ラビットチャレンジstudyai0027863

深層学習　後半

実装演習レポート北村裕斗　hiroto7018@gmail.com

ビデオ視聴学習者提出区分け	科目	章タイトル	1点100文字以上で要点のまとめ	実装演習結果キャプチャー又はサマリーと考察	「確認テスト」など自身の考察結果	演習問題や参考図書、修了課題など関連記事レポートによる加点
【d】 1つのURLで提出	深層学習day3 （基準点：11点）	Section1：再帰型ニューラルネットワークの概念	1点	1点	1点	1点
		Section2：LSTM	1点	1点	1点	1点
		Section3：GRU	1点	1点	1点	1点
		Section4：双方向RNN	1点	1点	1点	1点
		Section5：Seq2Seq	1点	1点	1点	1点
		Section6：Word2vec	1点	1点	1点	1点
		Section7：Attention Mechanism	1点	1点	1点	1点
	深層学習day4 （基準点：8点）	Section1：強化学習	1点	1点	1点	1点
		Section2：AlphaGo	1点	1点	1点	1点
		Section3：軽量化・高速化技術	1点	1点	1点	1点
		Section4：応用モデル	1点	1点	1点	1点
		Section5：Transformer	1点	1点	1点	1点
		Section6：物体検知・セグメンテーション	1点	1点	1点	1点

- 講義動画視聴およびソース実装演習を確実に実施しているかを審査します。
  - レポートが講義本筋に沿ってまとめてあるか。受講者自身の言葉で課題や気づきが記載されているか。
- 必須要件を満たさない場合、他の受講者のレポートのコピーなど不正が発覚した場合、差し戻しとします。
- 数式やコード演習結果の個々の間違いは審査に影響しません。なお、講師からのフィードバックはありませんのでご了承ください。
- 各科目以下の場合は差し戻し
  - 基準点以下
  - 実装演習を全く行なっていない場合（応用数学以外）

下記の要件の通り、区分ごとに単元レポートを作成、ご提出ください。

1）各章につき100文字以上で要点をまとめ、実装演習結果、確認テストについての自身の考察等を取り入れたレポートとする。

2）各科目の基準点が足りない場合、実装演習が不足する場合は差し戻しとする。

※各章は講義動画および講義資料（PDF）でご確認ください。

- **レポート掲載場所：自身のGitHub、SlideShareなどのBlogやWeb媒体で作成。お持ちでない方は新規アカウント（無料）を作成。**
- **レポート提出方法：レポート掲載ページのURLを【学習システム（LMS）内の指定フォーム】より提出。**

DAY3■再帰型ニューラルネットワークについて

Section1)再帰型ニューラルネットワークの概念

1-1RNN全体像1-1-1RNNとは

RNNとは

時系列データに対応可能な、ニューラルネットワークである

1-1-2時系列データ

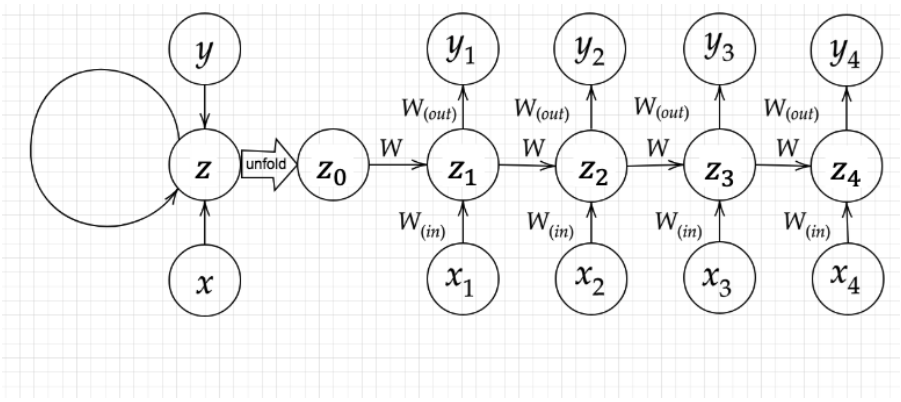
時系列データとは

時間的順序を追って一定間隔ごとに観察され，しかも相互に統計的依存関係が認められるようなデータの系列具体的な時系列データとは

・音声データ

・テキストデータ

1-1-3RNNについて



RNNの数学的記述

u[:,t+1] = np.dot(X, W_in) + np.dot(z[:,t].reshape(1, -1), W)

z[:,t+1] = functions.sigmoid(u[:,t+1])

np.dot(z[:,t+1].reshape(1, -1), W_out）

y[:,t] = functions.sigmoid(np.dot(z[:,t+1].reshape(1, -1), W_out))

確認テスト

　RNNのネットワークには大きくわけて3つの重みが ある。1つは入力から現在の中間層を定義する際に かけられる重み、1つは中間層から出力を定義する際に かけられる重みである。残り1つの重みについて説明せよ。

