_t = g(Uh_{t-1} + Wx_t)$$

直前の時刻の隠れ層と現時刻の入力層から現時刻の隠れ層を計算



可変長の入力を扱うことが可能

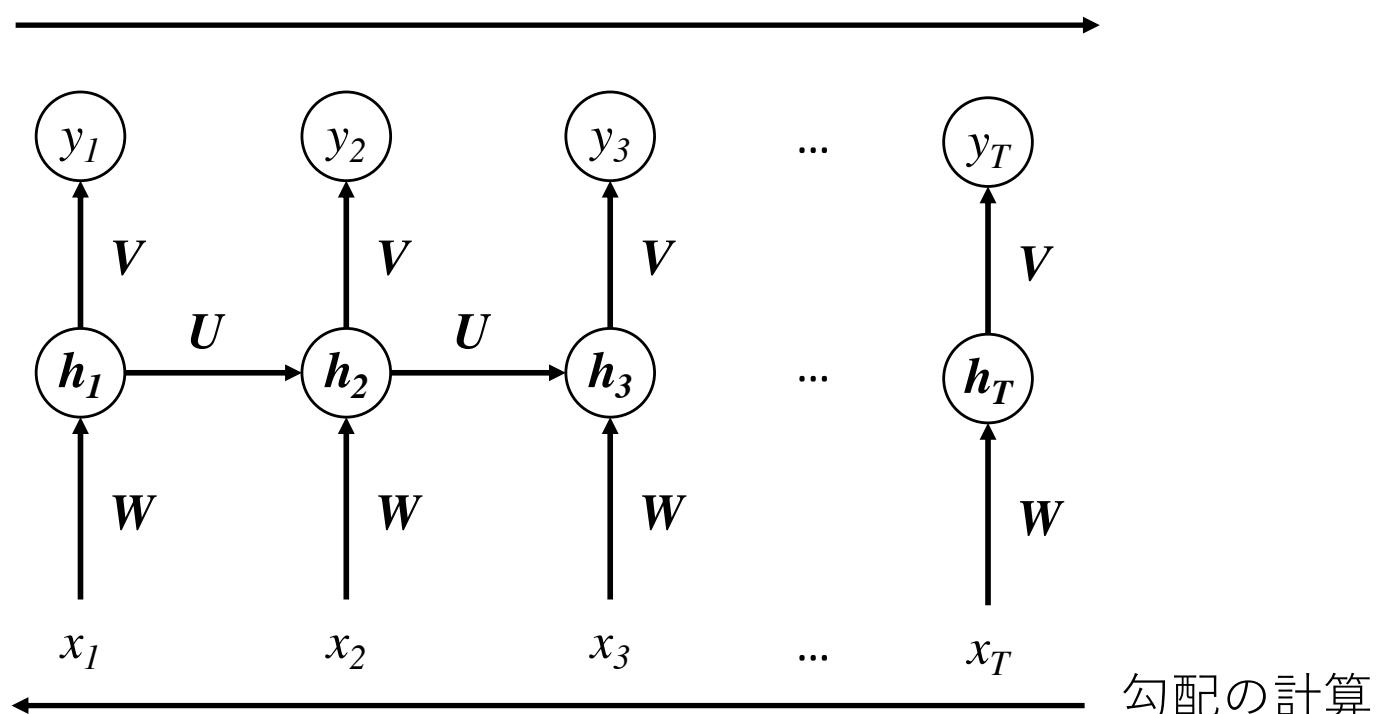
再帰型ニューラルネットワークの学習

- Backpropagation through time (BPTT)

