day3

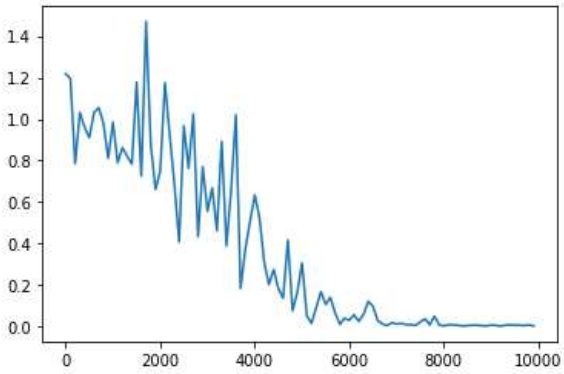
再帰型ニューラルネットワークの概念

再帰型ニューラルネットワーク（RNN）とは、音声データ、テキストデータ、株価などの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。各時点tごとにy_tが出力されるが、各時点の活性関数の入力には、1時点前の中間層の出力z_{t-1}が線形結合される。これにより全ての時点間の出力がつながっており、データの時系列的なつながりが考慮されている。

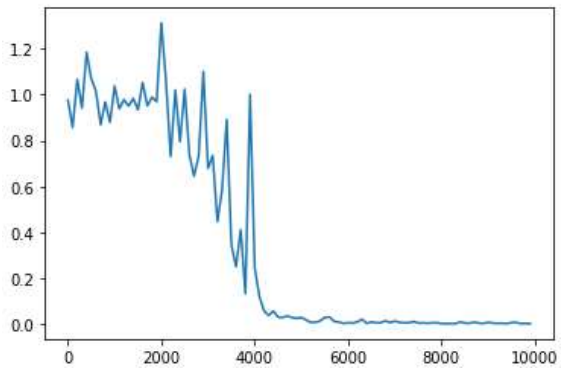
■実装演習

2値をバイナリ加算した結果を予測するRNN。バイナリ加算結果の各桁が取る値は、より下位の桁の影響を受ける。このため各桁毎に出力値を予測するRNNによって、バイナリ加算の結果が予測可能（算出ではない）となる。