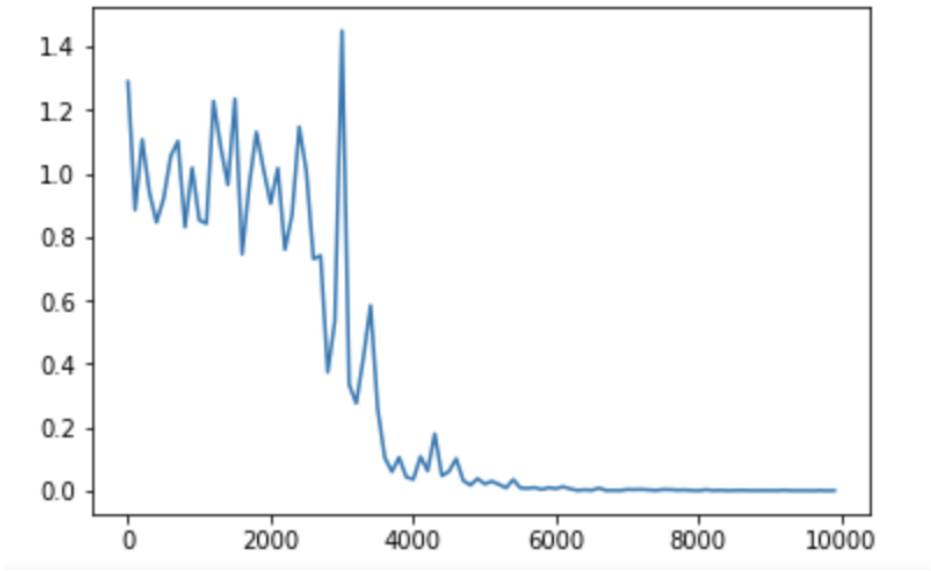
→ 中間層から次の中間層にかけられる重みがある

RNNの特徴とは

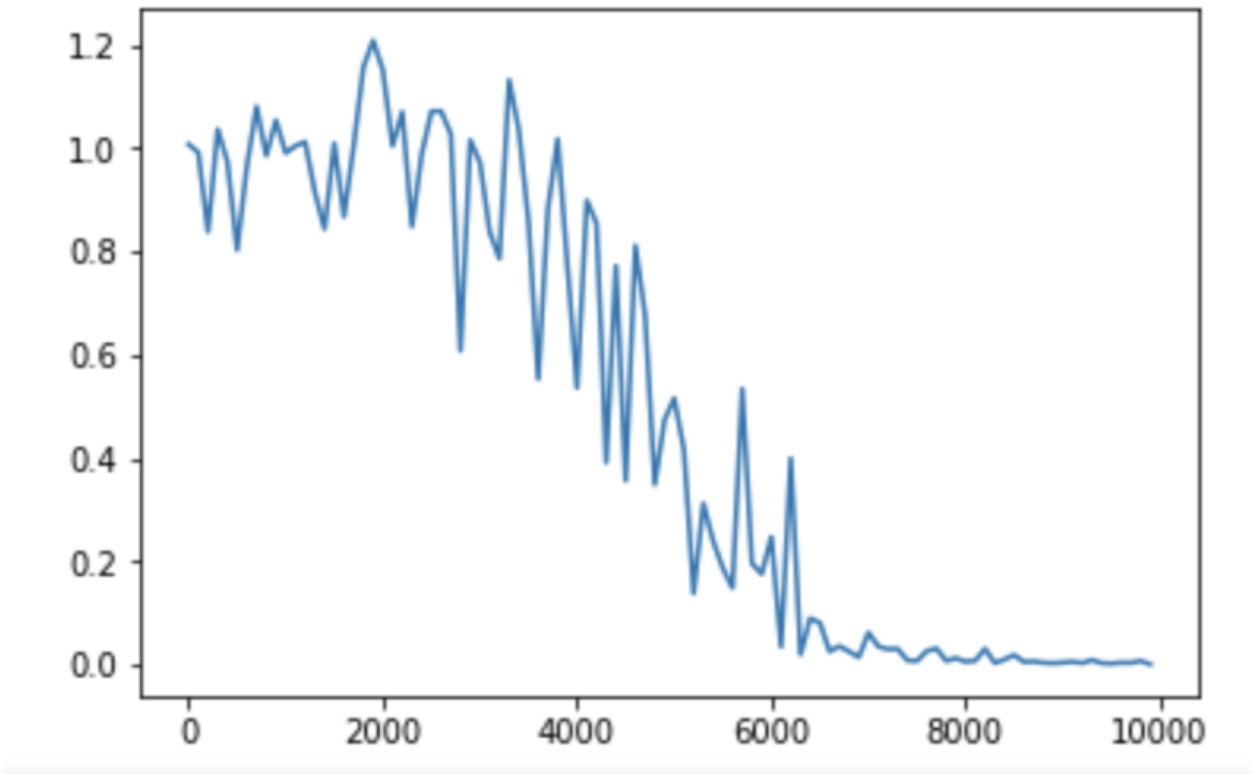
時系列モデルを扱うには、初期の状態と過去の時間t-1の状態を保持し、そこから次の時間でのtを再帰的に求める再帰構造が必要になる。

Jupyter演習

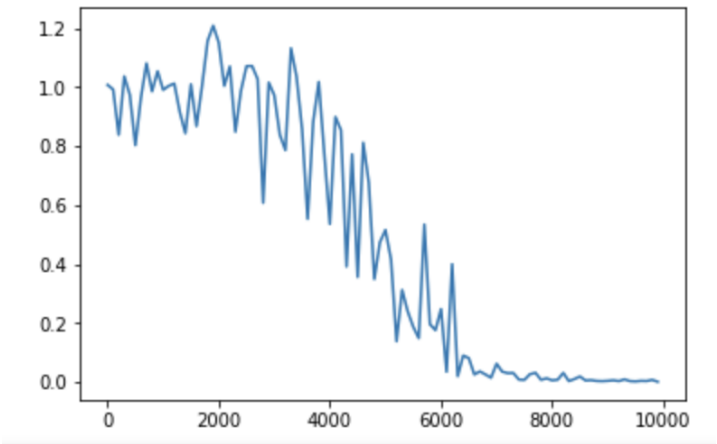
weight_init_std = 1


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
2000
4000
6000
8000
10000

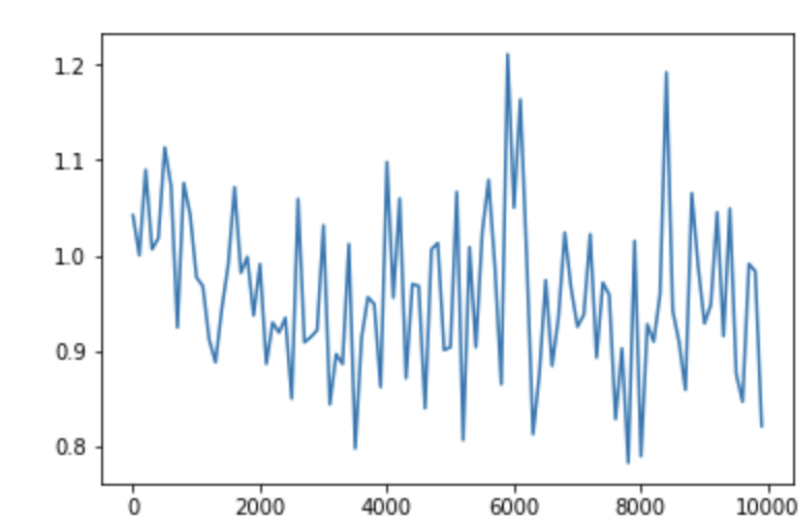
Xavier


1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
2000
4000
6000
8000
10000

He

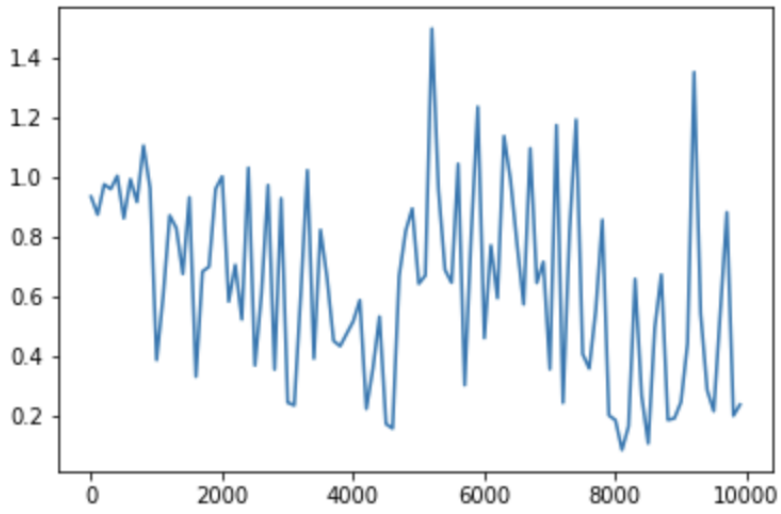

1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
2000
4000
6000
8000
10000

Relu


1.2
1.1
1.0
0.9
0.8
0
2000
4000
6000
8000
10000

tanh

```
def d_tanh(x):
    return 1/(np.cosh(x) ** 2)
```



1-2-1BPTTとは誤差逆伝播法の復習

BPTTとは

RNNにおいてのパラメータ調整方法の一種

→誤差逆伝播の一種

1-2-2BPTTの数学的記述

$$\frac{\partial E}{\partial W_{(in)}} = \frac{\partial E}{\partial u^t}\left[\frac{\partial u^t}{\partial W_{(in)}}\right]^T = \delta^t\left[x^t\right]^T$$

$$\frac{\partial E}{\partial W_{(out)}} = \frac{\partial E}{\partial v^t}\left[\frac{\partial v^t}{\partial W_{(out)}}\right]^T = \delta^{out,t}\left[z^t\right]^T$$

$$\frac{\partial E}{\partial W} = \frac{\partial E}{\partial u^t}\left[\frac{\partial u^t}{\partial W}\right]^T = \delta^t\left[z^{t-1}\right]^T$$