時間方向に展開した計算グラフ上で誤差逆伝播法

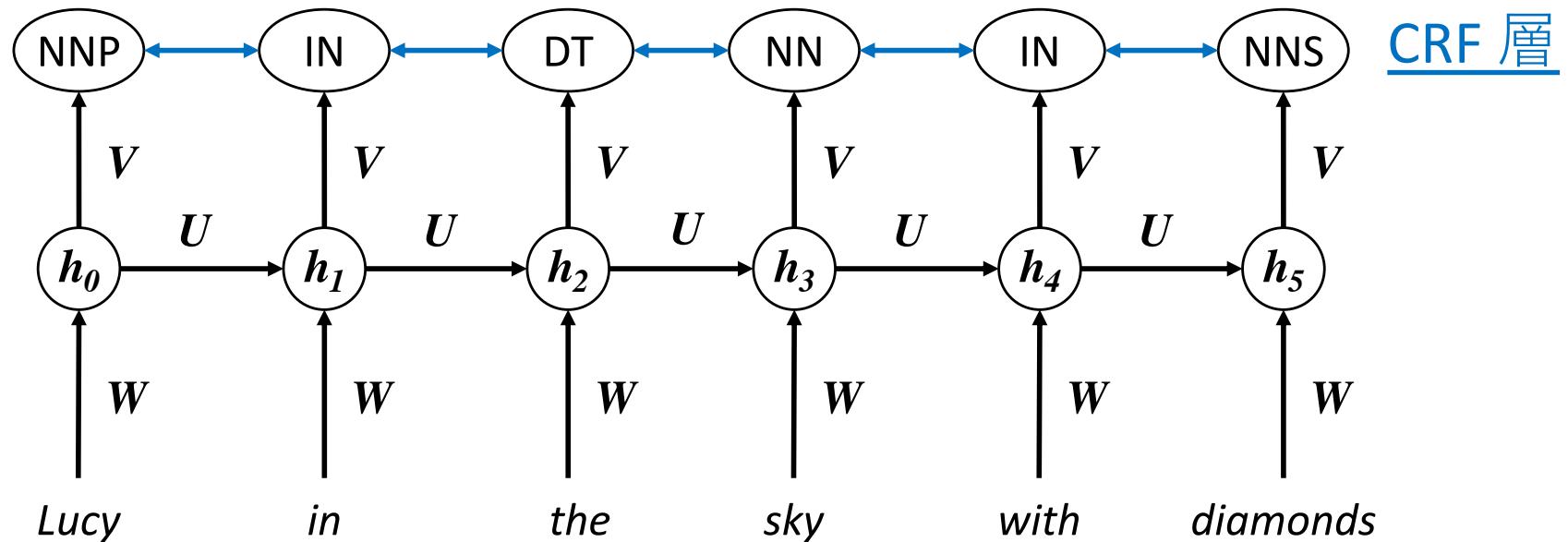
- ミニバッチのように同一パラメタに対する勾配を累積
- 入力が長く全展開できなければ途中で計算を打ち切る

