特別研究報告書

打ち切り型通時的誤差逆伝播法を用いた RNN の効率的な敵対的学習

指導教員　　下平英寿　教授

京都大学工学部情報学科
数理工学コース
平成 28 年 4 月入学

木村　拓仁

令和 2 年 1 月 29 日提出

摘要

RNN は文章分類や機械翻訳のような自然言語処理のタスクで広く用いられているが, データ量が少ない場合過学習を起こすため性能が低くなってしてしまう. このため, 学習データに人工的な処理を施しデータ量を増やすデータ拡張が用いられる. 敵対的学習は敵対的事例と呼ばれる分類器が誤判別するデータを生成し学習を行う手法であり, RNN を用いた自然言語処理のタスクに適用することで, テスト損失が大幅に改善することは先行研究で調べられている. しかし, RNN を用いた敵対的学習では, 学習に通常以上の時間がかかってしまう問題がある.

本研究では, RNN を用いた敵対的学習での学習時間を短縮することを目的とし, RNN の many-to-one モデルにおける計算量を軽減した敵対的学習を提案した. 提案手法では, 敵対的事例の作成に必要な勾配計算に打ち切り型通時的誤差逆伝搬法を用いることで, 計算量の軽減を図った. この実験には, 文章分類タスクを用い 4 つのデータセットで性能を調べた. その結果, 時間が短縮できただけでなく, パラメータ次第では性能が向上することが確認できた.

目次

1 序論

近年, 機械学習の手法は, 商品レコメンデーションや医療診断など社会の至る部分で用いられ, その中でも, 動物の神経回路を模したニューラルネットワークは, 数々の分野でブレイクスルーを起こしている. テキスト化された自然言語をコンピュータに処理させる一連の技術である自然言語処理分野では, 検索エンジンや機械翻訳やかな漢字変換などのタスクにおいて精度を大幅に向上させている.

しかし, これらの手法で十分な精度を達成するためには, 膨大なデータ量が必要な場合が多く, データ量が少ない場合は, 過学習を起こし性能が著しく悪くなってしまう場合が多く, 機械学習の手法を用いることができない. よって, データ拡張と呼ばれる学習データに人工的な処理を施し新たなデータを増やすことが行われる. 自然言語処理におけるデータ拡張は, ドメイン知識の要否で分類することができる. ドメイン知識を用いる場合, 同義語に置き換えること [1] や 2 単語間の依存構造を逆にすること [2] や翻訳した文章をまた逆翻訳した文章をデータとして用いること [3] が主に用いられる. ドメイン知識を用いない場合は, 単語を入れ替えること [4] やデータにノイズを乗せること [5] や分類器が誤判別するような人工データを作る敵対的学習 [6] が用いられる. 本論文では, 敵対的学習に焦点を置く.

敵対的学習は, 敵対的事例と呼ばれる分類器が誤判別するような学習データから生成されたデータを学習に用いる手法 [6] である. 機械学習の手法では, 与えられたデータの近傍点は同じ性質を持つとされるが全ての近傍点では必ずしもそうではないことが知られ [7], データに悪意持った微小な摂動を加えたデータが敵対的事例である. この手法を用いることで, RNN を用いた自然言語処理においても精度が向上しテスト損失が低くなることは [8] で確かめられている.

しかし, RNN を用いた敵対的学習では, 学習に時間がかかってしまう問題がある. これは, 与えられたデータに対する損失関数の勾配が敵対的事例の生成に必要となるためである.

本研究では, 敵対的学習における学習時間を短縮することを目的とし, 勾配計算に打ち切り型通時的誤差逆伝播法を用いた敵対的学習法を提案した. 実験として, 4 つのデータセットで通常の敵対的学習と提案手法を精度と損失値の観点から比較した結果, 時間が短縮されただけでなく, パラメータ次第では性能が向上することが確認できた.

本論文の構成は, 「準備」, 「提案手法」,「実験」, 「まとめ」からなっている. 2 章の「準備」では, 本論文でもちいた前提知識を説明する. 2.1 節ではニューラルネットワークの概要を説明し, 2.2 節では単語の分散表現を得る手法 word2vec[9] を説明し, 2.3 節では時系列データを扱える再帰型ニューラルネットワークを説明し, 2.4 節では敵対的学習について説明する. 3 章の「提案手法」では, 本論文で扱うタスクを説明し, 通常の敵対的学習と提案手法を比較しながら, 提案手法を模式図を用いて説明する. 4 章の「実験」では, 4.1 節で用いたデータセットについて説明し, 4.2 節で実験結果を示し提案手法では時間短縮されかつ性能が向上していることを示す. 4.3 節では, 提案手法のハイパーパラメータについて解析する. 5 章の「まとめ」では, 提案手法の概要をまとめ, 用いるべきハイパーパラメータの値について言及する.

2　準備

本章では, 本論文で必要となる前提知識を説明する. 2.1 節では, ニューラルネットワークの概要を説明する. 2.2 節では単語の意味をベクトル表現化する手法 word2vec について説明する. 2.3 節では, 過去の情報を保持できる回帰型ニューラルネットワーク (Recurrent Neural Network、RNN) について説明し, RNN の一種である LSTM と GRU について説明する. 2.4 節では, データ拡張手法の一つである敵対的学習について説明する.

2.1　ニューラルネットワーク

ニューラルネットワークとは, 人間の脳を模倣して考案された関数を学習できる構造のことである. 脳のニューロンに対応するものはユニットと呼ばれ, 様々にユニットが結合されたモデルが提案されている. 2.2 節の Skip-gram・CBoW, 2.3 節の RNN は全てニューラルネットワークの一種である. 本節では, ニューラルネットワークの一般的な構造およびパラメータ学習に使用する勾配降下法と勾配を効率的に求める誤差逆伝播法について述べる.

2.1.1　構造

ニューラルネットワークは, 線型写像と非線型写像を繰り返し行う関数であり, (1) から (4) のように定式化される.

$$z_1 = x \tag{1}$$
$$u_l = W_l z_l \ (l = 1, \cdots, N-1) \tag{2}$$
$$z_{l+1} = \sigma_l(u_l + b_l) \ (l = 1, \cdots, N-1) \tag{3}$$
$$f_{NN}(x) = z_N \tag{4}$$

ここで, $z_l (l = 1, \cdots, N-1)$ の各要素がユニットであり, 必要に応じてユニット数は手動で変更できる. 行列 W_l、$b_l (l = 1, \cdots, N-1)$ を重み, バイアスと呼び, これらが学習するパラメータとなる. $\sigma_l(\cdot) (l = 1, \cdots, N-1)$ は, 非線形関数であり, 活性化関数と呼ばれる. 活性化関数として, シグモイド関数 $\sigma(x) = 1/(1 + \exp(-x))$ や逆正接関数 $\sigma(x) = \tan^{-1}(x)$ や ReLU(Rectified Linear Unit) 関数 $\sigma(x) = \max(0, x)$ がよく用いられる. また, N を層数と呼び, 層数 N のニューラルネットワークを N 層ニューラルネットワークと呼ぶ. z_1、z_N に対応するユニットをまとめて入力層, 出力層と呼び, 入力層と出力層の間の $z_2, \cdots, z_{N-1}$ のユニットはまとめて中間層と呼ぶ. 中間層のユニット数および層数の最適な値は手動で探索する必要があるが, ニューラルネットワークは 3 層以上のとき, 中間層のユニット数を十分大きくとると, 任意の連続関数を任意の精度で近似できることが知られている.