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial u^t}\frac{\partial u^t}{\partial b} = \delta^t$$

$$\frac{\partial E}{\partial c} = \frac{\partial E}{\partial v^t}\frac{\partial v^t}{\partial c} = \delta^{out,t}$$

np.dot(X.T, delta[:,t].reshape(1,-1))

np.dot(z[:,t+1].reshape(-1,1), delta_out[:,t].reshape(-1,1))

np.dot(z[:,t].reshape(-1,1), delta[:,t].reshape(1,-1))

$$u^t = W_{(in)}x^t + W\ z^{t-1} + b$$

$$z^t = f\left(W_{(in)}x^t + W\ z^{t-1} + b\right)$$

$$v^t = W_{(out)}z^t + c$$

$$y^t = g\left(W_{(out)}z^t + c\right)$$

1-2-3BPTTの全体像

$$W_{(in)}^{t+1} = W_{(in)}^{t} - \epsilon \frac{\partial E}{\partial W_{(in)}} = W_{(in)}^{t} - \epsilon \sum_{z=0}^{T_t} \delta^{t-z} \left[x^{t-z} \right]^T$$

$$W_{(out)}^{t+1} = W_{(out)}^{t} - \epsilon \frac{\partial E}{\partial W_{(out)}} = W_{(in)}^{t} - \epsilon \delta^{out,t} \left[z^{t} \right]^T$$

$$W^{t+1} = W^{t} - \epsilon \frac{\partial E}{\partial W} = W_{(in)}^{t} - \epsilon \sum_{z=0}^{T_t} \delta^{t-z} \left[z^{t-z-1} \right]^T$$

$$b^{t+1} = b^{t} - \epsilon \frac{\partial E}{\partial b} = b^{t} - \epsilon \sum_{z=0}^{T_t} \delta^{t-z}$$

$$c^{t+1} = c^{t} - \epsilon \frac{\partial E}{\partial c} = c^{t} - \epsilon \delta^{out,t}$$

Section2) LSTM全体像（前回の流れと課題全体像のビジョン）

RNNの課題

時系列を遡れば遡るほど、勾配が消失していく。

→長い時系列の学習が困難

解決策

前回の授業で触れた勾配消失の解決方法とは、別で、構造自体を変えて解決したものがLSTM。

確認テスト

シグモイド関数を微分した時、入力値が0の時に最大値 をとる。その値として正しいものを選択肢から選べ。

→ 0.25

勾配爆発

勾配爆発とは？勾配が、層を逆伝播するごとに指数関数的に大きくなっていく。

演習チャレンジ

RNNや深いモデルでは勾配の消失または爆発が起こる傾向がある。勾配爆発を防ぐために勾配のクリッピングを行うという 手法がある。具体的には勾配のノルムがしきい値を超えたら、勾配のノルムをしきい値に正規化するというものである。以下は勾配のクリッピングを行う関数である。

(さ)にあてはまるのはどれか。

→ (1) gradient * rate

勾配のノルムがしきい値より大きいときは、勾配のノルムをしきい値に正規化するので、クリッピングした勾配は、勾配×(しきい値/勾配のノルム)と計算される。つまり、gradient * rate　である。

つまり、delta_t = delta_t.dot(U)となる。

2-1CEC

勾配消失および勾配爆発の解決方法として、勾配が、1であれば解決できる。

CECの課題

入力データについて、時間依存度に関係なく重みが一律である。

ニューラルネットワークの学習特性が無いということ。

入力層→隠れ層への重み入力重み衝突。

隠れ層→出力層への重み出力重み衝突。

2-2入力ゲートと出力ゲート

入力・出力ゲートの役割とは

入力・出力ゲートを追加することで、それぞれのゲートへの入力値の重みを、重み行列W,Uで可変可能とする。

→CECの課題を解決。

2-3忘却ゲート

忘却ゲート誕生の経緯

LSTMブロックの課題

LSTMの現状　CECは、過去の情報が全て保管されている。

課題　過去の情報が要らなくなった場合、削除することはできず、保管され続ける。

解決策　過去の情報が要らなくなった場合、そのタイミングで情報を忘却する機能が必要。

→忘却ゲートの誕生

覗き穴結合

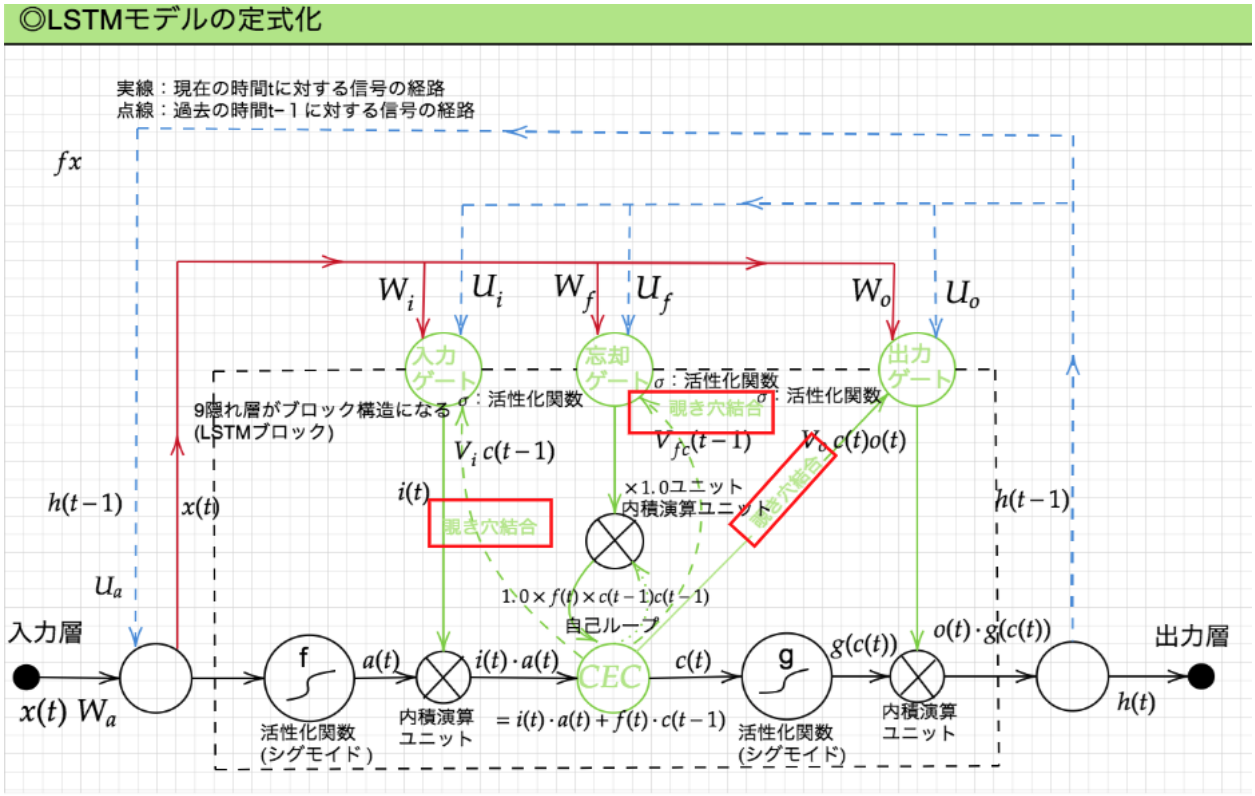
課題

CECの保存されている過去の情報を、任意のタイミングで他のノードに伝播させたり、あるいは任意のタイミングで忘却させたい。CEC自身の値は、ゲート制御に影響を与えていない。