損失の計算



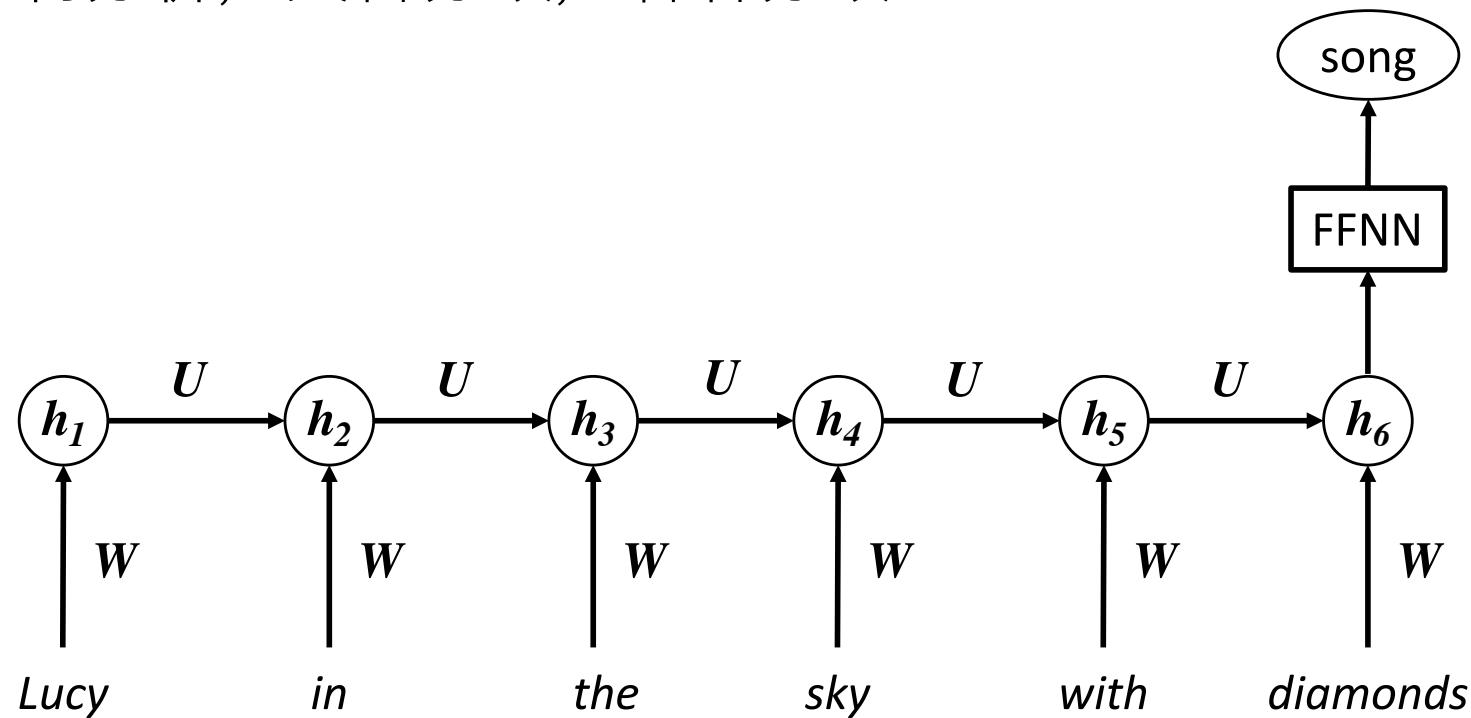
再帰型ニューラルネットワークの応用 (1/3): 系列ラベリング

- 入力に対して対応するラベルを逐次出力
(softmax 関数により出力層で多クラス分類)
 - 品詞タグ付け, 固有表現抽出 (BIOタギング), etc.
- 隣接ラベル間の依存関係をCRF層でモデル化