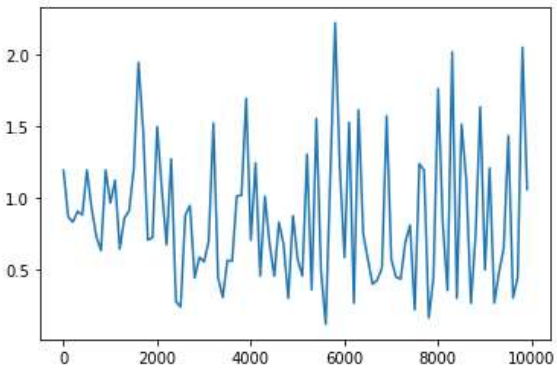
1)活性化関数：シグモイド関数、重み：標準正規分布よりランダムサンプリング



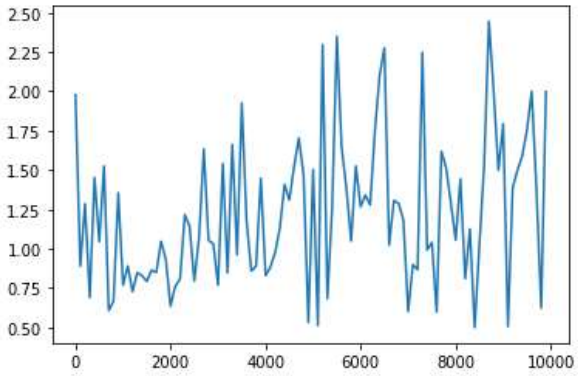
2)活性化関数：シグモイド関数、重み：Xavier



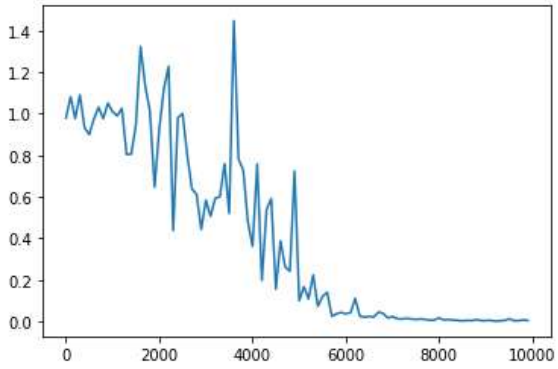
3)活性化関数：ReLU関数、重み：He



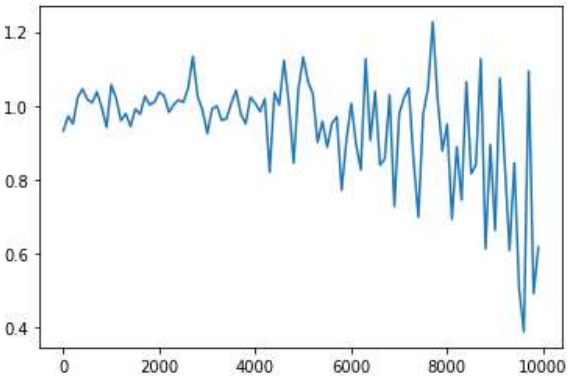
3)活性化関数：tanh関数、重み：He



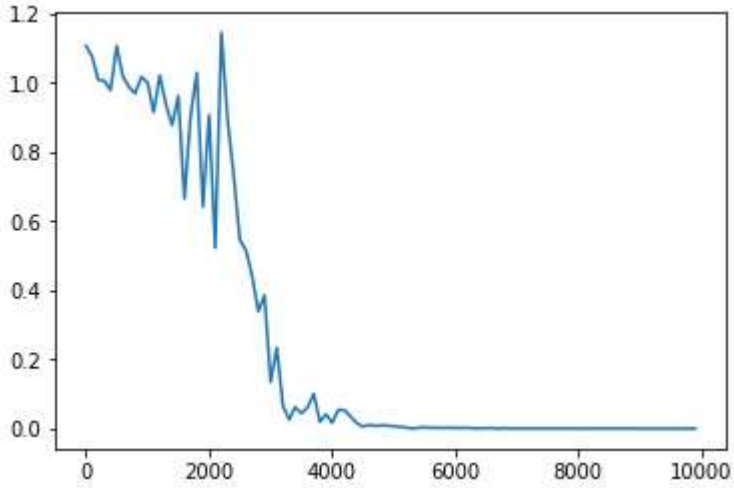
5)活性化関数：シグモイド関数、重み：Xavier、中間層ノード32（ 1)の2倍）



6)活性化関数：シグモイド関数、重み：Xavier、学習率0.05 （ 1)の1/2倍）



7)活性化関数：シグモイド関数、重み：Xavier、学習率0.2 （ 1)の2倍）



■考察
上記の1)の例では約6,000回の学習により誤差関数がほぼゼロとなり、予測の正解率が100%を概ね達成。これにXavier初期化手法を合わせると約4,000回で、さらに学習率を2倍にすると約3,000回で誤差関数が0に収束した。また活性関数として勾配消失が起きにくいReLU関数やtanh関数を用いると、勾配発散が観測され、学習は進まなかった。モデルは妥当でも、学習の成否がハイパーパラメータや活性化関数の選択次第で決まることがよく分かった。
■他記事レポート
〈機械学習基礎〉数式なし！ LSTM・GRU超入門　AGIRobots
要旨
RNNが長期にわたる特徴の学習は上手くできない原因は2つ
①勾配消失
　層が深くなると勾配消失が起こる。RNNでは後述の重み衝突問題も同時に生じるため、活性化関数選択や事前学習といった方法とは別のアプローチで勾配消失に対処する。
②重み衝突
　入力重み衝突「現時刻に入力された信号の有用性に従って重みを更新するが、今は有用でなくても将来有用な情報だったら重みをどう調整すべきか」
　出力重み衝突「現時刻に出力された信号の有用性に従って重みを更新するが、今は有用でなくても将来有用な情報だったら重みをどう調整すべきか」
　つまり重み衝突とはRNNにおいて入出力信号が長期的な特徴を持つか短期的特徴をもつかわからないので重みを大きくすべきか小さくすべきかが判断できない状態