覗き穴結合とは

→CEC自身の値に、重み行列を介して伝播可能にした構造。

2-4覗き穴結合



Section3) GRU

課題　LSTMでは、パラメータ数が多く、計算負荷が高くなる問題があった。

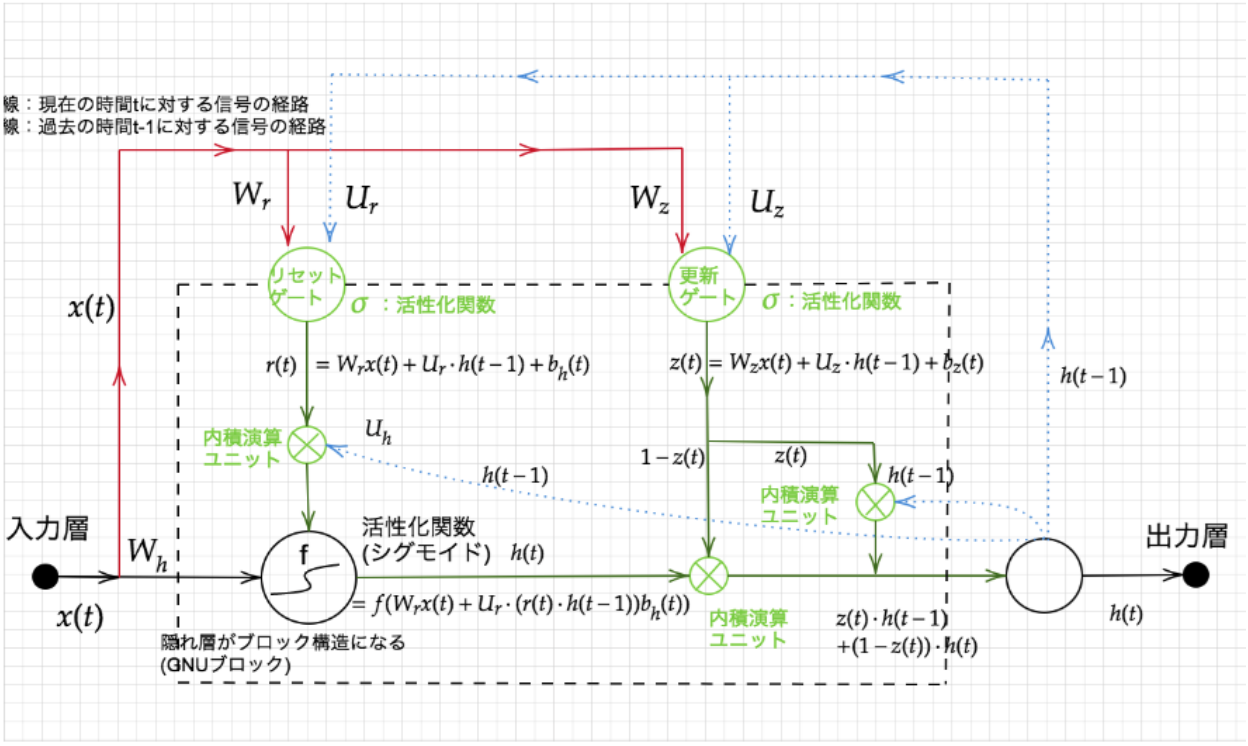
→GRU誕生

GRUとは？

従来のLSTMでは、パラメータが多数存在していたため、計算負荷が大きかった。しかし、GRUでは、そのパラメータを大幅に削減し、精度は同等またはそれ以上が望める様になった構造。

メリット

計算負荷が低い。



確認テスト

LSTMとGRUの違いを簡潔に述べよ

→ パラメータの数がLSTMの方が多い。そのためGRUの方が実行速度の面で有利 どっちがどのタスクで有利かはケースバイケース

Section4)双方向RNN

双方向RNN

過去の情報だけでなく、未来の情報を加味することで、精度を向上させるためのモデル実用例文章の推敲や、機械翻訳等

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, ーステッ 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ハ゜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)にあてはまるのはと゛れか

→ 4 np.concatenate([h_f, h_b[::-1]], axis=1)

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。

※ np.concatenateは配列の結合。axisに指定した軸方向に新しい要素を追加するため、axisによっては配列の形状が噛み合わないとエラーが発生します。(1次元配列ではaxis=1以上はエラー)

■RNNでの自然言語処理

Section5)Seq2Seq全体像

Seq2seqとは

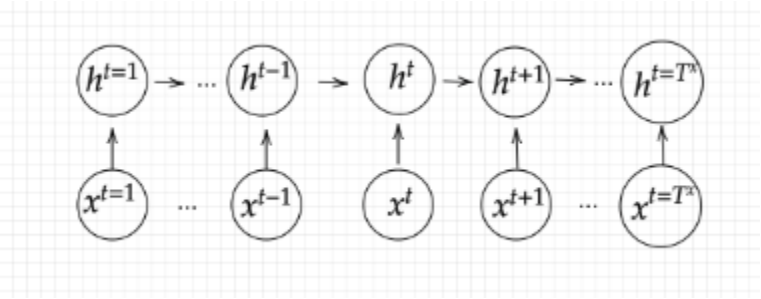
Encoder-Decoderモデルの一種を指します。

Seq2seqの具体的な用途とは

機械対話や、機械翻訳などに使用されている。

5-1Encoder RNN

Encoder RNNユーザーがインプットしたテキストデータを、単語等のトークンに区切って渡す構造。



Taking :文章を単語等のトークン毎に分割し、トークンごとのIDに分割する。

Embedding :IDから、そのトークンを表す分散表現ベクトルに変換。

Encoder RNN:ベクトルを順番にRNNに入力していく。

5-2Decoder RNN

Decoder RNN処理手順

2.Sampling:生成確率にもとづいてtoken をランダムに選びます。

3.Embedding:2で選ばれたtoken をEmbedding してDecoder RNN への次の入力とします。
4.Detokenize:1 -3 を繰り返し、2で得られたtoken を文字列に直します。

確認テスト

下記の選択肢から、seq2seqについて説明しているものを選べ。

(1)時刻に関して順方向と逆方向のRNNを構成し、それら2つの中間層表現を特徴量として利用 するものである。

(2)RNNを用いたEncoder-Decoderモデルの一種であり、機械翻訳などのモデルに 使われる。

(3)構文木などの木構造に対して、隣接単語から表現ベクトル(フレーズ)を作るという演算を再帰的に行い(重みは共通)、文全体の表現ベクトルを得るニューラルネットワークである。

(4)RNNの一種であり、単純なRNNにおいて問題となる勾配消失問題をCECとゲートの概念を導入することで解決したものである。

→ 2

演習チャレンジ

機械翻訳タスクにおいて、入力は複数の単語から成る文(文章)であり、それぞれの単語はone-hotベクトルで表 現されている。Encoderにおいて、それらの単語は単語埋め込みによ り特徴量に変換され、そこからRNNによって (一般にはLSTMを使うことが多い)時系列の情報をもつ特徴へとエンコードされる。以下は、入力である文(文 章)を時系列の情報をもつ特徴量へとエンコードする関数である。ただし_activation関数はなんらかの活性化関 数を表すとする。

(き)にあてはまるのはどれか。

→ 1

単語wはone-hotベクトルであり、それを単語埋め込みにより別の特徴量に変換する。これは埋め込み行列E を用いて、E.dot(w)と書ける。

5-3HRED

Seq2seqの課題

一問一答しかできない

問に対して文脈も何もなく、ただ応答が行われる続ける。

→HRED誕生

HREDとは？

過去n-1 個の発話から次の発話を生成する。

→Seq2seqでは、会話の文脈無視で、応答がなされたが、HREDでは、前の単語の流れに即して応答されるため、より人間らしい文章が生成される。

システム:インコかわいいよね。

ユーザー:うん

システム:インコかわいいのわかる。

HREDの構造

HREDとは？Seq2Seq+ Context　RNN

Context RNN: Encoder のまとめた各文章の系列をまとめて、これまでの会話コンテキスト全体を表すベクトルに変換する構造。

→過去の発話の履歴を加味した返答をできる。

HREDの課題

HRED は確率的な多様性が字面にしかなく、会話の「流れ」のような多様性が無い。

→同じコンテキスト（発話リスト）を与えられても、答えの内容が毎回会話の流れとしては同じものしか出せない。

HRED は短く情報量に乏しい答えをしがちである。

→短いよくある答えを学ぶ傾向がある。ex)「うん」「そうだね」「・・・」など。

5-4VHRED

VHREDとは？HREDに、VAEの潜在変数の概念を追加したもの。

→HREDの課題を、VAEの潜在変数の概念を追加することで解決した構造。

確認テスト

seq2seqとHRED、HREDとVHREDの違いを簡潔に述べよ。

Seq2seqとHREDの違い → 1問1答しか出来ない構造を改修したのがHRED HREDとVHREDの違い → 同じ文脈だと毎回同じ答えしか出ないHREDを改修したのがVHRED