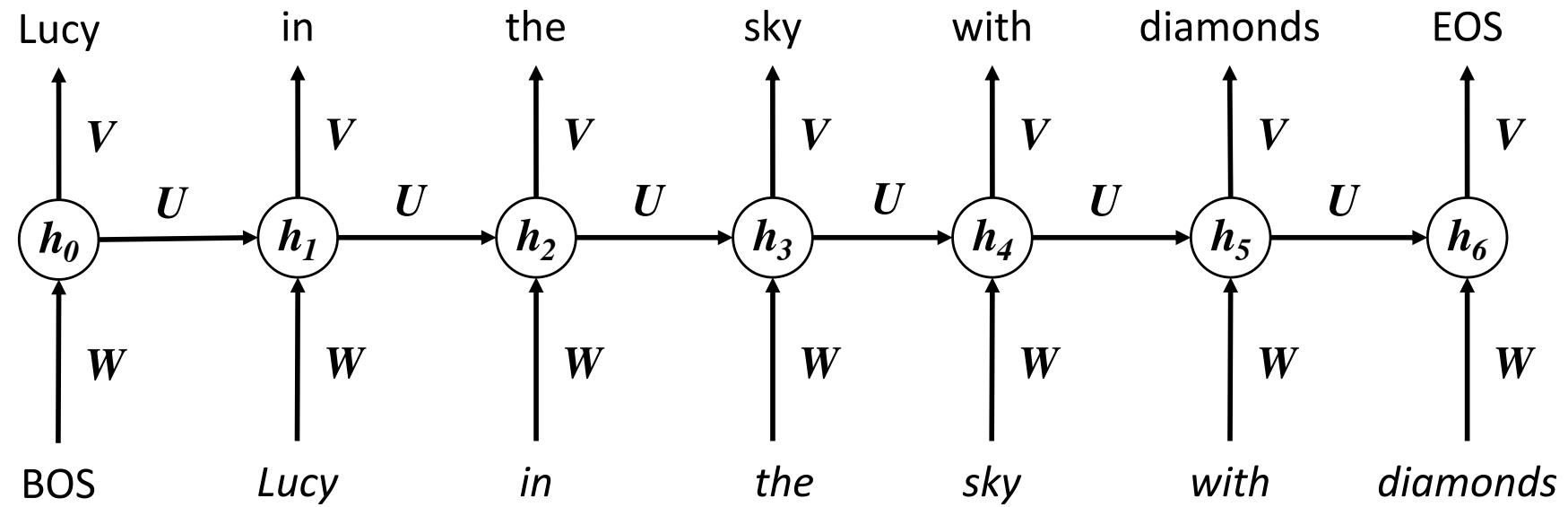
再帰型ニューラルネットワークの応用 (2/3): 分類

- 最終時刻の隠れ層を入力とする順伝播型ニューラルネットワーク(FFNN)で出力のラベルを計算
 - 評判分析, 文書分類, 著者分類 etc.



再帰型ニューラルネットワークの応用 (3/3): 生成

- 出力を次の時刻の入力とし、BOS から系列を生成



単純な RNN の問題

- 単純な RNN の隠れ層が捉える文脈は実際には局所的なものに限られる
 - 各隠れ層(固定サイズ)で当該時刻の出力と全未来の出力の両方に必要な情報を保持する必要がある

The flights the airline was cancelling were full.

- **勾配消失問題**: 内積の繰り返しにより過去の勾配のほぼ0になる(パラメタが更新されない)

過去の情報を適切に忘れる(言い換えると必要な情報を選択的に記憶する)仕組みが必要

LSTM (Long-short Term Memory)

[Hochreiter+ 97, Gers+ 99]