LSTM

RNNには、時系列を遡れば遡るほど勾配が消失するため、長い時系列の学習が困難となる課題があった。LSTMはRNN構造の中に、下記の部品が組み込まれている。
1)CEC…過去の入力情報を含め、情報をため込んでおくための部品
2)入力ゲート…今回の入力値と前回の出力値を元に、今回の入力をどのようにCECにため込むかを学習する。
3)出力ゲート…今回の入力値と前回の出力値を元に、CECにためられている情報をどれくらい使うか、を学習する。
4)忘却ゲート…CECには過去の記憶が貯め続けられる。不要になった過去の情報を削除するための部品。今回の入力値と前回の入力値をもとに、CECからの削除の仕方を学習する。
■他記事レポート
再帰型ニューラルネットワークの「基礎の基礎」を理解する　～ディープラーニング入門｜第3回
要旨
　LSTM（Long Short Term Memory）は神経科学の短期記憶、長期記憶からヒントを得てデザインされたもの。RNNとの違いで顕著なのはメモリセルであり、これは時間変化と共に保持すべき情報と忘れるべき情報とを含んでいる。この情報の変化を制御するのが忘却ゲートという関数である。これはメモリセルに対し、各時刻で重大な影響のある入力があればその記憶を強化したり、逆に重要でない情報を消去したりする働きを持つ。

メモリセルと忘却ゲートがLSTMのコアとなるアイデアであるが、LSTMでは中間層の入出力に対してもそれぞれ入力ゲート、出力ゲートが設けられている（構成方法は忘却ゲートと同様）。各時刻でどれくらい新しいデータを反映するか、または出力するかを制御する。

GRU

LSTMの課題として、パラメター数が多いがための計算コストの高さがあった。そこでGRUではパラメータを大幅に削除することで計算コストを削減しつつ、精度をLSTMと同等以上に保つことに成功した。LSTMの各部品（CEC、入力ゲート、出力ゲート、忘却ゲート）の代わりに、リセットゲートと更新ゲートを持つ。

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。
過去の情報を加味する順伝播と未来の情報を加味する逆伝播とがある。
・順伝播…今回の入力と1時点前の出力との和を活性化関数に通すことで出力が生まれる。
・逆伝播…今回の入力と1時点後の出力との和を活性化関数に通すことで出力が生まれる。
各中間層では順伝播と逆伝播それぞれの出力の結合ベクトルに対して重みを掛け、活性化関数を通すことで、出力が生まれる。

Seq2Seq

2つのネットワーク化ドッキングして使われている自然言語処理のモデル。ある文章を入力とし、別の自然な文章を出力する事を目指したモデル。エンコーダーとデコーダーという2つのネットワークからなる。
1)エンコーダー
エンコーダの学習が進むと、その文脈の意味が意味ベクトルとして符号化される。エンコーダーはこのベクトル値が真の文脈をベクトル化した正解値に近づくように学習が進む。
〇文章の意味の符号化
まずは1つ1つの単語をベクトル化する（embedding表現）。この時、各単語の意味の近さがベクトルの近さとして保たれるように、ベクトル化を行う（このベクトル化自体が大きな課題であり、Masked Language Modelなど機械学習による特徴量抽出で行われる）。単語のembeddingさえ出来れば、各単語をRNN処理すれば、全単語の意味情報を考慮した文脈ベクトルが出力される。
〇Masked Language Model
文章を単語毎に分解し、どこか一つの単語をマスクする。その上でマスクされた部位に自然に当てはまる単語が何かを学習する。当てはまる単語は意味ベクトルが近くなるように学習される。これにより単語のembeddingが可能となる。意味ベクトルの正解値を与える必要がない、教師無し学習である。
2)デコーダー
次にデコーダーはエンコーダーが出力した意味ベクトルを入力に受け取り、確率に従って最初の1単語目を生成する。2単語目は1単語目の出力と1単語目の入力を元に出力される。このように再帰的な出力をすることで、ある意味入力に対して別の文章が生成される。デコーダーではその文章が自然となるように学習を進める。