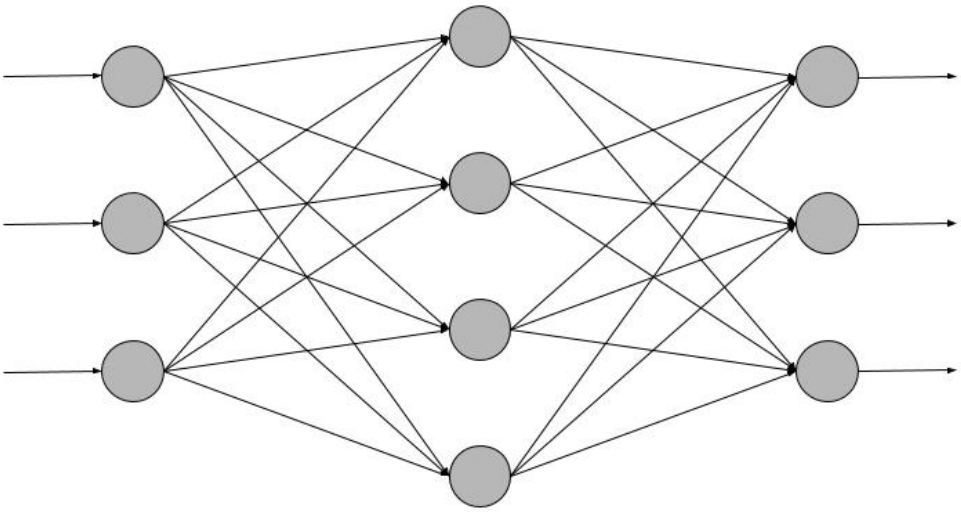

図 1　ニューラルネットワークの模式図: 丸がユニットを表し, ユニットの縦の列が層である. 左から, 順に入力層, 中間層, 出力層と呼ぶ.

2.1.2　学習法

パラメータの学習では, 入力データ x の出力 $f_{NN}(x)$ と対応する目標値 y のズレを設定し, このズレを小さくするようなパラメータに更新する. このズレを測る関数 $E(x, y; W_1, b_1, \cdots, W_{N-1}, b_{N-1})$ を損失関数と呼び, 誤差が最小化するように W_l、b_l を最適化することが学習に対応する.

損失関数の最適化には, 勾配降下法が用いられる. 勾配降下法は, パラメータ θ の微分可能な関数 $f(\theta)$ を最小にする θ を見つけるために, 適当なパラメータ η を用いて, θ を $-\eta\nabla_\theta f(\theta)$ だけ手動で選んだ回数分反復的に更新する方法であり, k 回更新した θ_k から θ_{k+1} は, (5) で与えられる.

$$\theta_{k+1} = \theta_k - \eta\nabla_\theta f(\theta_k) \tag{5}$$

本論文で用いた学習法 Adam [11] は, 勾配降下法を改良したものであり, (6) から (11) に従って学習する.

$$g_k = \nabla_\theta f_k(\theta_{k-1}) \tag{6}$$
$$m_k = \beta_1 m_{k-1} + (1-\beta_1) g_k \tag{7}$$
$$v_k = \beta_2 v_{k-1} + (1-\beta_2) {g_k}^2 \tag{8}$$
$$\hat{m}_k = m_k/(1-\beta_1) \tag{9}$$
$$\hat{v}_k = v_k/(1-\beta_2) \tag{10}$$
$$\theta_k = \theta_{k-1} - \alpha \hat{m}_k/\sqrt{(\hat{v_k}+\epsilon)} \tag{11}$$

g_k, v_k, θ_k はベクトルであり, α, β_1, β_2 はスカラーである. ベクトルの 2 乗はアダマール積を表し, ベクトル同士の割り算は要素ごとの割り算を表し, m_0 と v_0 は 0 とする. ハイパーパラメータの値は $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \eta = 10^{-3}$ が良いとされている [11].

損失関数の与え方により勾配降下法は, バッチ学習, 逐次学習, ミニバッチ学習に分けられる. バッチ学習法は, 損失関数を (12) とする勾配降下法である. ただし, n は全データ数とし, 学習データとその教師信号は $(x_i, y_i)(i = 1, \cdots, n)$ とする.

$$l(\theta) = \Sigma_{i=1}^{n} E(x_i, y_i, \theta) \tag{12}$$

確率的勾配降下法は, バッチ学習と比較し, 計算量が小さい勾配法であり, 逐次学習とミニバッチ学習は, 確率的勾配降下法の一種である. 逐次学習は, ランダムに選んだ学習データを用いて損失関数を (13) とする勾配降下法である. ただし, ランダムに選ばれた学習データとその教師信号は (x_c, y_c) とする.

$$l(\theta) = E(x_c, y_c, \theta) \tag{13}$$

ミニバッチ学習は, 全データからランダムに選ばれた複数個の学習データを用いて損失関数を (14) とした勾配降下法である. ただし, ランダムに選ばれた c 個の学習データを $(x_{n_i} y_{n_i})(i = 1, \cdots, c)$ とする.

$$l(\theta) = \Sigma_{i=1}^{c} E(x_{n_i}, y_{n_i}, \theta) \tag{14}$$

誤差逆伝播法は, パラメータ $W_l, b_l (l = 1, \cdots, N-1)$ に対する勾配を効率良く求める手法である. W_l, b_l のどちらにも同じアルゴリズムを適用できるため, ここでは W_l の勾配について示す. $\delta_l \equiv \partial E/\partial u_l$ とすると, δ_{l+1} と δ_l の関係は次の式のようになる.

$$\delta_{N-1} = \frac{\partial E}{\partial u_{N-1}} = \frac{\partial z_N}{\partial u_{N-1}} \frac{\partial E}{\partial z_N} = \frac{\partial \sigma_{N-1}(u_{N-1})}{\partial u_{N-1}} \frac{\partial E}{\partial z_N} \tag{15}$$

$$\delta_l = \frac{\partial E}{\partial u_l} = \frac{\delta u_{l+1}}{\delta u_l} \frac{\delta E}{\delta u_{l+1}} = \frac{\partial z_{l+1}}{\partial u_l} \frac{\partial u_{l+1}}{\partial z_{l+1}} \delta_{l+1} = \frac{\partial \sigma_i(u_l)}{\partial u_l} W_{l+1}^T \delta_{l+1} \tag{16}$$

ここで, $W_l = (\cdots, w_j, \cdots)^T$ とおき、$u_l = W_l z_l$ に注意すると,

$$\frac{\partial E}{\partial w_j} = \frac{\partial u_{lj}}{\partial w_j} \frac{\partial E}{\partial u_{lj}} = z_l \delta_{lj} \tag{17}$$

ただし, u_{lj} は u_l の j 番目の要素とし,δ_{lj} は δ_l の j 番目の要素とする.
誤差逆伝播法は, (15) と (16) を繰り返し用いることで δ_l を求め, (17) で求める勾配を求める方法である.

2.2 word2vec

深層学習における自然言語処理分野では, 基本的な処理単位が文字・文・文章であるため, 画像認識や音声認識の分野のように直接データをニューラルネットワークに入力することができない. このため, 単語は分散表現と呼ばれるベクトル表現に置き換えられる. 分散表現の獲得の仕方 [9, 12, 13] には様々なものが考案されているが, 本論文では word2vec[9] の Skip-gram の分散表現を [19] でダウンロードして用いる.
任意の文字列に対して確率を定義する確率モデルは言語モデルと呼ばれ, word2vec は 2 つの言語モデル Skip-gram, CBoW の総称である. Skip-gram と CBoW は, 3 層のニューラルネットワークであり, 分散表現を獲得することを目的として作られた言語モデルである. 以下では, 扱う単語列を $(w_1, \cdots, w_N)$ とし, W をコーパスに含む全ての単語の集合とし, $W^{'}$ は W に含まれる単語数とする.