- ゲート関数により情報の伝播を制御

$$g_t = \tanh(\mathbf{U}_g h_{t-1} + \mathbf{W}_g x_t)$$

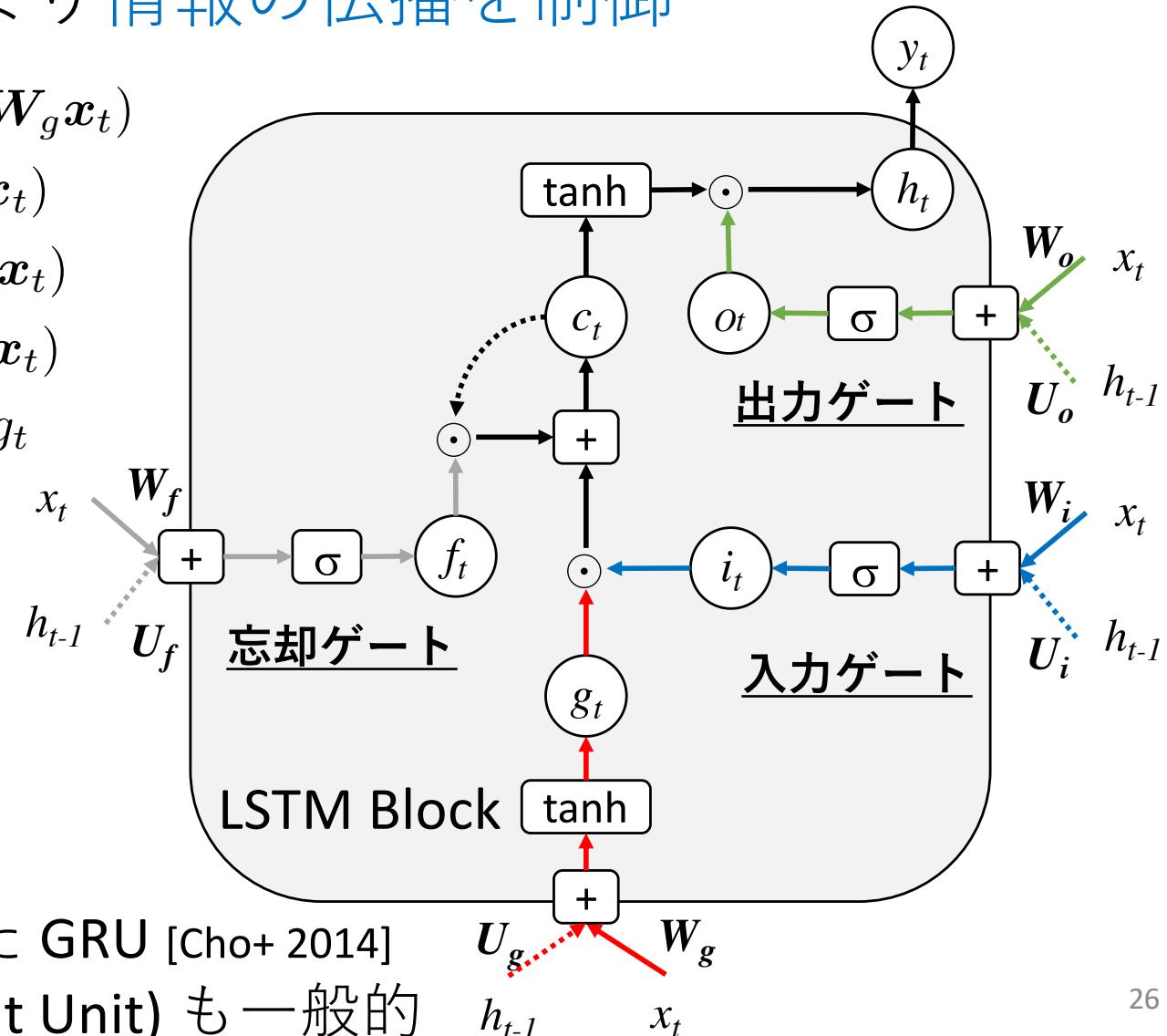
$$i_t = \sigma(\mathbf{U}_i h_{t-1} + \mathbf{W}_i x_t)$$

$$f_t = \sigma(\mathbf{U}_f h_{t-1} + \mathbf{W}_f x_t)$$

$$o_t = \sigma(\mathbf{U}_o h_{t-1} + \mathbf{W}_o x_t)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