Word2vec

単語のembedding表現を得るための手法。下記にその概要の手順で示す。

1. 単語をone-hotベクトル表現にする。
2. 単語のone-hotベクトルをembedding表現に変換する重みパラメータ（単語の分散表現ベクトル）を定義する。
3. 学習により重みパラメータの最適化を行う。
4. 学習後の重みパラメータを、目的の単語のone-hotベクトルに掛けることでembedding表現をえる。

■参考書レポート

ゼロから作るDeep Learning2 第3章ward2vec

要点

近年の単語ベクトル化の手法のほとんどが分布仮説「単語の意味は、周囲の単語によって形成される」に基づく。その手法の主なものとして「カウントベースの手法」と「推論ベースの手法」とがある。カウントベースの手法は計算量が膨大な点が課題であるが、推論ベースの手法はそれを解消している点でメリットがある。

推論ベースの手法ではコンテキストを入力として、それに挟まれる単語の出現確率を予測する。その予測精度が上がるように学習を進める。

Attention Mechanism

seq2seqは入力文章の長短に関わらず、その意味を固定次元のベクトルに出力する。そのため長い文章が入力の時はうまく機能しない。それを解消したのがAttention Mechanismである。Attention Mechanismでは、入力文章と出力文章の各単語間で関係性が強いものを学習し、それを選択的にseq2seqの隠れ層に用いることで、文章の意味ベクトルの次元が固定されていても、長い文章を扱えるようになった。

day4

強化学習

機械学習は教師あり学習、教師無し学習、強化学習の3つに分類される。

強化学習では方策関数と行動価値関数を定義して、このパラメータを学習する。

目的関数は、方策関数と行動価値関数をもとに算出した報酬の期待値となり、これの最大化することを目指して、行動価値関数と方策関数を学習する。

・行動価値関数

状態と行動とを入力として価値を出力する関数。価値とは、エージェントが今の方策を続けたときの価値の予測値である。

・方策関数

エージェントがとる都度都度の行動を決めるための関数。行動価値関数の出力を最大化するように方策関数は学習が進む。ある状態で実際にどのような行動をとるかの確率を出力することで、行動が決定される。

AlphaGo

囲碁で勝つことを学習させたニューラルネットワーク。

1)AlphGo Lee

方策関数

盤面特徴を入力とし、畳み込み層、ReLU関数を重ね、最後にソフトマックス層で19×19マスの着手確率を出力

価値関数

盤面特徴を入力とし、畳み込み層、ReLU関数を重ね、最後に全結合（tanh）で現局面の勝率を出力

上記2つのニューラルネットワークを学習させるに当たり、最初から強化学習では上手くいかないので、下記のステップが踏まれている。

①過去の棋譜データを教師データとして、人間と同じ手が打てるように方策関数を学習。

②強化学習によって方策関数を学習

③強化学習によって価値関数を学習

2)AlphaGo Zero

AlphaGo Leeの後に出たニューラルネットワークで以下の特徴をもつ。

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

③方策関数と価値関数を1つのネットワークに統合

④Residual Netを導入

軽量化・高速化技術

1)分散化

・データ並列化

　親モデルを各ワーカーにコピーし、各ワーカーに分割したデータをそれぞれ学習させる。

・モデル並列化

　親モデルを分割して各ワーカーに割り当て、各ワーカーがそれぞれ学習する。全てのデータで学習が終わった後で、各学習済みモデルをワーカーから取得し、1つのモデルに復元する。モデルが大きい時に効果的な手法。

