【CNN】

畳み込みニューラルネットワークと呼ばれ画像認識タスクでよく使われる。畳み込み層・プーリング層からなり。畳み込み層で特徴が抽象化され抽出される。CNNでは2次元の情報をそのまま引き継いで学習を進めることが出来る。CNNで有名なモデルに

AlexNetがある。AlexNet は5層の畳み込み層及びプーリング層などそれに続く3層の全結合層から構成されている。過学習を防ぐために、4096サイズの全結合層の出力にドロップアウトを使用している。

・練習問題

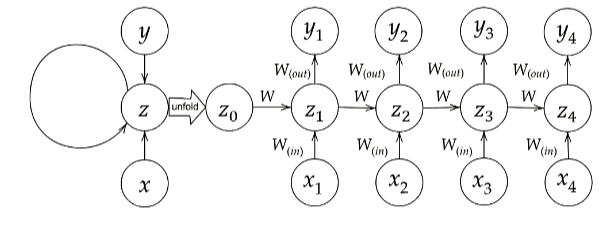
入力画像5×5　パッティング1、ストライド2、3×3のフィルタで畳み込み演算した場合の出力サイズは3×3である。

【RNN】

RNNとは、時系列データに対応可能な機械学習モデルである。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、相互に統計的依存関係が認められるようなデータ系列のことを意味する。具体的な時系列データには、音声データや、テキストデータ、株価のデータなどがある。時間的なつながりがあるテキストデータは単語一つ一つがある時刻のデータとして扱う、前の単語と次の単語は関連性があるので、時系列データとみなせる。

【RNNのアーキテクチャ】

RNNは入力層、中間層、出力層から構成される。時間的なつながりをうまく学習させるために、中間層の出力を再び中間層の入力とする。下図に示すように、前の時間の中間層の出力を次の中間層の入力として用いる。中間層では再帰的な入出力処理を行うことで、時系列関係を学習する。中間層では、前の時刻の中間層の出力と現時刻の入力から、出力を計算する。RNNでは重みが、入力層から中間層までの重み、中間層から出力層までの重み、そして、前の中間層からの出力から現在時刻の中間層の入力までの重みの3つの重みが存在する。



・練習問題

Q）RNNのネットワークには大きく分けて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。

1. 前の時刻の中間層から次の時刻に中間層に至る重み

RNNで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。再帰構造はRNNのR（Recalent）に由来する。

サンプルプログラム）バイナリ加算

2進数同士の足し算では桁上がり（繰り上がり）がある。小さい桁から大きい桁へ計算していくことで発生し、時系列データの処理とみることが出来る。繰り上がりという処理をRNNの機械学習モデルに学習させてモデルを構築してみる。時間的なつながりの全時間（サンプルでは8単位時間分）の学習をやって1回の学習分となる。逆伝播の学習でも1回の学習で8単位時間分の学習が必要である。

・演習チャレンジ

テキストデータにおいて隣り合う単語同士の特徴量を合成する。このとき、2つの入力ベクトルをつなげる（concatnate）ことで、各々のベクトルの特徴量を保持したまま次の学習が可能である。重みのサイズを調整すれば、ベクトルサイズが膨大になることを防ぐことが出来る。

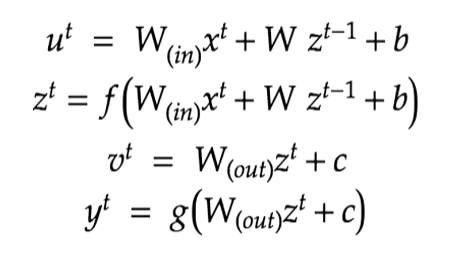
・誤差逆伝播法の復習

計算結果、誤差から微分を逆演算することで、不要な再帰計算を避けて微分を計算できる。誤差の計算では、誤差関数、（二乗誤差関数）を利用する。各層の重みに対する誤差の微分は連鎖律を用いることで遡って計算可能となる。