2.2.1 Skip-gram

Skip-gram では, 損失関数は条件付き確率 $p(w_{i+j}|w_i)$ を用いて (18) と定義される.

$$-\sum_{i=1}^{W} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{i+j}|w_i) \tag{18}$$

c はウィンドウサイズと呼ばれ, w_{i+j} の $i+j$ が 0 以下または $N+1$ 以上のとき, $\log p(w_{i+j}|w_i) = 1$ とする.
条件付き確率 $p(w_{i+j}|w_i)$ は (19) と定義される.

$$p(w_{i+j}|w_i) = \frac{\exp(v_{w_{i+j}}^{'T} v_{w_i})}{\sum_{w \in W} \exp(v_w^{'T} v_i)} \tag{19}$$

ここで, v_{w_i} は w_i に対応する求める分散表現とし, $v_{w_i}^{'}$ は w_i に対する文脈上での分散表現とする.

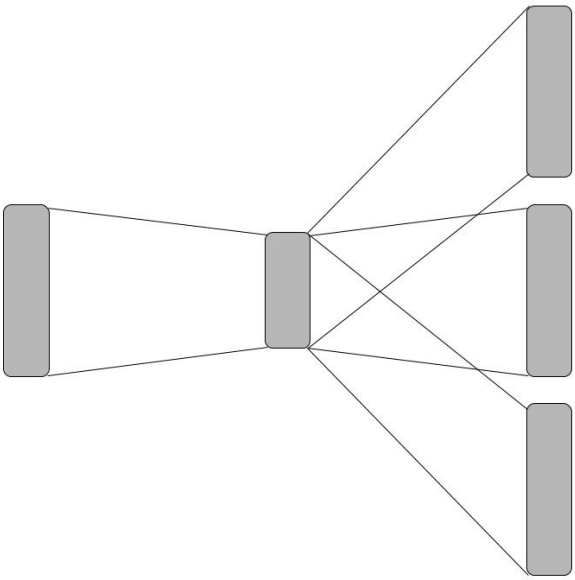

図 2　Skip-gram の模式図: 左から順に, 入力層, 中間層, 出力層を表す. 各単語は, one-hot ベクトルと呼ばれる 1 つの要素が 1 で他は 0 となる固有のベクトルに置き換えられ, ニューラルネットワークに入力される. 入力層に入れられた対象単語の one-hot ベクトルは, 中間層で対応する分散表現に変換され, 出力層で周辺のウィンドウサイズ分の単語の予測が条件付き確率の形で出力される.

2.2.2　CBoW

CBoW では, 損失関数を条件付き確率 $p(w_i|w_{i-c},\cdots,w_{i+c})$ を用いて (20) と定義される.

$$-\sum_{t=1}^{W}\log\, p(w_i|w_{i-c},\cdots,w_{i+c}) \tag{20}$$

c はウィンドウサイズと呼ばれ, $w_{i-c},\cdots,w_{i+c}$ の内, 1 つの添字が 0 以下または $N+1$ 以上のとき, $p(w_i|w_{i-c},\cdots,w_{i+c})=1$ とする.

条件付き確率は, $p(w_i|w_{i-c},\cdots,w_{i+c})$ は以下の式により定義する.

$$p(w_i|w_{i-c},\cdots,w_{i+c})=\frac{\exp(\sum_{-c\leq j\leq c,j\neq 0} v_{w_{i+j}}^{'T} v_{w_i})}{\sum_{w\in W}\exp(\sum_{-c\leq j\leq c,j\neq 0} v_w^T v_{w_{i+j}}^{'})} \tag{21}$$

ここで, v_{w_i} は w_i に対応する求める分散表現とし, $v_{w_i}^{'}$ は w_i に対する文脈上での分散表現とする.

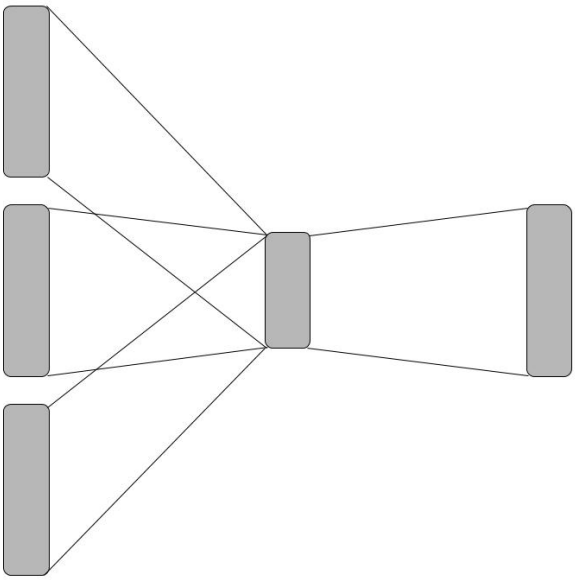

図 3　CBoW の模式図: 左から順に, 入力層, 中間層, 出力層を表す. 各単語は, one-hot ベクトルと呼ばれる 1 つの要素が 1 で他は 0 となる固有のベクトルに置き換えられ, ニューラルネットワークに入力される. 入力層に入れられた文脈の単語の one-hot ベクトルは, 中間層で各単語に対応する分散表現を足したものに変換され, 出力層で対象単語の予測が条件付き確率の形で出力される.

2.3　RNN

再帰型ニューラルネットワーク (Recurrent Neural Network, RNN) は, 音声信号処理, 自然言語処理, 機械翻訳の分野で時系列データを扱う際に用いられるニューラルネットワークである. RNN では, 各層でデータが入力される. RNN の各層では, 現時刻の入力ベクトルと以前の層の情報を含んだ前時刻の隠れ状態ベクトルを使って, 現時刻の隠れ状態ベクトルを出力する。RNN の学習には勾配降下法が用いられるが, 勾配を求めるときには誤差逆伝搬法と同様の考えを用いた通時的誤差逆伝播法が用いられる. 打ち切り型通時的誤差逆伝播法は, 手動で設定した中間層で購買計算を止める手法であり, 本論文では鍵となる.

本節では最も単純な構造である単純再帰型ネットワークについて説明した後, 長時間の情報を保持できる LSTM について説明し, LSTM の一種である GRU について説明する.

2.3.1　単純再帰型ネットワーク

単純再帰型ネットワークは, 次のような線型写像と非線形写像を繰り返し行う関数である.

$$u_t = Wx_t + \tilde{W}h_{t-1} \tag{22}$$

$$h_t = \sigma(u_t + b) \tag{23}$$

$$y_t = \sigma_{out}(W_{out}h_t + b_{out}) \tag{24}$$

ここで, 時刻 $t(t = 1, \cdots, N)$ での入力ベクトルは x_t, 隠れ状態ベクトルは h_t, 出力層の出力ベクトルは y_t とし, $z_0 = 0$ とする. また, 入力層と中間層間の変換行列を W, 中間層と出力層間の変換行列を W_{out}, 隠れ状態ベクトルに対する変換行列を $\tilde{W}$ とする. σ、σ_{out} は, 活性化関数とする.
RNN では, 勾配消失問題と呼ばれる, 長時間の情報を踏まえるように学習できない問題がある. これは, 通時的誤差逆伝播法で時間を遡るにつれて, 勾配が 0 に近くなり勾配降下法でパラメータがほとんど更新されないことによって起こる.