$$h_t = o_t \odot \tanh(c_t)$$

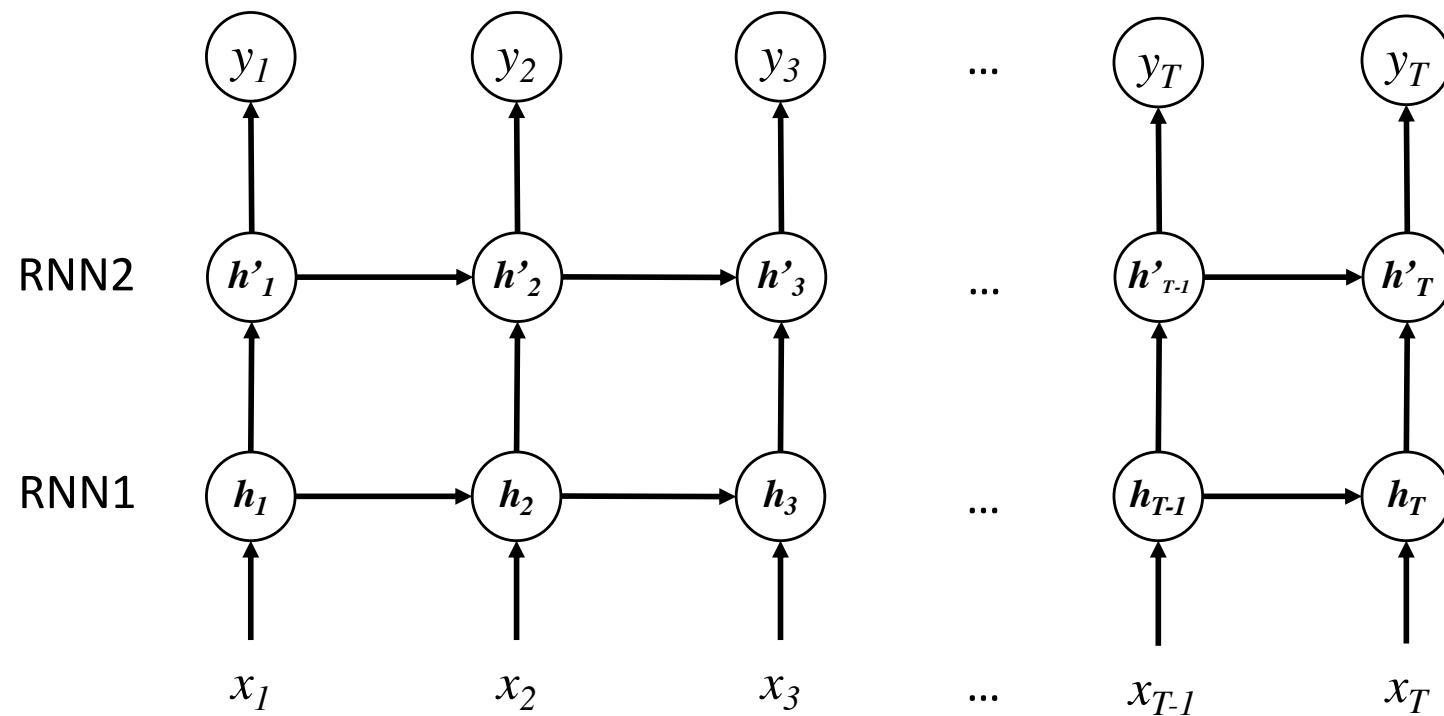


深層学習における未知語問題とその対策

- 深層学習ではパラメタ数を抑えるため扱える語彙サイズに対する物理的な制限が存在
- 当然、学習データにない単語は未知語となる
- 対策1:未知語の数を減らす
 - Byte-Pair Encoding 等により被覆率の高い語彙集合を得る
- 対策2: 未知語用の埋め込みを用意する
 - 学習データ中の低頻度語を未知語とみなす
- 対策3: 文字レベルの処理により入力を補強
 - 文字埋め込みから単語の埋め込みを計算