・GPUによる並列化

　CPU…複雑な処理が得意な高性能コアが少数

　GPU…カンタンな処理が得意なコアが多数。ニューラルネットの単純な行列演算を高速処理可能。

2)モデル軽量化

・量子化

　通常64bitで扱うパラメータを、32bitや16bitで扱うことで、メモリと演算処理の削減をする試み。

　精度低下が起こりうるが、ほとんど精度が保たれたまま高速化できることもある。

・蒸留

　学習済みの精度の高いモデルの知識を軽量なモデルに継承させる

・プルーニング

モデル精度に寄与が少ないニューロンを削減することで、軽量化・高速化をする。重みが閾値以下のニューロンを削減し、再学習を行う。

応用モデル

MobileNet

軽量化・高速化・高精度を実現したCNNモデル。一般的な畳み込み処理を、チャネルごとに畳みこむDepthwiseConvolutionと、入力マップのポイントごとに畳みこむPointwiseConvolutionに分解することで計算量を削減。このような分解を行っても精度は保たれた。

DenseNet

ResNet同様、前方の層から後方の層へのアイデンティティ接続を介してパスを作る事で勾配消失を軽減。ResNetでは前1層の入力のみ後方の層へ入力させるのに対し、DenseNetでは前方の各層からの出力全てが後方の層へ入力させる。

Wavenet

時系列データに対して畳み込み（Dilated Convolution）を適用。層が深くなるにつれて畳みこむリンクを離し、パラメータ数に対する受容野が広い。

Transformer

TransformerとはRNNの代わりにAttentionを用いたseq2seq。RNNを使わないので計算量が大幅に削減されるにもかかわらず、圧倒的な性能を発揮する。エンコーダーではself-Attentionが、デコーダーではself-Attentionとsource target Attentionが用いられる。

self-Attention

Attentionのクエリ、キー、バリューがすべて同じ入力から設定。翻訳においては入力の文章における各単語間の関連性を確率にしたものが行列で出力される。

source target Attention

Attentionのクエリが翻訳先、キー、バリューが翻訳元から設定される。翻訳先と元の言語間で、単語間の対応関係が確率分布の形で出力される。

TransformerではRNNを用いないので、単語の語順情報をエンコードした上で入力（単語のベクトル表現）にconcatする。

■実装演習

Tanaka Corpusを用いた英語→日本語翻訳のseq2seqを、RNNおよびTransformerのそれぞれで実装。同じエポック数10で比較した際、処理速度は同等（0.3min/epoch）だが翻訳精度BLEUはTransformerの方が大きい（BLEU：Transformer 33.2、RNN 16.3）。

1)RNNで実装した場合

```
Epoch 1 [0.3min]: train_loss: 44.60  train_bleu: 7.49  valid_loss: 44.93  valid_bleu: 6.45
----------------------------------------------------------------------
Epoch 2 [0.3min]: train_loss: 40.49  train_bleu: 10.92  valid_loss: 42.53  valid_bleu: 9.90
----------------------------------------------------------------------
Epoch 3 [0.3min]: train_loss: 37.60  train_bleu: 13.68  valid_loss: 41.15  valid_bleu: 12.47
----------------------------------------------------------------------
Epoch 4 [0.3min]: train_loss: 35.63  train_bleu: 15.81  valid_loss: 40.26  valid_bleu: 12.77
----------------------------------------------------------------------
Epoch 5 [0.3min]: train_loss: 33.41  train_bleu: 18.57  valid_loss: 40.37  valid_bleu: 15.95
----------------------------------------------------------------------
Epoch 6 [0.3min]: train_loss: 32.19  train_bleu: 20.14  valid_loss: 40.07  valid_bleu: 15.75
----------------------------------------------------------------------
Epoch 7 [0.3min]: train_loss: 30.70  train_bleu: 22.01  valid_loss: 40.20  valid_bleu: 16.62
----------------------------------------------------------------------
Epoch 8 [0.3min]: train_loss: 29.30  train_bleu: 24.36  valid_loss: 40.03  valid_bleu: 14.45
----------------------------------------------------------------------
Epoch 9 [0.3min]: train_loss: 28.47  train_bleu: 25.46  valid_loss: 40.96  valid_bleu: 17.39
----------------------------------------------------------------------
Epoch 10 [0.3min]: train_loss: 27.06  train_bleu: 27.96  valid_loss: 40.77  valid_bleu: 16.62
----------------------------------------------------------------------
```