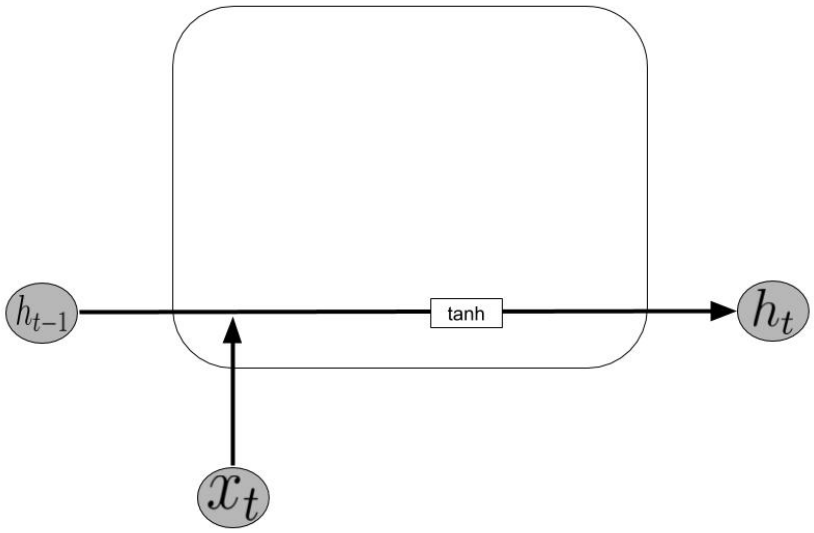

図 4　RNN における一層の内部模式図:　h_{t-1} は, 時刻 $t-1$ における隠れ状態ベクトルを表し, x_t は, 時刻 t における単語の分散表現を表す. h_{t-1} と x_t を結合したものが, W によって線形変換され, tanh 関数によって非線形変換される.

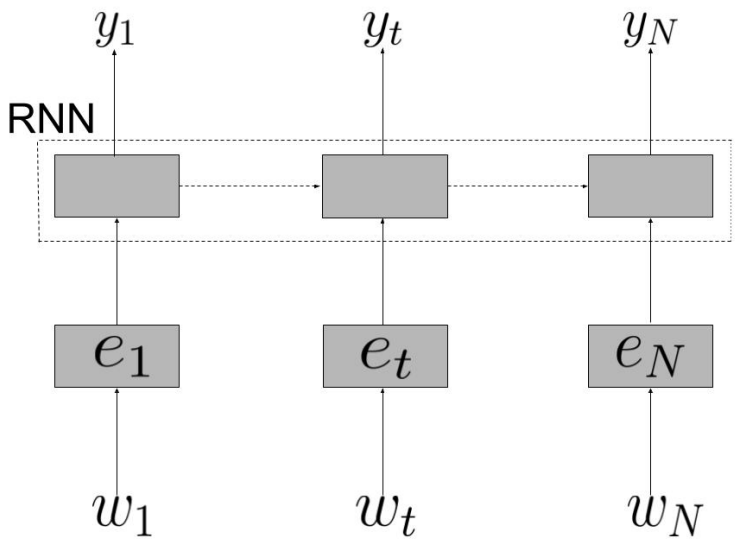

図 5　自然言語処理における RNN の模式図:　各層の関係を表した図になっている．時刻 t において, w_t は単語の one-hot ベクトルを表し, e_t は w_t の分散表現を表し, y_t は出力されるベクトルを表す．タスクに用いる場合は, y_t が用いられるが, 出力される全ての y_t を用いないタスクもある．本論文で扱う感情分析タスクでは, y_N のみを用いる．y_N のみを用いる場合は, many-to-one モデルと呼ばれ, 全ての y_t を用いる場合は, many-to-many モデルと呼ばれる．

2.3.2　LSTM

LSTM[14] は, RNN の勾配消失問題を軽減するように設計された構造であり, (25) から (30) で表される．LSTM では, RNN の h_t とは別にセルと呼ばれる過去の情報を保持するベクトルと入力ゲート, 忘却ゲート, 出力ゲートと呼ばれる 3 つのゲートを導入する．ただし, $*$ は, アダマール積を表す.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{25}$$
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{26}$$
$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C) \tag{27}$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{28}$$
$$y_t = \sigma(W_y[h_{t-1}, x_t] + b_y) \tag{29}$$
$$h_t = y_t * \tanh(C_t) \tag{30}$$

以下では, 図 6 を用いて LSTM の構造を解説する．セルは過去の情報を保持するために用いられ, ゲートは選択的に情報を通すために用いられる．(25) の忘却ゲート f_t は, 図では緑色の矢印で用いられ, C_{t-1} から C_t に保持する情報を取捨選択するのに用いられる．(27) の $\tilde{C}_t$ は C_t に加えられる新たなセルの候補値であり, (26) の入力ゲート i_t は, 図では赤の矢印で用いられ, $\tilde{C}_t$ から新たな

セル C_t に加える情報を取捨選択するのに用いられる. 入力ゲートと忘却ゲートにより, 新たなセル C_t は (28) となる. (29) の出力ゲートは, 図では青の矢印で用いられ, セル状態のどの部分を h_t として出力するかを選択するのに用いられる. (30) の h_t は, セル状態を tanh 関数を用いて-1 と 1 の間に圧縮したものと出力ゲートにアダマール積を施すことで実装される.

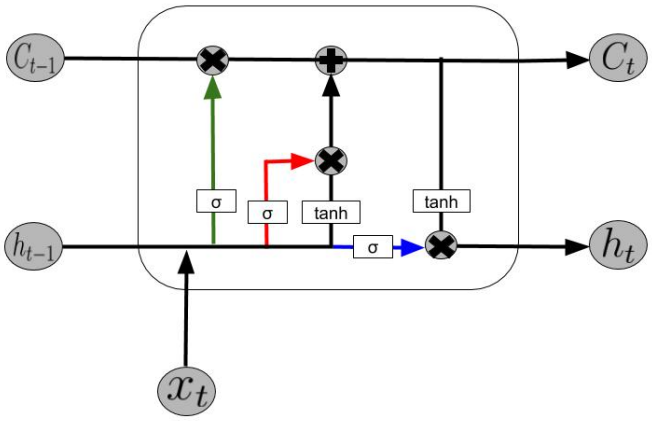

図 6　LSTM の一層の内部模式図: σ はシグモイド関数を, tanh は tanh 関数を表している. $\times$ は, アダマール積を表し, $+$ は, ベクトルの足し算を表す. 緑の矢印は忘却ゲートの流れを表し, 赤の矢印は入力ゲートの流れを表し, 青の矢印は出力ゲートの流れを表す.　時刻 t において, x_t は入力されるベクトルを表し,　C_t はセルを表し, h_t は隠れ状態ベクトルを表す.

2.3.3 GRU

GRU(Gated Recurrent Unit)[15] は, LSTM を簡略化した構造であり, セルを使わずに長期の情報を隠れ状態ベクトルに集約できる.

$$z_t = \sigma(W_z[h_{t-1}, x_t] + b_z) \tag{31}$$
$$r_t = \sigma(W_r[h_{t-1}, x_t] + b_r) \tag{32}$$
$$\tilde{h}_t = \tanh(W[r_t * h_{t-1}, x_t]) \tag{33}$$
$$h_t = (1 - u_t) * h_{t-1} + u_t * \tilde{h}_t \tag{34}$$

以下では, 図 7 を用いて GRU の構造を解説する. GRU は再設定ゲート r_t と更新ゲート u_t の二つのゲートを用いて長期記憶を保持する. (31) の更新ゲート u_t は, 図では青の矢印で用いられ, 一時刻前の隠れ状態ベクトルを減衰させ保持する情報を取捨選択するのに用いられる. 新たなセル h_t は, 新しいセル候補である $\tilde{h}_t$ と h_{t-1} と更新ゲートを用いて, (34) とされる. 再設定ゲート r_t は,

図では緑色の矢印で用いられ, 新たなセルの候補値 $\tilde{h}_t$ の生成にあたり, 一時刻前の隠れ状態ベクトルで保持する情報を取捨選択するのに用いられる. LSTM と GRU のどちらが優れているかは分かっていないが, GRU は LSTM よりも少ない計算量・空間量で済むため, 広く用いられている.