5-5VAE

5-5-1オートエンコーダー

オートエンコーダとは

教師なし学習の一つ。そのため学習時の入力データは訓練データのみで教師データは利用しない。

オートエンコーダ具体例

MNISTの場合、28x28の数字の画像を入れて、同じ画像を出力するニューラルネットワークということになります。

オートエンコーダ構造

オートエンコーダ構造説明

入力データから潜在変数zに変換するニューラルネットワークをEncoder逆に潜在変数zをインプットとして元画像を復元するニューラルネットワークをDecoder。

メリット

次元削減が行えること。

5-5-2VAE

通常のオートエンコーダーの場合、何かしら潜在変数zにデータを押し込めているものの、その構造がどのような状態かわからない。

→VAEはこの潜在変数zに確率分布z~N(0,1)を仮定したもの。

→VAEは、データを潜在変数zの確率分布という構造に押し込めることを可能にします。

確認テスト

VAEに関する下記の説明文中の空欄に当てはまる言葉を答えよ。

自己符号化器の潜在変数に____を導入したもの。

→ 確率分布

Section6)Word2vec

word2vec課題:RNNでは、単語のような可変長の文字列をNNに与えることはできない。

→固定長形式で単語を表す必要がある。

学習データからボキャブラリを作成

I want to eat apples. I like apples.

→{apples,eat,I,like,to,want}

word2vecメリット

大規模データの分散表現の学習が、現実的な計算速度とメモリ量で実現可能にした。

✗:ボキャブラリ×ボキャブラリだけの重み行列が誕生。

○:ボキャブラリ×任意の単語ベクトル次元で重み行列が誕生。

ボキャブラリ数×単語ベクトルの次元数の重み行列

Section7)AttentionMechanism

課題

seq2seq の問題は長い文章への対応が難しいです。seq2seq では、2単語でも、100単語でも、固定次元ベクトルの中に入力しなければならない。

「入力と出力のどの単語が関連しているのか」の関連度を学習する仕組み。

解決策:文章が長くなるほどそのシーケンスの内部表現の次元も大きくなっていく、仕組みが必要になります。Attention Mechanism

確認テスト

RNNとword2vec、seq2seqとseq2seq+Attention Mechanismの違いを簡潔に述べよ。

word2vecはwordに対して重みをつけて、現実的な処理速度にしている。RNNはボキャブラリ×ボキャブラリでデータを処理する seq2seqに比べてAttention Mechanismは長い文章を入れたときにでも翻訳として成り立ちやすい。入出力間の重要度が学習に対して効いてくる

演習チャレンジ

以下は再帰的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。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。

(く)にあてはまるのはどれか

→ 2 W.dot(np.concarenate([left, right))

Day4

Section1) 強化学習

1-1強化学習とは

長期的に報酬を最大化できるように環境のなかで行動を選択できるエージェントを作ることを目標とする機械学習の一分野行動の結果として与えられる利益(報酬)をもとに、行動を決定する原理を改善していく仕組み

1-2強化学習の応用例

マーケティングの場合

環境:会社の販売促進部

エージェント:プロフィールと購入履歴に基づいて、キャンペーンメールを送る顧客を決めるソフトウェアである。

行動:顧客ごとに送信、非送信のふたつの行動を選ぶことになる。報酬:キャンペーンのコストという負の報酬とキャンペーンで生み出されると推測される売上という正の報酬を受ける

1-3探索と利用のトレードオフ

環境について事前に完璧な知識があれば、最適な行動を予測し決定することは可能。

→どのような顧客にキャンペーンメールを送信すると、どのような行動を行うのかが既知である状況。

→強化学習の場合、上記仮定は成り立たないとする。不完全な知識を元に行動しながら、データを収集。最適な行動を見つけていく。

過去のデータで、ベストとされる行動のみを常に取り続ければ他にもっとベストな行動を見つけることはできない。

探索が足りない状態

未知の行動のみを常に取り続ければ、過去の経験が活かせない。

→利用が足りない状態

1-4強化学習のイメージ

環境　状態S

↓報酬　観測　行動

エージェント

方策π　　行動価値関数:Q(s,a)

価値V　　方策関数: π(s,a)

1-5強化学習の差分

強化学習と通常の教師あり、教師なし学習との違い結論:目標が違う・教師なし、あり学習では、データに含まれるパターンを見つけ出すおよびそのデータから予測することが目標・強化学習では、優れた方策を見つけることが目標

1-6行動価値関数

価値関数とは・価値を表す関数としては、状態価値関数と行動価値関数の2種類があるある状態の価値に注目する場合は、状態価値関数状態と価値を組み合わせた価値に注目する場合は、行動価値関数

1-7方策関数

方策関数とは方策ベースの強化学習手法において、ある状態でどのような行動を採るのかの確率を与える関数のことです。

1-8方策勾配法

定義方法・平均報酬・割引報酬和上記の定義に対応して、行動価値関数:Q(s,a)の定義を行い。方策勾配定理が成り立つ。

Section2) Alpha Go

AlphaGoの学習は以下のステップで行われる

1、教師あり学習によるRollOutPolicyとPolicyNetの学習

PolicyNetの教師あり学習KGS Go Server（ネット囲碁対局サイト）の棋譜データから3000万局面分の教師を用意し、教師と同じ着手を予測できるよう学習を行った。具体的には、教師が着手した手を1とし残りを0とした19×19次元の配列を教師とし、それを分類問題として学習した。この学習で作成したPolicyNetは57%ほどの精度である。

2、強化学習によるPolicyNetの学習

現状のPolicyNetとPolicyPoolからランダムに選択されたPolicyNetと対局シミュレーションを行い、その結果を用いて方策勾配法で学習を行った。PolicyPoolとは、PolicyNetの強化学習の過程を500Iteraionごとに記録し保存しておいたものである。現状のPolicyNet同士の対局ではなく、PolicyPoolに保存されているものとの対局を使用する理由は、対局に幅を持たせて過学習を防ごうというのが主である。この学習をminibatch size 128で1万回行った。

3、強化学習によるValueNetの学習

PolicyNetを使用して対局シミュレーションを行い、その結果の勝敗を教師として学習した。教師データ作成の手順は1、まずSL PolicyNet(教師あり学習で作成したPolicyNet)でN手まで打つ。2、N+1手目の手をランダムに選択し、その手で進めた局面をS（N+1）とする。3、S（N+1）からRLPolicyNet（強化学習で作成したPolicyNet）で終局まで打ち、その勝敗報酬をRとする。S(N+1）とRを教師データ対とし、損失関数を平均二乗誤差とし、回帰問題として学習した。この学習をminibatch size 32で5000万回行ったN手までとN+1手からのPolicyNetを別々にしてある理由は、過学習を防ぐためであると論文では説明されている

AlphaGo(Lee) とAlphaGoZeroの違い

1、教師あり学習を一切行わず、強化学習のみで作成

2、特徴入力からヒューリスティックな要素を排除し、石の配置のみにした

3、PolicyNetとValueNetを1つのネットワークに統合した

4、Residual Net（後述）を導入した

5、モンテカルロ木探索からRollOutシミュレーションをなくした

Alpha Go Zeroの学習法

Alpha Goの学習は自己対局による教師データの作成、学習、ネットワークの更新の3ステップで構成される自己対局による教師データの作成現状のネットワークでモンテカルロ木探索を用いて自己対局を行う。まず30手までランダムで打ち、そこから探索を行い勝敗を決定する。自己対局中の各局面での着手選択確率分布と勝敗を記録する。教師データの形は（局面、着手選択確率分布、勝敗）が1セットとなる。学習自己対局で作成した教師データを使い学習を行う。NetworkのPolicy部分の教師に着手選択確率分布を用い、Value部分の教師に勝敗を用いる。損失関数はPolicy部分はCrossEntropy、Value部分は平均二乗誤差。ネットワークの更新学習後、現状のネットワークと学習後のネットワークとで対局テストを行い、学習後のネットワークの勝率が高かった場合、学習後のネットワークを現状のネットワークとする。