・確認テスト

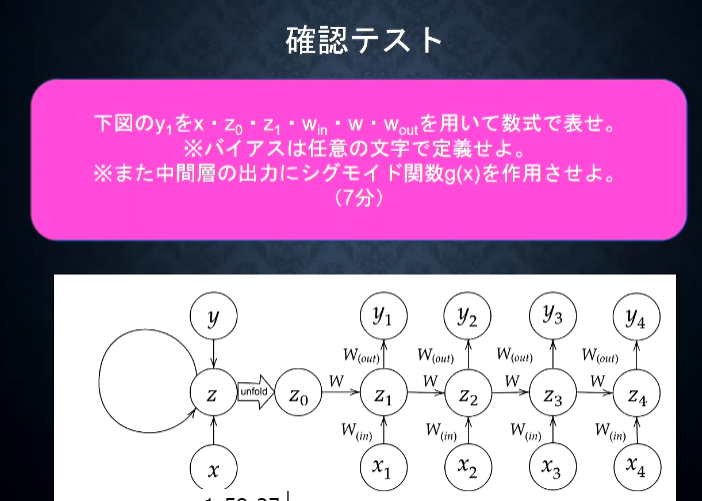
ｚ＝ｔ＾2　ｔ＝ｘ+ｙの時、dz/dt=2t dt/dx=1よりdz/dx＝dz/dt \* dt/dx =2(x+y)である。

【BPTT（バックプロパゲーションスルータイム）】



RNNの逆伝播計算方法を説明する。上図に占めす誤差客伝播の式において3つの重みについて微分する。誤差Eは入力の重み、出力の重みの式から誤差を計算する。u,vは出力ｙについての関数ではないが、微分を長くたどれば微分計算可能で誤差を算出できる。これをまとめてEをu,vについて微分し、δt,δout\_tと表す。各時間すべてに対して、微分するため、Ｔの添え字をつけておく。バイアスについての微分は、δそのものとなる。δtは、前の時刻のδt-1であらわされておち関係式が成り立つ。

確認テスト



1. y1=g(Wout+s1+c),s=Winx1+Ws0+b

b,cはバイアス

中間層の値は、過去の時刻の値の特徴を持ち合わせている。パラメータの更新においては、中間層に至るまでの重みWin、前の中間層から、次の中間層に至る重みでは、過去のすべての中産層の項を足し合わせることが特徴である。中間層から出力に至る重みは出力の時点ではすでに、過去すべての中間層の値を反映済みなので、時間的なさかのぼりを考慮しない。

・BPTTの全体像

中間層の項を介して、式が連鎖的につながっている。これが過去の知識を学習するモデル構造である。

・プログラムの確認

バイナリ加算を使ったRNNの実装サンプル

時間のforループで時系列の計算を行っている。それぞれの時刻において誤差計算を行う。各自刻の誤差の総和をとって全体の誤差を計算する。勾配更新計算では、前述のとおり、3つの重みに関して更新量を計算する。

コードの演習問題



dh(t)/dh(t-1)-Uから過去にさかのぼるごとにUがかけられているので正解は2である。

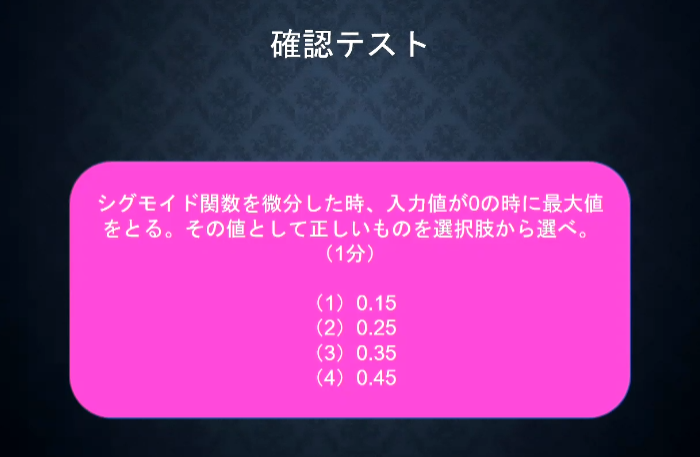
【LSTM】

RNNの課題として時系列をさかのぼればさかのぼるほど、勾配が消失していく。（勾配消失問題）よって、長い時系列の学習が困難となる。そこでネットワークの構造自体を変えてこの問題を解決したモデルがLSTMである。