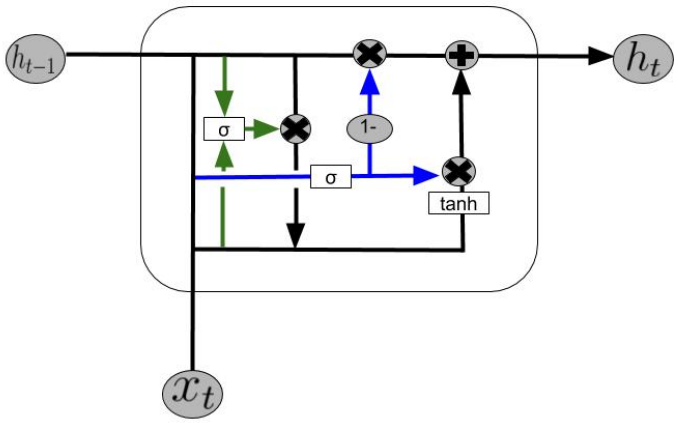

図 7　GRU の一層の内部模式図: σ は, シグモイド関数を, tanh は, tanh 関数を表している. $\times$ は, アダマール積を表し, $+$ は, ベクトルの足し算を表す. 青の矢印が更新ゲートに関わる操作を表し, 緑の矢印が再設定ゲートに関わる操作を表す. 時刻 t において, x_t は, 入力されるベクトルを表し, h_t は隠れ状態ベクトルを表す.

2.4 敵対的学習

敵対的学習 [6] は, パラメータを学習する際に, 学習データの他に敵対的事例と呼ばれる学習データに人工的な処理を施し作ったデータを用いることで汎化性を上げる正則化手法である.

機械学習の手法では, 与えられたデータの近傍点は同じ性質を持つとされる. しかし, 全ての近傍点では必ずしもそうではないことが知られ [7], 敵対的事例は, 与えられたデータと異なる性質を持つ近傍点のことである. 通常, 敵対的事例は, データに人為的な微小な摂動を加えることによって生成され, 悪意を持った微小な摂動は, 入力データに対する損失値を求め損失関数が増大する勾配方向の値が用いられる. [7] では敵対的事例について次のことも報告されている.

- あるデータで学習したモデルの敵対的事例は, 異なるデータセットで学習したモデルでも誤分類される.
- ロジスティック回帰のような線形分類器も敵対的事例に脆弱性を示す.

敵対的学習は, [6] によって考案された, 敵対的事例を学習データとして用いることで, 頑強な重みを学習し汎化性を上げる学習法である.

下にこの擬似コードを示す. 通常の学習との違いは, 擬似コードにおいて 5-7 で敵対的事例を生成している点と, 損失関数 $\tilde{E}(x, y, \theta)$ に [6] によって提案された敵対的事例を用いる際の損失関数 (35) を用いる点である. 本論文では簡単のため, $\alpha = 0$ を用いる. ただし,w_1 は勾配の重みとする.

$$\tilde{E}(x, y, \theta) = \alpha E(x, y, \theta) + (1 - \alpha) E(x + \epsilon \nabla_x E(x, y, \theta), y, \theta) \tag{35}$$

自然言語処理における敵対的学習では, 分散表現に微小な摂動を加える形で実装される. [8] では, 自然言語処理における敵対的学習が調べられ, 次のことが報告されている.

- 学習データが少なくモデルが過学習する容量がある場合, 精度が上がらなくてもテスト損失は大幅に改善する.
- 学習データが多くなるにつれて, 敵対的学習の機能は弱まり, 通常の学習と変わらなくなる.

Algorithm 1 Adversarial Training

Input: 入力データ: X, モデルのパラメータ:θ, 教師ラベル:y, 微小な摂動の重み w_1, 勾配の重み w_2

Output: モデルのパラメータ θ

for $i = 1$ to epoch_size **do**
  for j=1 to iteration_size **do**
    $batched\_X \leftarrow X[(j-1) * batch\_size : j * batch\_size]$
    $batched\_y \leftarrow y[(j-1) * batch\_size : j * batch\_size]$
    $prediction\_tmp \leftarrow model(batched\_X)$
    $loss\_tmp \leftarrow loss(prediction\_tmp, batched\_y)$
    $batched\_X += w_1 * \nabla_{batched\_X} loss\_tmp$
    $prediction \leftarrow model(batched\_X)$
    $loss \leftarrow loss(prediction\_tmp, prediction, batched\_label)$
    $\theta -= w_2 * \nabla_\theta loss$
  end for
end for

3　提案手法

本論文では, RNN の many-to-one モデル (図 8) を用いるタスクにおける効率の良い敵対的学習を提案する.　また, この提案手法は, RNN の many-to-many モデル (図 5) でも容易に拡張することができる. 本論文のモチベーションは, RNN を用いて長時間のデータや大量のデータで敵対的学習を行うときに, 通常の約 2 倍以上かかる時間を短縮することである.

通常, 敵対的学習を用いる場合, 図 9 のように微小な摂動を各単語ベクトルに対して足したベクトルを RNN に入力する. 本研究では, 全ての単語ベクトルに対して微小な摂動を足すのではなく, 図 10 のように y_N に近い一部の単語ベクトルにのみ微小な摂動を加えることを提案する. これを行うことによって, 深部までの勾配を計算する必要がなくなるため計算時間を短縮できることが期待でき, 勾配消失問題により通常の敵対的学習と同様の精度が得られることが期待できる. 実装にあたっては, 打ち切り型通時的誤差逆伝播法を用いて摂動のための勾配計算を行う.

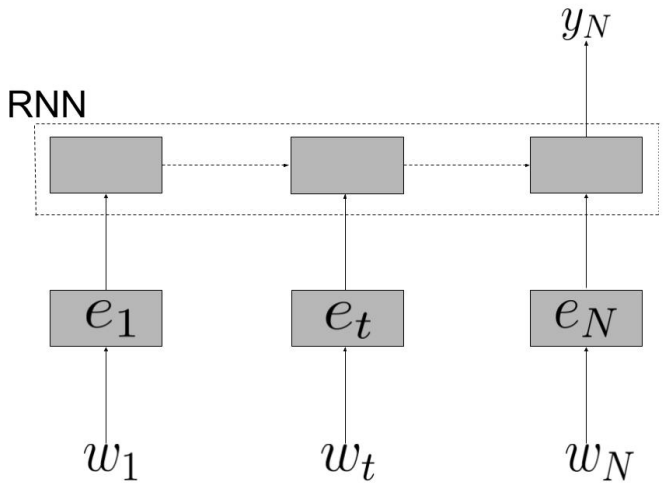

図 8　many-to-one モデルにおける通常の学習法の模式図:　時刻 t において, w_t は各単語の one-hot ベクトルを表し, e_t は対応する単語の分散表現を表し, y_t は出力されるベクトルを表す.

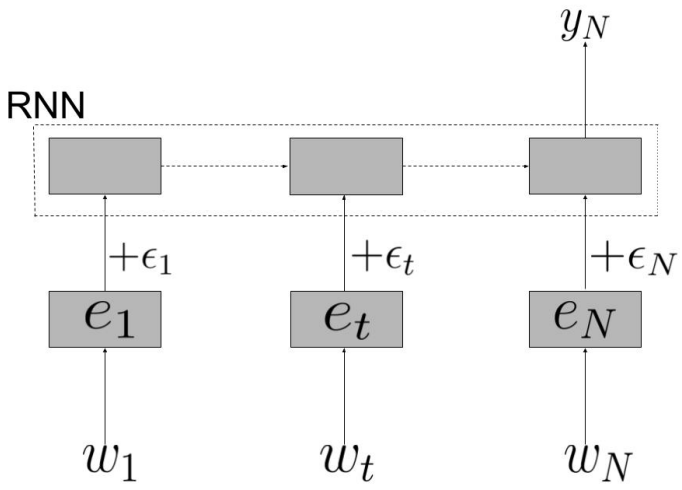

図 9　many-to-one モデルにおける敵対的学習法の模式図:時刻 t における単語の分散表現に微小な摂動 ϵ_t が加えられた形になっている．時刻 t において, w_t は各単語の one-hot ベクトルを表し, e_t は対応する単語の分散表現を表し, y_t は出力されるベクトルを表す.