Section3) 軽量化・高速化技術

3-1モデル並列

モデル並列化•親モデルを各ワーカーに分割し、それぞれのモデルを学習させる。全てのデータで学習が終わった後で、一つのモデルに復元。•モデルが大きい時はモデル並列化を、データが大きい時はデータ並列化をすると良い。

モデルのパラメータ数が多いほど、スピードアップの効率も向上する。

3-2データ並列

親モデルを各ワーカーに分割し、それぞれのモデルを学習させる。全てのデータで学習が終わった後で、一つのモデルに復元。•モデルが大きい時はモデル並列化を、データが大きい時はデータ並列化をすると良い。

3-3GPU

GPU

•比較的低性能なコアが多数

•簡単な並列処理が得意

•ニューラルネットの学習は単純な行列演算が多いので、高速化が可能

モデルの軽量化の利用

モデルの軽量化はモバイル, IoT 機器において有用な手法モバイル端末やIoT はパソコンに比べ性能が大きく劣る主に計算速度と搭載されているメモリ

モデルの軽量化は計算の高速化と省メモリ化を行うためモバイル,IoT 機器と相性が良い手法になる

3-4量子化

量子化(Quantization)とは

ネットワークが大きくなると大量のパラメータが必要なり学習や推論に多くのメモリと演算処理が必要

→通常のパラメータの64 bit 浮動小数点を32 bit など下位の精度に落とすことでメモリと演算処理の削減を行う

量子化の利点と欠点利点

計算の高速化省メモリ化欠点精度の低下

極端な量子化

極端な量子化を考える。表現できる値が0,1 の1 bitの場合a = 0.1 が真値の時、関数y(x) = axを近似する場合を考える際、学習によってa が0.1 を得る必要があるしかし、量子化によってa が表現できる値が0,1 のため求められる式はy(x) = 0, y(x) =xのようになり、誤差の大きな式になってしまう。量子化する際は極端に精度が落ちない程度に量子化をしなければならない

3-5蒸留

蒸留とは

精度の高いモデル精度-高い計算コスト-高い

→軽量なモデル精度-普通計算コスト-低い

蒸留の利点

下記のグラフはCifar 10 データセットで学習を行ったレイヤー数と精度のグラフになる表のback propagation は通常の学習、Knowledge Distillation は先に説明した蒸留手法、Hint Taraing は蒸留は引用論文で提案された蒸留手法図から蒸留によって少ない学習回数でより精度の良いモデルを作成することができている

3-6プルーニング

ネットワークが大きくなると大量のパラメータなるがすべてのニューロンの計算が制度に寄与しているわけではない

モデルの精度に寄与が少ないニューロン削減することでモデルの軽量化、高速化が見込める。

モデルの軽量化まとめ

量子化：重みの精度を下げることにより計算の高速化と省メモリ化を行う技術

蒸留：複雑で精度の良い教師モデルから軽量な生徒モデルを効率よく学習を行う技術

プルーニング：寄与の少ないニューロンをモデルから削減し高速化と省メモリ化を行う技術

Section4) 応用技術

4-1MobileNet

一般的な畳み込みレイヤーは計算量が多いMobileNetsはDepthwise ConvolutionとPointwise Convolutionの組み合わせで軽量化を実現

Depthwise Convolution

●仕組み

○入力マップのチャネルごとに畳み込みを実施

○出力マップをそれらと結合(入力マップのチャネル数と同じになる)

Pointwise Convolution

●仕組み

○1 x 1 convとも呼ばれる(正確には1 x 1 x c)

○入力マップのポイントごとに畳み込みを実施

○出力マップ(チャネル数)はフィルタ数分だけ作成可能(任意のサイズが指定可能)

4-2　DenseNet

●DenseNetとResNetの違い

○DenseBlockでは前方の各層からの出力全てが後方の層への入力として用いられる
○RessidualBlockでは前1層の入力のみ後方の層へ入力

●DenseNet内で使用されるDenseBlockと呼ばれるモジュールでは成長率(Growth Rate)と呼ばれるハイパーパラメータが存在する。

○DenseBlock内の各ブロック毎にk個ずつ特徴マップのチャネル数が増加していく時、kを成長率と呼ぶ

4-3　Layer正規化/Instance正規化

●Layer Norm

○N個のsampleのうち一つに注目。H x W x Cの全てのpixelが正規化の単位。

○RGBの3チャネルのsampleがN個の場合は、あるsampleを取り出し、全てのチャネルの平均と分散を求め正規化を実施(図の青い部分に対応)。特徴マップごとに正規化された特徴マップを出力