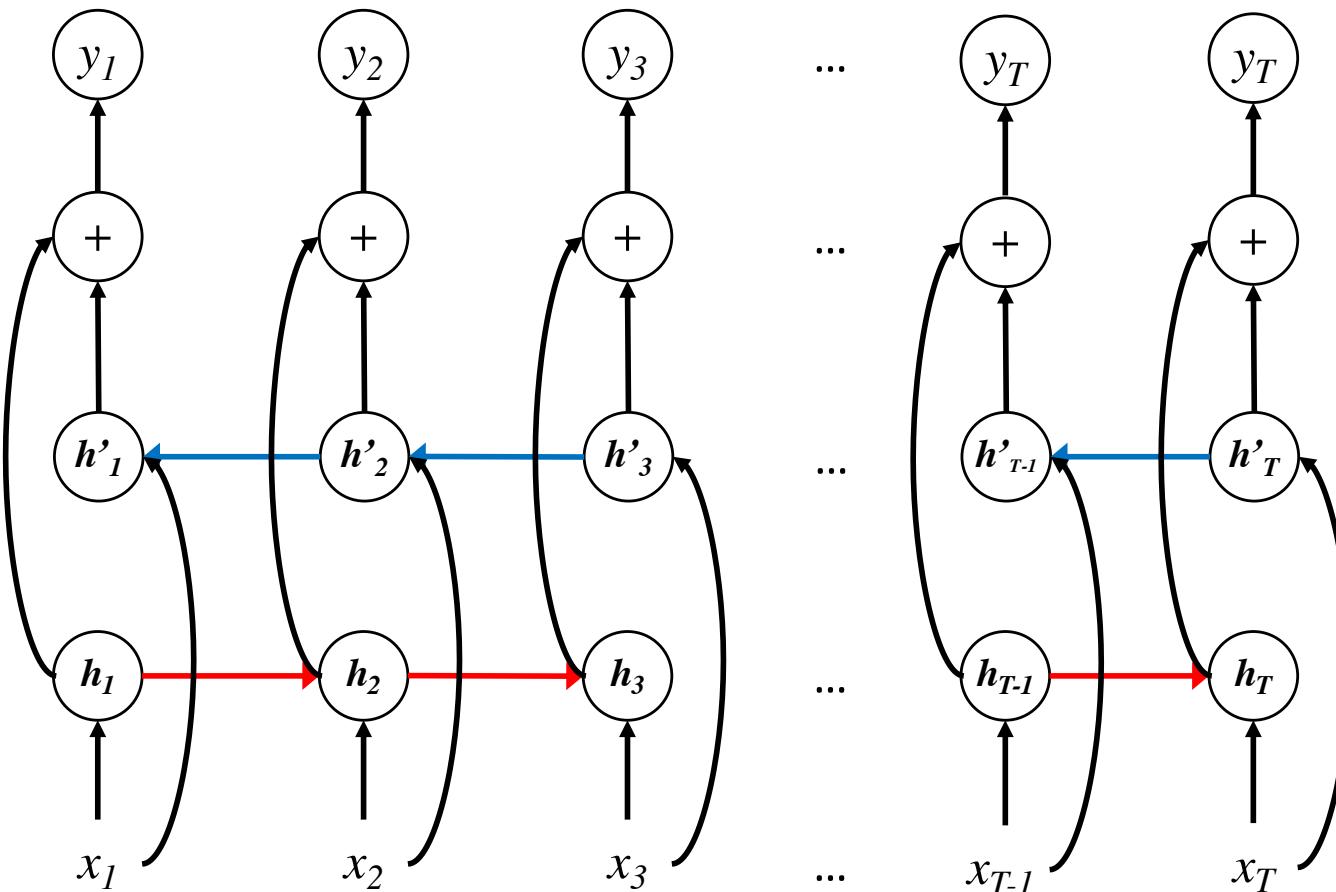
RNN の拡張 (1/2): Stacked RNNs

- RNNの出力を別のRNNに入力
 - 後段の RNN ほど特徴量の抽象化が進む



RNN の拡張 (2/2): Bidirectional RNNs

- 入力を左から右に読むRNNと右から左に読むRNNの出力を連結



英語の固有表現抽出での評価 [Lample+ 2016]

- Dataset: CoNLL-2003
- 事前学習 (*-LSTM): English Gigaword ver. 4

Model	F_1	
CRF [Passos+ 2014]	87.93	
CRF w/ Brown clusters	90.05	Brown クラスタや skip-gram を用れば人手の素性でも高精度
CRF w/ skip-gram	89.68	
Bi-LSTM w/ char Bi-LSTM	89.15	教師信号を学習データで与えるか 素性設計(辞書等)で与えるかの違い
Bi-LSTM-CRF	90.20	
Bi-LSTM-CRF w/ char Bi-LSTM	90.94	CRF層 + 文字 bi-LSTM により高精度化
Stacked LSTM	87.96	
Stacked LSTM w/ char Bi-LSTM	90.33	

本日のまとめ

- 素性工学: 入力の抽象化とその組み合わせ(詳細化)
- 深層ニューラルネットワーク:
活性化関数に基づくユニットを結合した計算グラフ
sigmoid, tanh, ReLU, softmax, etc.
 - 入力の適切な抽象化を自動で学習
 - 学習: 誤差逆伝播法 + 確率的勾配降下法
 - FFNN: 順伝播型ニューラルネットワーク(固定長入出力)
 - RNN: 再起型ニューラルネットワーク(可変長入出力)