・勾配消失問題の復習

複数の層が何層もあるニューラルネットの場合、誤差逆伝播法によって下位の層に進んでいくにつれて、勾配がどんどん小さくなる。そのため、勾配降下法による更新では、階層のパラメータはほとんど更新しても変化せずに、学習によって最適値に収束しなくなる。

シグモイド関数は0～1の値しかとりえないので、勾配消失問題を引き起こす活性化関数である。層が多くなると、誤差逆伝播において微分の連鎖律でシグモイド関数を何回もかけることになり、値がどんどん小さくなる。



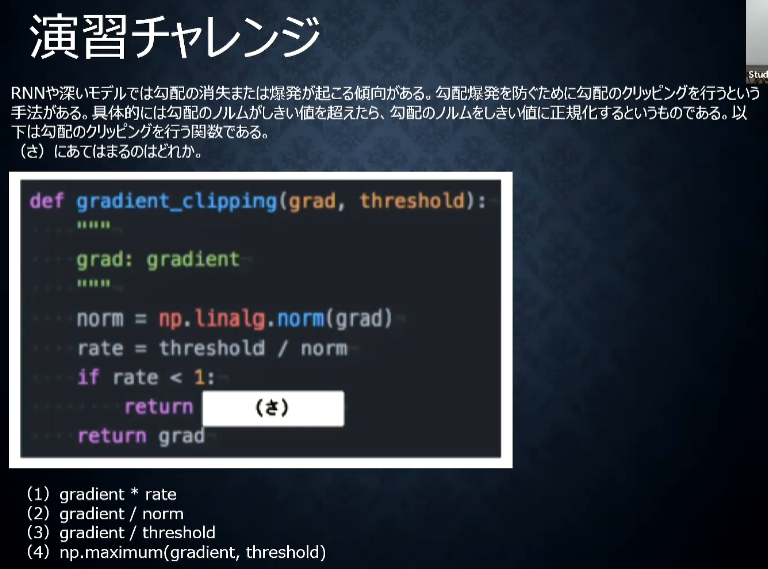
答えは2番。0.25最大値として正負中心に対象に減衰する

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現象の事。

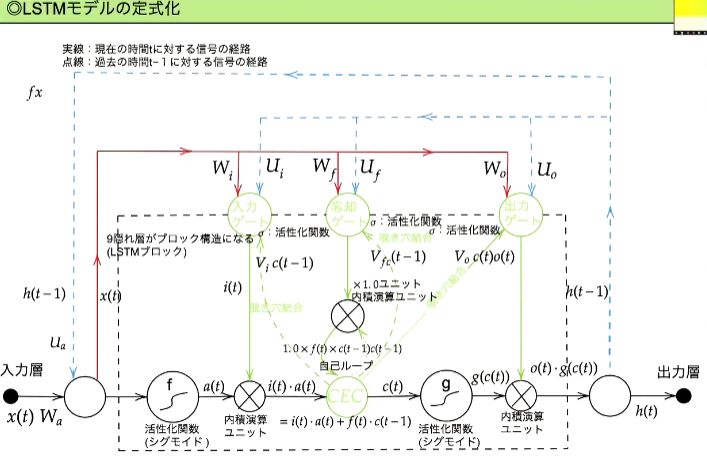
例えば恒等関数が活性化関数である場合発生する。学習率の設定で極端に大きい値を設定してしまうと、勾配爆発が発生することが多い。

・チャレンジ演習



Ifの条件文内は勾配のノルムが閾値を超えた場合の処理であるから、勾配のノルムを閾値に正規化する計算に該当する。閾値に正規化するためには、threshold/norm\*grad=rate\*gradの計算が必要である。

【LSTMのアーキテクチャ】



上図がLSTMの全体構造である。青い点線が、時間的なループを表し。黒い点線部が中間層の部分を示す。

【CEC】

CECはLSTMの重要な層である。CECは過去の時刻のデータを記憶する役割のみを果たしている。これまでのRNNでは記憶と、学習が同時に行われていたので、それらを分離するという発想から考案された。従来のRNNは層が深くなると、勾配消失、および勾配爆発が発生してしまうが、勾配を1とすることで勾配消失・勾配爆発を回避できる。課題としては、入力データについて時間依存度に関係なく重みが一定となってしまい、ニューラルネットワークの学習特性がない。そこで、CECの周りに学習機能をもった層を配置する。