2)Transformerで実装した場合

```
Epoch 1 [0.3min]: train_loss: 77.31  train_bleu: 4.69  valid_loss: 41.28  valid_bleu: 10.66
----------------------------------------------------------------------
Epoch 2 [0.3min]: train_loss: 39.42  train_bleu: 12.25  valid_loss: 32.22  valid_bleu: 17.45
----------------------------------------------------------------------
Epoch 3 [0.3min]: train_loss: 32.05  train_bleu: 18.01  valid_loss: 28.00  valid_bleu: 22.65
----------------------------------------------------------------------
Epoch 4 [0.3min]: train_loss: 28.32  train_bleu: 21.62  valid_loss: 25.83  valid_bleu: 25.13
----------------------------------------------------------------------
Epoch 5 [0.3min]: train_loss: 25.85  train_bleu: 24.50  valid_loss: 24.36  valid_bleu: 26.89
----------------------------------------------------------------------
Epoch 6 [0.3min]: train_loss: 24.00  train_bleu: 26.76  valid_loss: 23.12  valid_bleu: 28.73
----------------------------------------------------------------------
Epoch 7 [0.3min]: train_loss: 22.57  train_bleu: 28.48  valid_loss: 22.16  valid_bleu: 30.38
----------------------------------------------------------------------
Epoch 8 [0.3min]: train_loss: 21.37  train_bleu: 30.15  valid_loss: 21.46  valid_bleu: 31.10
----------------------------------------------------------------------
Epoch 9 [0.3min]: train_loss: 20.34  train_bleu: 31.52  valid_loss: 20.95  valid_bleu: 31.88
----------------------------------------------------------------------
Epoch 10 [0.3min]: train_loss: 19.43  train_bleu: 32.81  valid_loss: 20.31  valid_bleu: 33.23
----------------------------------------------------------------------
```

■参考書レポート

ゼロから作るDeep Learning2　第8章Attention

要点

機械翻訳の例でAttentionを考えると、「翻訳先の単語」と対応関係にある「翻訳元の単語」の情報を選び出すこと、そしてその情報を利用して翻訳を行う仕組みがAttentionである。なお単語の対応関係を表す情報はアライメントと呼ばれ、Attention以前は人手で作られてきた。逆に言うと、Attentionではアライメントを機械で自動で生成することに成功している。

物体検知・セグメンテーション

物体検知・セグメンテーションでは物体の分類に加え、物体の位置が出力に福あれる。

広義の物体認識タスク（画像を入力とした場合）

・Classification(分類)

　出力：1つのクラスラベル

・Object Detection(物体検知)

　出力：Bounding Box[bbox/BB]

・Semantic Segmentation(意味領域分割)
　出力：クラスラベル（ピクセル毎）
・Instance Segmentation(個体領域分割)
　出力：クラスラベル＋Instanceの分類（ピクセル毎）

Bounding Box
　検出した物体の「位置」「分類」「Confidenc」eの3つを入力画像中に重ねて表示。

代表的なデータセット
クラス数とBox/画像を軸に、4つの主要なデータセットがある。目的に応じた適切なデータセットの選択が必要。
・VOC12
　クラス数：少、Box/画像：少
・ILSVRC17
　クラス数：多、Box/画像：少、　ImageNetのサブセット
・MS COCO18
　クラス数：少、Box/画像：多
・OICOD18
　クラス数：多、Box/画像：多、　Open Images V4のサブセット

物体検知の評価指標
評価指標としてmAPやmAP_{COCO}が考えられている。
・mAP…クラス毎のAPの平均
・mAP_{COCO}…0.5～0.95まで0.05刻みでIoUを固定した時のmAPの平均
これら指標を算出する際の前提の考え方を以下にまとめる。
1)IoU（BB位置の予測精度）
　予測と実際の重なり度合：IoU(Intersection over Union)=Area of Overlap/Area of Union
2)Precision-Recall
　Confidenceの閾値以上のBBが出力される。
　　出力BBの数＝TP + FP　※IoUが閾値以上がTP、閾値未満がFP
　IoU閾値0.5で固定した時のPreision-Recall曲線の下側面積（AP）が、あるクラスにおける検知精度