○ミニバッチの数に依存しないので、上記の問題を解消できていると考えられる

Layer Normは、入力データや重み行列に対して、以下の操作を施しても、出力が変わらないことが知られている。

○入力データのスケールに関してロバスト

○重み行列のスケールやシフトに関してロバスト

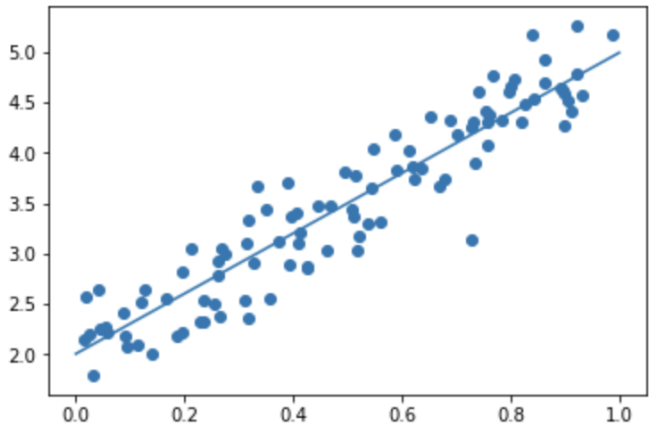
4-4Wavenet

深層学習を用いて結合確率を学習する際に、効率的に学習が行えるアーキテクチャ

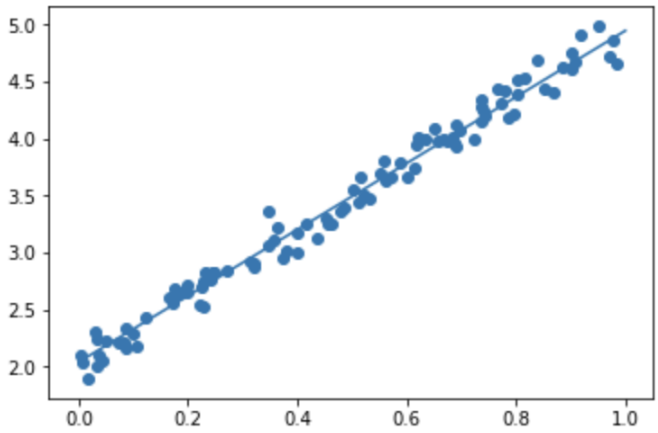
Section5)例題解説

線形回帰

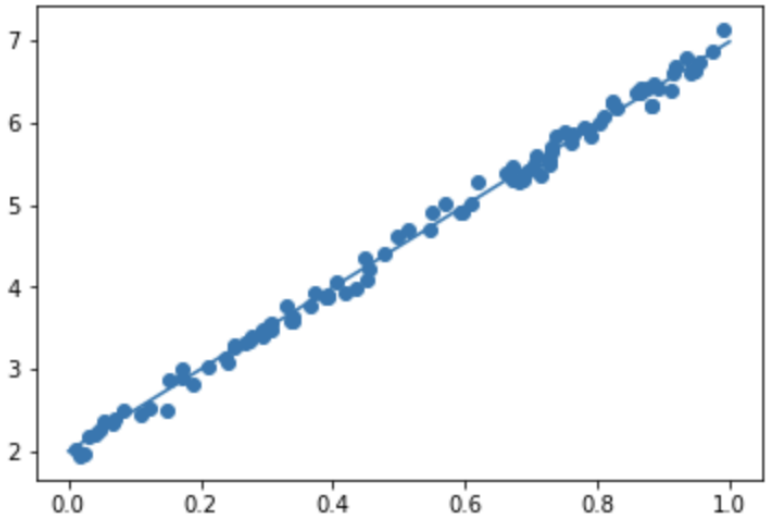
noise=0.3



noise=0.1

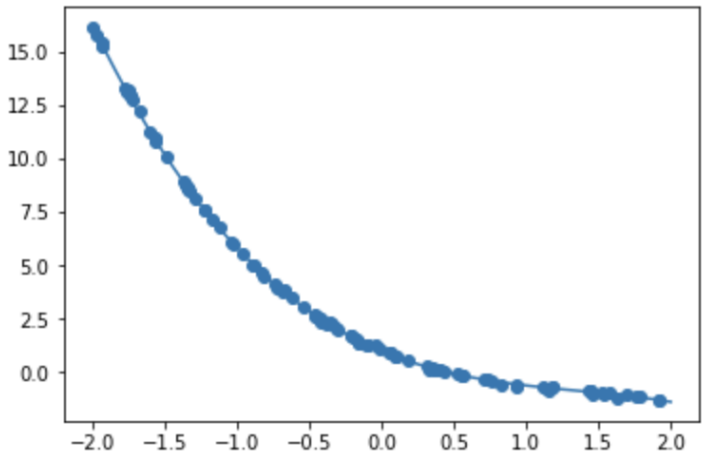


d=5x+2, noise=0.1

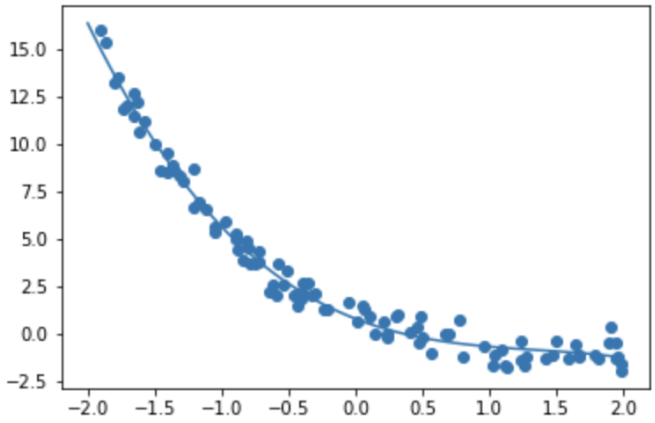


非線形回帰

noise = 0.05

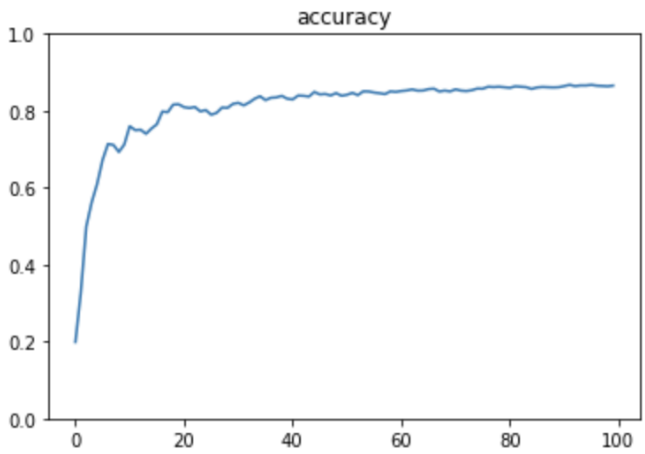


noise = 0.6

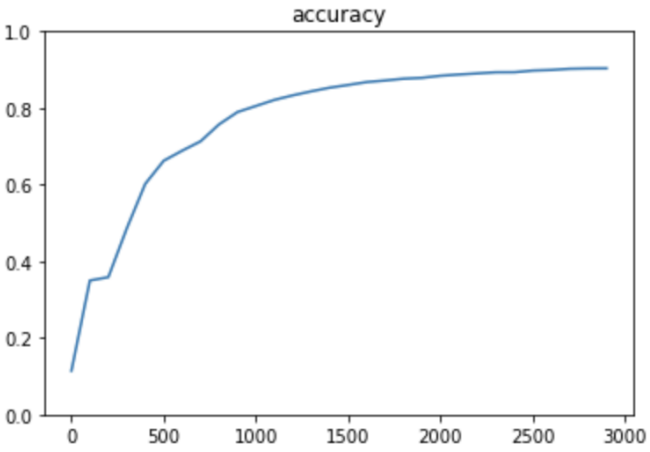


TensorFlowによるMNIST

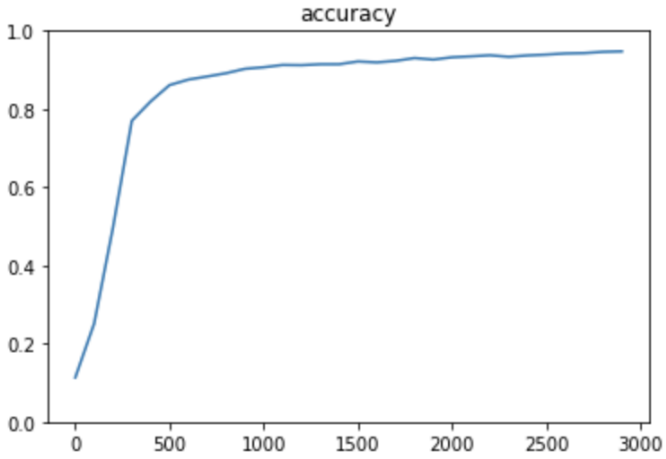
1層



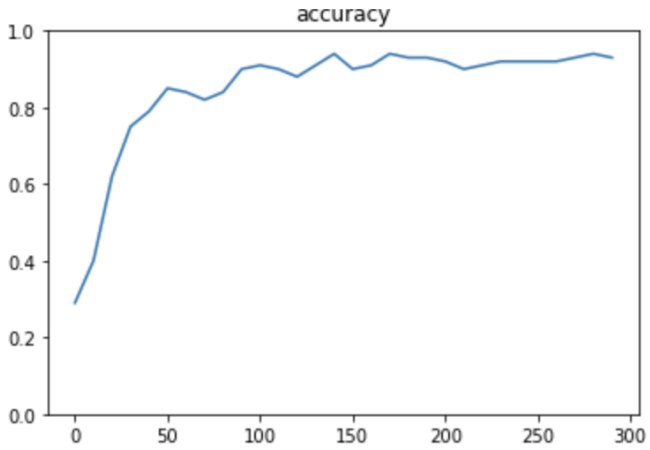
3層(Adam)



3層(Momentum)



CNN(dropout_rate=0)


accuracy
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
300

確認テスト

1 Inception module

GoogleNetの特徴としてInception moduleと呼ばれる複数のフィルタ郡により構成されたブロックが挙げられる。Inception moduleに関する記述として誤っているものはどれか