【入力ゲートと出力ゲート】

CECに入力されたデータをどれだけ記憶させるかを入力ゲートで制御し、CECで記憶したデータをどれだけ取り出し使うかを出力データで制御する。入力ゲートはCECへの記憶のさせ方を学習する。出力ゲートではCECの記憶データを使えばよいかを学習する。

入力ゲートは、元時刻の入力データと、前回の出力値をもとに、今回の入力データをどう記憶させるかを計算する。図中ではViがどれくらい記憶させるかを表している。出力ゲートでは、現在の入力と前回の出力データから現在のデータをどう使うかを学習する。

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可変可能とし、学習機能がないというCECの課題を解決できる。

【忘却ゲート】

入力ゲートと出力ゲートとCECだけでは、CECは過去の情報がすべて記憶されている状態である。過去の情報が不要となった場合、削除できず保存され続けてしまうことが課題であり、あまりにも古い情報に左右されてしまう恐れがある。解決策としては、過去の情報がいらなくなった場合、そのタイミングで情報を忘却する機能を付加すればよい。これが忘却ゲートの役割である。忘却ゲートも入力ゲートと出力ゲートと同様に、今回の入力データと前回の出力値に基づいて、ECEの記憶データをどれだけ忘れさせるかを学習する。

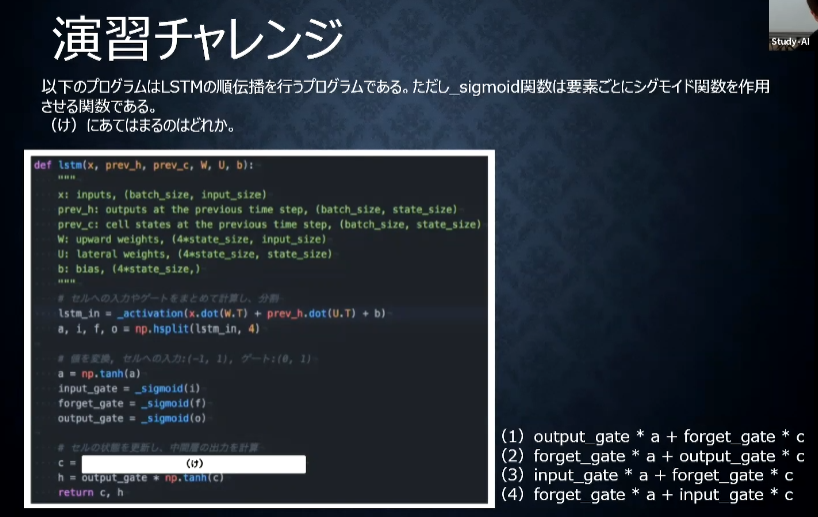
C（ｔ）は現時刻でCECが覚えている情報。A(t),i(t)元時刻の情報、f(t)がどれだけ忘れるかという情報を示す。