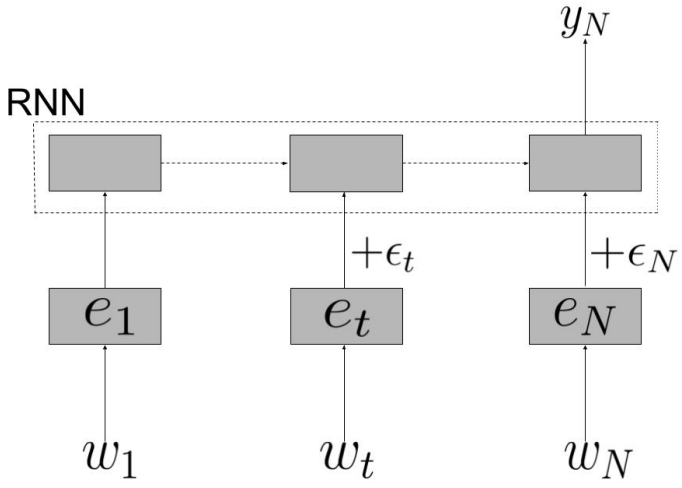

図 10　many-to-one モデルにおける提案手法の模式図: 部分的な単語の分散表現に微小な摂動 ϵ_t が加えられた形になっている．時刻 t において, w_t は各単語の one-hot ベクトルを表し, e_t は対応する単語の分散表現を表し, y_t は出力されるベクトルを表す.

4　実験

この章では, 3 章で提案した手法の性能を調べる. 4.1 節では, 用いたデータセット MR, SST-1, SST-2, TREC について説明する. 4.2 節では, 2.4 節で述べた微小な摂動の重み w_1 で最も性能が良かった結果を示し, 必ずエポック毎の計算時間が削減され, 全てのデータセットにおいて, 同程度またはそれ以上の性能が得られることを示す. 4.3 節では, w_1 による性能の変化と正則化手法である Dropout[20] とバッチ正規化 [21] との併用について検証する. 4 章では, 各ハイパーパラメータを次のように共通して設定している. 学習には, Adam を用い, $\alpha = 1.0 \cdot 10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ とした. バッチ数は 48 とし, 用いる RNN は GRU で h_t の次元は 200 とした. Dropout またはバッチ正規化は y_N に施し, Dropout を用いる場合は, 選出確率を 0.5 とした. また, 提案手法では, 各コーパスの平均長の勾配を微小な摂動として付与した.

4.1　用いたデータセットについて

本研究では, 文章分類タスクを用いて実験する. 感情分析タスクは, 文章をいくつかの与えられたカテゴリーに分類する文章分類タスクの一種であり, カテゴリーがポジティブ, ニュートラル, ネガティブのような感情となっている. MR, SST-1, SST-2 は感情分析タスクであり, TREC は文章分類タスクである. [8] において, データ数が少ない場合に敵対的学習は通常の学習より性能が良くなることが示されているため, MR, SST-1, SST-2 では, 学習データ数は与えられている学習データの 50% をランダムに選んだものを用いることとし, TREC には下のデータ数を与えられた学習データからランダムに選んだものを用いることにした.

データセットの基本情報および統計情報は以下の通りである. ただし, 統計情報の各データは小数第一位で四捨五入している. 各単語に対する単語ベクトルは, Skip-gram によって学習された [19] を用いている.

- MR データセット:一文の映画評論に対しポジティブかネガティブを当てるタスク [16]
- SST-2: 一文に対しポジティブかネガティブかニュートラルのどれかを当てるタスク [17]
- SST-1: SST-2 と同じデータセットであるが、一文に対しベリーポジティブかポジティブかニュートラルかネガティブかベーリーネガティブの 5 つの分類から当てるタスク [17]
- TREC: 質問を省略 (ABBREVIATION), 説明 (DESCIPTION), 存在 (ENTITY), 場所 (LOCATION), 人間 (HUMA), 数値 (NUMERIC) の 6 つのカテゴリーに分類するタスク [18]

表 1　各データセットにおけるクラス数と文長の統計情報

データセット	クラス数	平均文長	文長の標準偏差	最長文長
MR	2	21	9	59
SST-2	3	19	9	56
SST-1	5	19	9	56
TREC	6	10	4	37

表 2　各データセットにおける統計情報

データセット	用いた分散表現の数	全単語数	学習データ数	検証データ数	テストデータ数
MR	15822	21114	5331	1775	3556
SST-2	15761	19486	4272	1101	2210
SST-1	15761	19486	4272	1101	2210
TREC	7392	8944	1500	500	1000

4.2　結果

4.1 節で述べたデータセット MR, SST-1, SST-2, TREC を用いて通常の学習と敵対的学習と提案手法を比較した結果が, 表 3, 表 4, 表 5 である. MR では $w_1 = 10$, SST-1 では $w_1 = 1$, SST-1 では $w_1 = 10$, TREC では $w_1 = 10$ を用いた結果を示している. 表には, RNN におけるパラメータの初期値依存性をなくすために, 各条件の元で学習を 5 回行って得られた平均値を載せてある. 提案手法は, Dropout との併用で性能が向上したため, これらを組み合わせた結果で比較してある. 表 3 では, 各データセットにおける 1 エポックあたりに必要となる計算時間を示してある. 表 3 により以下の考察が得られる. 通常の学習と Dropout を除き, 最も性能が良かったものには, 太字にしてある.

- 提案手法は, 敵対的学習より短い計算時間で 1 エポックあたりの学習ができ, 敵対的学習で問題となる学習時間を短縮できている.
- 提案手法による計算時間は, 敵対的学習と通常の学習による時間の平均値に近い値になっていることがわかる. これは, 提案手法ではコーパスの平均文長の勾配を付与しているため, 通時的誤差逆伝播法に必要となる計算量が半減するためであると考えられる.　データ量が多い場合やコーパスの平均文長が大きい場合, 学習アルゴリズムにおいて通時的誤差逆伝播法が最も計算量が大きくなるため, 平均文長を用いた提案手法は, 通常の敵対的学習の計算時間を約 25% 削減できた手法となると考えられる.

また, エポック数も, 通常の敵対的学習を行なう場合よりも少なく済むことが実験で観察された. これは, 敵対的事例を作成する際に深部の勾配も用いる敵対的学習の場合, 損失関数の値が提案手法

よりも大きくなるために学習が進みにくくなることに起因すると考えられる．以上のことにより，時間を短縮する目的は，提案手法で達成できていると考えられる．

表 3　各データセットにおける 1 エポックの学習に必要な時間 (s)

データセット	通常の学習	Dropout	敵対的学習	敵対的学習 +Dropout	提案手法 +Dropout
MR	10.08	10.17	18.78	18.75	**14.86**
SST-2	11.36	10.80	21.68	22.24	**16.39**
SST-1	11.33	11.59	21.43	21.47	**15.89**
TREC	2.19	2.13	3.78	3.85	**2.85**

表 4, 表 5 では各データセットにおける精度とテストデータセットにおける損失値を示してある．表 4 と表 5 により以下の考察が得られる．最も性能が良かったものには，太字としている．