→ (a)が間違い。スパースなデータが増えているため

2 Auxiliary Less

GoogleNetnのの特徴としてAuxiliary Lessの存在が挙げられる。Auxiliary Lossに関する記述といて間違っているものを選べ

→ (a)が間違い。Classifiersを増やすことによって計算量は増える

3 ResNet

深くモデルを設計出来なかった問題は？

→ (a) 勾配消失問題

4 ResNet

層をまたがる結合の名前

→ (a)。Identity mapping、いわゆるスキップコネクションのこと

5 Residual Brock

Residual Brockの導入によって期待されたもの

→ (a)。ブロックへの入力にこれ以上変換が必要ない場合は重みが0となり、小さな変換が求められる場合は対応する小さな変換をより見つけやすくなる

6 転移学習

転移学習の記載として誤ったものはどれか？

→ (あ) 。ワンショット学習というののはあるクラスにたいして画像を1枚用意して、特徴ベクトルを抽出する。特徴ベクトルが同じクラスであれば近い、離れていれば遠いということで判断する。転移学習の説明としては不当

7 一般物体検出アルゴリズム

一般物体検出アルゴリズムとして正しいものを選べ

→ (a) (b)は最大2つというのが間違い。必ず定義された2つの候補領域が生成される (c) 2つの誤差の単純和が間違い。2つの誤差の重み付けまで表現されている (d)出力についてはカテゴリのプラスの確率と2つの候補領域が出力される

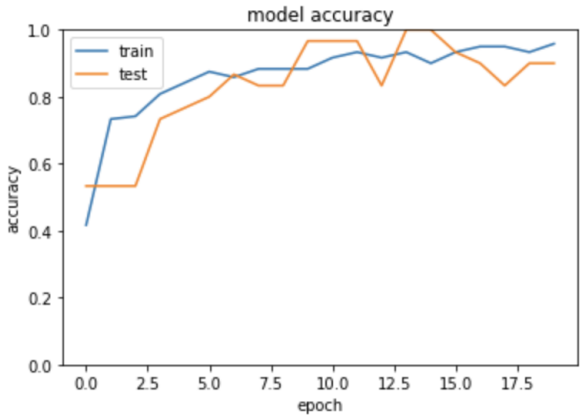
確認テスト

GG, GoogleNet, ResNetの特徴をそれぞれ簡潔に述べよ

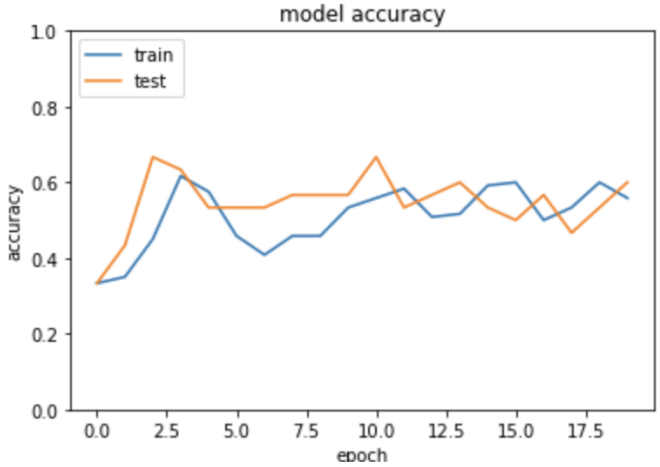
→ VGGについては2014からあるモデル 、単純なモデルの積み重ねになっているのでシンプルなモデル。ただしパラメータ数は多い → GoogleNetはinception moduleを使っている。次元削減やスパースなもので表現されるのが特徴 → ResNetはスキップコネクション：identity moduleを使って深い学習が出来る

irisデータセットを実装

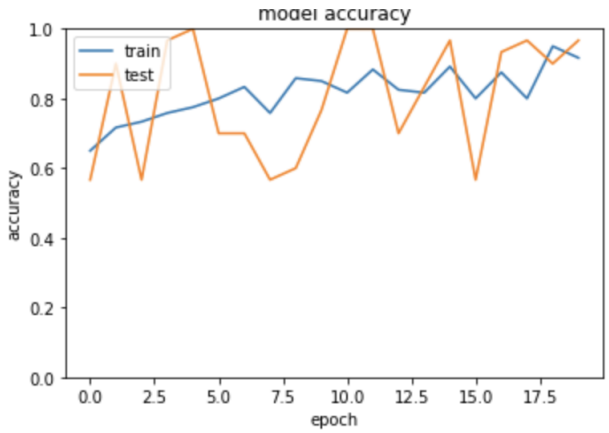
relu



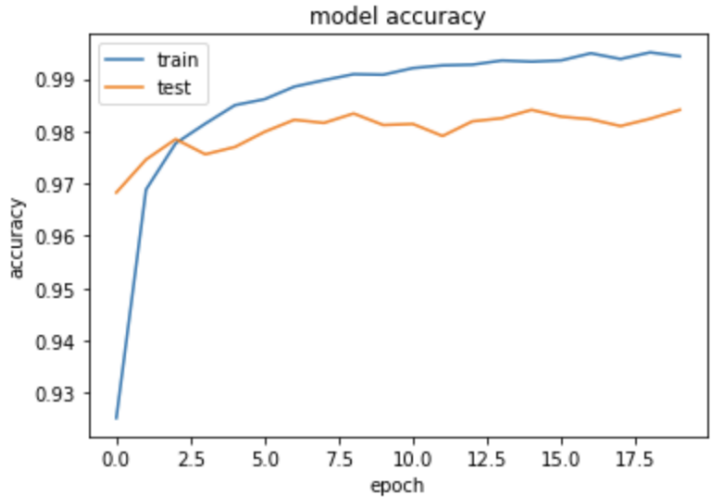
sigmoid



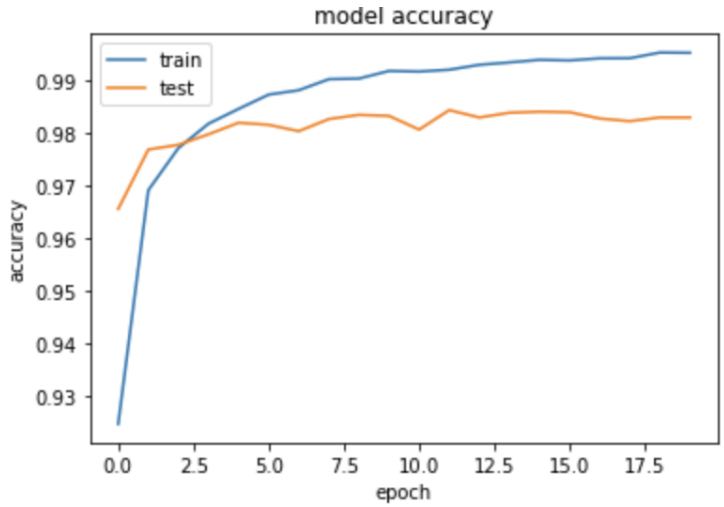
SGD(lr=0.1)



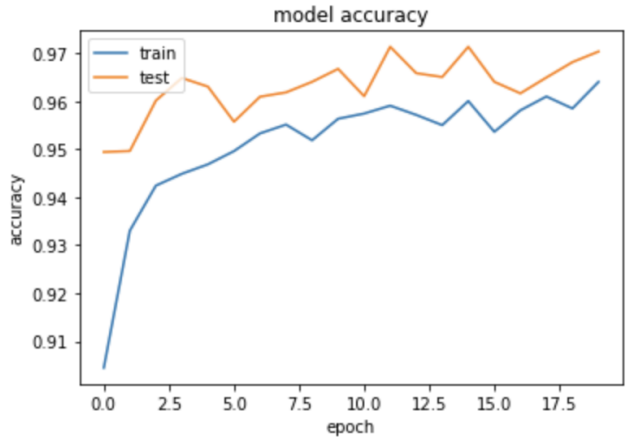
MNIST(NN)



sparse_categorical_crossentropy


model accuracy
train
test
accuracy
epoch
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

Adam(lr=0.01)


model accuracy
train
test
accuracy
epoch
0.97
0.96
0.95
0.94
0.93
0.92
0.91
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5