- 表 4 において通常の学習と Dropout の精度を比較すると，Dropout を施すことで，SST-1 と TREC は性能が落ち，MR と SST-2 は，性能がわずかに向上することがわかる．しかし，敵対的学習や提案手法ほどの性能の向上は見られない．表 5 において通常の学習と Dropout の損失値を比較すると，MR を除いたデータセットで性能が悪くなっている．また，敵対的学習や提案手法ほどの性能の向上は見られない．これは，用いたデータセットのデータ量を与えられたものより少なくしているため，正則化機能を持つ Dropout がうまく働かなく，データ拡張手法である敵対的学習と提案手法ではうまく働いているためであると考えられる．
- 表 4 により，通常の敵対的学習と Dropout を併用することで精度が落ちることがわかる．しかし，後の 4.2 節で示すが，提案手法では Dropout との併用で性能が向上する．データ量が多い場合は，敵対的学習の正則化機能が弱まることが [8] によって調べられているため，いかなる量のデータセットでも汎化性を持てるように，他の正則化手法との併用が重要となってくるため，提案手法は，この面で通常の敵対的学習より優れていると考えられる．
- 表 5 では，提案手法は SST-1 で最も性能が良くなっているが，その他のデータセットにおいても，敵対的学習と敵対的学習 +Dropout を用いたものと同程度の性能を示していることがわかる．

したがって，提案手法は，時間を短縮できるだけでなく，　通常の敵対的学習と同程度の性能を得ることができることがわかる．

表 4　各データセットにおいて最も性能が良かった精度

データセット	通常の学習	Dropout	敵対的学習	敵対的学習 +Dropout	提案手法 +Dropout
MR	0.7616	0.7664	**0.7822**	0.7808	0.7770
SST-2	0.6816	0.6828	**0.6940**	0.6903	0.6903
SST-1	0.4504	0.4499	0.4620	0.4586	**0.4624**
TREC	0.8114	0.8064	0.8578	0.8554	**0.8602**

表 5　各データセットにおいて最も性能が良かったテストデータの損失値

データセット	通常の学習	Dropout	敵対的学習	敵対的学習 +Dropout	提案手法 +Dropout
MR	0.4955	0.4871	0.4625	**0.4573**	0.4673
SST-2	0.7583	0.7591	**0.7531**	0.7555	0.7540
SST-1	1.2549	1.2587	1.2294	1.2369	**1.2279**
TREC	0.6145	0.6251	**0.4383**	0.4415	0.4396

4.3　パラメータ解析

4.3.1　摂動のパラメータの解析

この節では, 2.4 節で与えた微小な摂動の重み w_1 の解析を行う. 表 6 では, テストデータの損失値に関する結果を, 表 7 では精度に関する結果を示してある. 表 6 と表 7 により以下の考察が得られる. 各データセットの手法で, 最も性能が良かったものは太字にしてある.

- 表 6 により, 全てデータセットで $w_1 = 0.01, 100$ では, 提案手法の性能が通常の敵対的学習の性能を下回っていることがわかる. 他に行った実験でも同様のことが観測されたため, 提案手法は鋭敏性を持つことがわかる. 提案手法を用いる場合は, w_1 を 0.1 から 10 程度に取ると良いことがわかる.
- 表 6 により, SST-1, SST-2, TREC における提案手法の $w_1 = 1.0, 10$ での損失値は, 通常の敵対的学習以上の性能が得られることがわかる. また, w_1 として最適な値は, データセットによって変わるが, $w_1 = 1, 10$ のどちらかで最も性能が良くなることが見て取れるため, w_1 として最適な値は, 1 から 10 程度の値だと考えられる.
- 時間的な観点では, w_1 が大きい方がよりエポック数を必要とすることが観測された. TREC では, $w_1 = 0.01$ のときに必要なエポック数は約 10 程度であったが, $w_1 = 100$ では, 40 エポック程度必要であった. したがって, 実際に用いる場合は, 小さい w_1 が望ましいと考えられる.
- 表 7 において, MR を除くデータセットにおいて, 提案手法が最も性能が良い精度が得られることがわかる.　敵対的学習または提案手法を用いることで,1% 以上性能を上げることができ, TREC では, 5% 以上の性能の向上が見られた.

表 6　摂動の重み $w_1 = 0.01, 0.1, 1, 10, 100$ におけるテストデータの損失値

データセット	手法	$w_1 = 0.01$	$w_1 = 0.1$	$w_1 = 1.0$	$w_1 = 10$	$w_1 = 100$
MR	通常の学習	0.4955				
	Dropout	0.4871				
	敵対的学習 +Dropout	0.4907	0.4899	0.4751	**0.4573**	0.5030
	提案手法 +Dropout	0.4932	0.4859	0.4770	**0.4673**	0.5748
SST-2	通常の学習	0.7583				
	Dropout	0.7590				
	敵対的学習 +Dropout	0.7607	0.7540	**0.7407**	0.7555	0.8363
	提案手法 +Dropout	0.7520	0.7557	**0.7405**	0.7540	0.8756
SST-1	通常の学習	1.2549				
	Dropout	1.2587				
	敵対的学習 +Dropout	1.2495	1.2493	**1.2369**	1.2548	1.3423
	提案手法 +Dropout	1.2571	1.2419	**1.2279**	1.2485	1.4133
TREC	通常の学習	0.6145				
	Dropout	0.6251				
	敵対的学習 +Dropout	0.5791	0.5650	0.5152	**0.4415**	0.5246
	提案手法 +Dropout	0.6434	0.5837	0.4958	**0.4396**	0.5995

表 7　摂動の重み $w_1 = 0.01, 0.1, 1, 10, 100$ におけるテストデータの精度

データセット	手法	$w_1 = 0.01$	$w_1 = 0.1$	$w_1 = 1.0$	$w_1 = 10$	$w_1 = 100$
MR	通常の学習	0.7616				
	Dropout	0.7664				
	敵対的学習 +Dropout	0.7646	0.7655	0.7773	**0.7808**	0.7499
	提案手法 +Dropout	0.7652	0.7667	0.7740	**0.7770**	0.7478
SST-2	通常の学習	0.6816				
	Dropout	0.6828				
	敵対的学習 +Dropout	0.6800	0.6835	**0.6914**	0.6903	0.6871
	提案手法 +Dropout	0.6859	0.6830	0.6872	**0.6903**	0.6792
SST-1	通常の学習	0.4504				
	Dropout	0.4499				
	敵対的学習 +Dropout	0.4536	0.4542	0.4586	**0.4593**	0.4322
	提案手法 +Dropout	0.4490	0.4597	**0.4624**	0.4614	0.4016
TREC	通常の学習	0.8114				
	Dropout	0.8064				
	敵対的学習 +Dropout	0.8260	0.8204	0.8292	**0.8554**	0.8216
	提案手法 +Dropout	0.8132	0.8082	0.8366	**0.8602**	0.8082

4.3.2　他の正則化手法との併用についての解析

この節では, 敵対的学習と提案手法を正則化手法である Dropout やバッチ正規化と併用することについて考察する. 最も性能が良かったものは, 太字にしてある. 表 8, 9, 10, 11 では, 各データ

セットにおける手法別の結果を示している. 表より以下のような考察が得られる.

- 全てのデータセットにおいて敵対的学習と Dropout を併用することで性能が落ちることがわかるが, 提案手法では, Dropout と併用することで性能が上がることがわかる.
- バッチ正規化は提案手法と併用することで性能が著しく低下し, 通常の敵対的学習では提案手法ほどの性能の低下はみられない.

表 8　MR データにおける $w_1 = 10$ のときの他の正則化手法との併用

手法名	精度	テスト損失
通常の学習	0.7616	0.4955
Dropout	0.7664	0.4871
敵対的学習	0.7822	0.4625
敵対的学習 +Dropout	0.7808	**0.4573**
敵対的学習 + バッチ正規化	**0.7872**	0.4624
提案手法	0.7786	0.4650
提案手法 +Dropout	0.7770	0.4673
提案手法 + バッチ正規化	0.7728	0.4761

表 9　SST-1 データにおける $w_1 = 1$ のときの他の正則化手法との併用

手法名	精度	テスト損失
通常の学習	0.4504	1.2549
Dropout	0.4499	1.2587
敵対的学習	0.4620	1.2294
敵対的学習 +Dropout	0.4586	1.2369
敵対的学習 + バッチ正規化	0.4166	1.3102
提案手法	0.4579	1.2349
提案手法 +Dropout	**0.4624**	**1.2279**
提案手法 + バッチ正規化	0.4193	1.3238

表 10　SST-2 データにおける $w_1 = 10$ のときの他の正則化手法との併用

手法名	精度	テスト損失
通常の学習	0.6816	0.7583
Dropout	0.6828	0.7591
敵対的学習	**0.6940**	**0.7531**
敵対的学習 +Dropout	0.6903	0.7555
敵対的学習 + バッチ正規化	0.6784	0.7844
提案手法	0.6874	0.7676
提案手法 +Dropout	0.6903	0.7540
提案手法 + バッチ正規化	0.6808	0.7864

表 11　TREC データにおける $w_1 = 10$ のときの他の正則化手法との併用

手法名	精度	テスト損失
通常の学習	0.8114	0.6145
Dropout	0.8064	06251
敵対的学習	0.8578	0.4383
敵対的学習 +Dropout	0.8554	0.4415
敵対的学習 + バッチ正規化	0.8388	0.4769
提案手法	0.8572	0.4400
提案手法 +Dropout	**0.8602**	**0.4396**
提案手法 + バッチ正規化	0.7354	0.7819

5　まとめ

本論文では, RNN を用いた敵対的学習での学習時間を短縮することを目的とし, RNN の many-to-one モデルにおける計算量を軽減した敵対的学習を提案した. 実験ではタスクを文章分類タスクに絞り, 4 つのデータセットで提案手法の性能を調べた. その結果, 全てのデータセットで時間が短縮され, 通常の敵対的学習と同程度またはそれ以上の性能が得られた.

提案手法はハイパーパラメータによって大きく性能が異なる. このため, 本手法を用いる場合, 敵対的事例を作るときの摂動の重み w_1 は 1 から 10 程度の値が良く, w_1 が小さいほど学習に必要なエポック数が少なくなる. また, 提案手法は Dropout と併用することで性能が上がる.

今後の研究として, RNN の many-to-one モデルを用いる他のタスクでも性能を調べるつもりである. また, 敵対的学習の理論的な側面に興味を持ったので, 大学院では敵対的学習の最適化について研究したいと考えている.

謝辞

卒業論文を書くにあたって, 沢山の方に協力していただきました. ここに感謝の意を表します. 特に, 本論文の作成にあたり丁寧に指導してくださった下平英寿先生と, なかなか結果の出ない実験を最後まで暖かく見守り, 適切な指導をしてくださった下平研究室の先輩 Geewook Kim さんには, 大変感謝しています.

参考文献

[1] Xiang Zhang and Yann LeCun. Text understanding from scratch. arXiv preprint arXiv:1502.01710, 2015.

[2] Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. Improved relation classification by deep recurrent neural networks with data augmentation. arXiv preprint arXiv:1601.03651, 2016.

[3] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. The International Conference on Learning Representations (ICLR), 2018.

[4] Jason W. Wei, and Kai Zou. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019.

[5] Zhang, Dongxu, and Zhichao Yang. Word Embedding Perturbation for Sentence Classification. arXiv preprint arXiv:1804.08166, 2018.

[6] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. The International Conference on Learning Representations The International Conference on Learning Representations (ICLR), 2015.

[7] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. The International Conference on Learning Representations (ICLR), 2014.

[8] Petr Bělohlávek, Ondřej Plátek, Zdeněk Žabokrtský, and Milan Straka. Using Adversarial Examples in Natural Language Processing. Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC), 2018.

[9] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. The International Conference on Learning Representations (ICLR), 2013.

[10] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. Empirical Methods in Natural Language Processing (EMNLP), 2014.

[11] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980, 2014.

[12] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014.

[13] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics (TACL), 2016.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural Computation 9 (8): 1735–1780, 1997.

[15] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. arXiv:1406.1078, 2014.

[16] Pang and Lee. https://www.cs.cornell.edu/people/pabo/movie-review-data/, 2015.

[17] Sentiment Treebank. http://nlp.stanford.edu/sentiment/

[18] Li and Roth. http://cogcomp.cs.illinois.edu/Data/QA/QC/

[19] Pre-trained word vectors, https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?usp=sharing

[20] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15, page 1929–1958, 2014.

[21] Sergey Ioffe, and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv preprint arXiv:1502.03167, 2015.

付録 A　様々な正則化手法

A.1　早期終了規則

学習を進めるときに 1epoch 終了ごとに検証データで汎化性能を推定し, 汎化誤差が上昇し始める前に学習を終了する方法である.

A.2 Dropout

過学習を防ぎ, 汎化性を向上させる手法である. Dropout では, 学習中は一定の割合でユニットを不活性となる 0 とし, 認識時は全てのユニットを用いることで, 過学習を防ぐ. 学習されたモデルを用いる場合は, 元の出力にドロップアウトの割合を乗じる. 入力層では 80% 程度が, 中間層では 50% 前後が活性化するのが良いとされている.

A.3 バッチ正規化

ミニバッチ学習を行う際に生じる, ある層のバッチごとの特徴分布の偏りを解消するアルゴリズムである. c 個の入力データからなるミニバッチ $B = \{x_1, \cdots, x_c\}$ に対して, 平均 μ と分散 σ_B を計算し正規化し, スケーリング係数 γ とシフト量 β により, ミニバッチ間での分布が近くなるように調節する.

$$\mu_B = \frac{1}{c}\sum_{i=1}^{c} x_i \tag{36}$$

$$\sigma_B^2 = \frac{1}{c}\sum_{i=1}^{c}(x_i - \mu_B)^2 \tag{37}$$

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma_B^2 + \epsilon}} \tag{38}$$

$$y_i = \gamma\hat{x}_i + \beta \tag{39}$$

バッチ正規化を用いると, 大きな学習係数を用いることができ, NN の重みの初期値がそれほど性能に影響しなくなる利点がある.

※線に沿って切り取って下さい。

特別研究報告書

打ち切り型通時的誤差逆伝播法を用いた RNN の効率的な敵対的学習

指導教員　　下平英寿　教授

京都大学工学部情報学科
数理工学コース
平成 28 年 4 月入学

木村　拓仁

令和 2 年 1 月 29 日提出

打ち切り型通時的誤差逆伝播法を用いたRNNの効率的な敵対的学習

木村　拓仁

令和元年度

打ち切り型通時的誤差逆伝播法を用いた RNN の効率的な敵対的学習

木村　拓仁

摘要

RNN は文章分類や機械翻訳のような自然言語処理のタスクで広く用いられているが, データ量が少ない場合過学習を起こすため性能が低くなってしてしまう. このため, 学習データに人工的な処理を施しデータ量を増やすデータ拡張が用いられる. 敵対的学習は敵対的事例と呼ばれる分類器が誤判別するデータを生成し学習を行う手法であり, RNN を用いた自然言語処理のタスクに適用することで, テスト損失が大幅に改善することは先行研究で調べられている. しかし, RNN を用いた敵対的学習では, 学習に通常以上の時間がかかってしまう問題がある.

本研究では, RNN を用いた敵対的学習での学習時間を短縮することを目的とし, RNN の many-to-one モデルにおける計算量を軽減した敵対的学習を提案した.　提案手法では, 敵対的事例の作成に必要な勾配計算に打ち切り型通時的誤差逆伝搬法を用いることで, 計算量の軽減を図った. この実験には, 文章分類タスクを用い４つのデータセットで性能を調べた. その結果, 時間が短縮できただけでなく, パラメータ次第では性能が向上することが確認できた.