確認テスト



「とても」は情報としては不要となったので忘却ゲートが作用すると考えられる。

チャレンジ演習



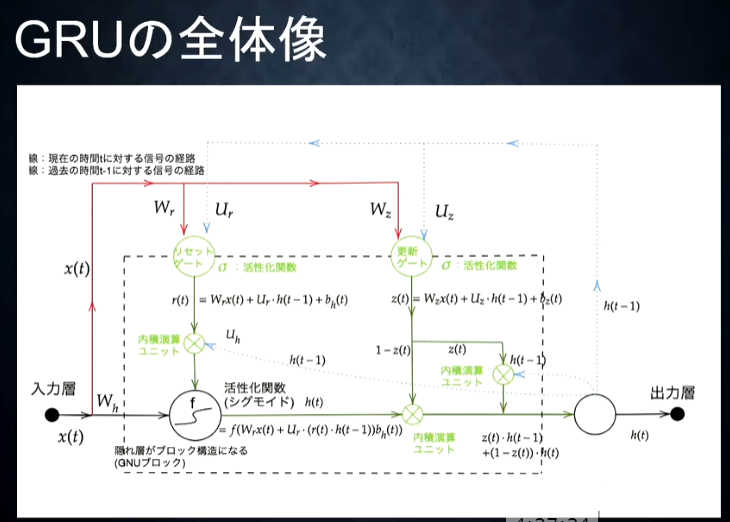
セルの状態を更新する計算となるので、回答は3である。

【覗き穴結合】

LSTMの課題としてCECの保存されている過去の情報を、任意のタイミングでほか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場合がある。CEC自身の値は、ゲート制御に影響を与えていない。このためCECの状態も入力ゲートと出力ゲートと忘却ゲートの学習の判断材料として利用する。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である。しかし、実際にはあまり効果は得られないことが多い。

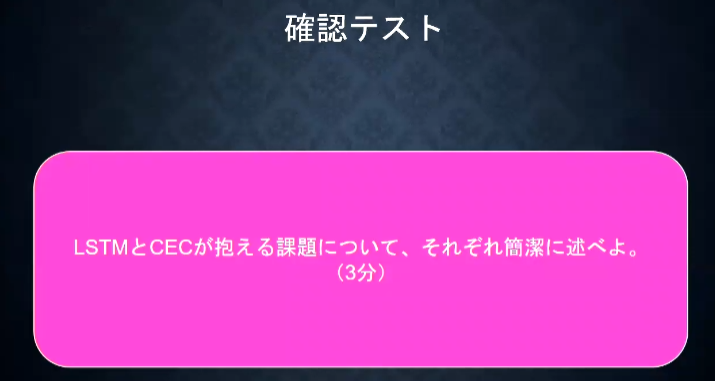
【GRU】

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。このため、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ようになった構造となっている。メリットとしては計算負荷が小さいことがあげられる。



上図にGRUの全体構造を示す。LSTMとの違いは、CECと忘却ゲート、入力ゲートと出力ゲートがなくなり、リセットゲートと更新ゲートがあることである。隠れ層として計算状態を保存し、リセットゲートでは、隠れ層の状態をどのような状態で保持するかということを制御する。リセットゲートと更新ゲートは今回の入力と前回の出力値で学習を進める。更新ゲートでは前回の記憶と今回の記憶をどれくらいの重みで出力するかを制御する。

確認テスト



LSTMの課題：4つのゲートがあり、パラメータ数が多く、計算量が多い

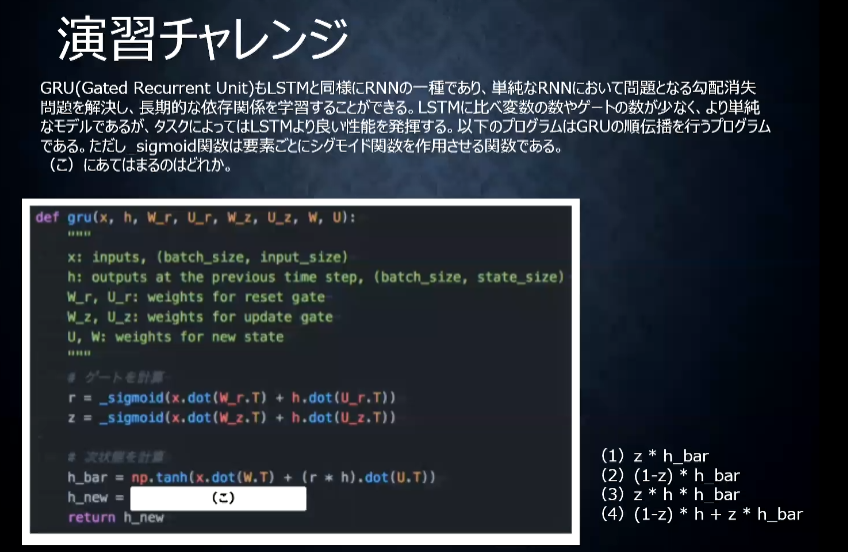
CECの課題：勾配が1であり学習能力がない

【プログラムの実装】

predict\_word.ipynb

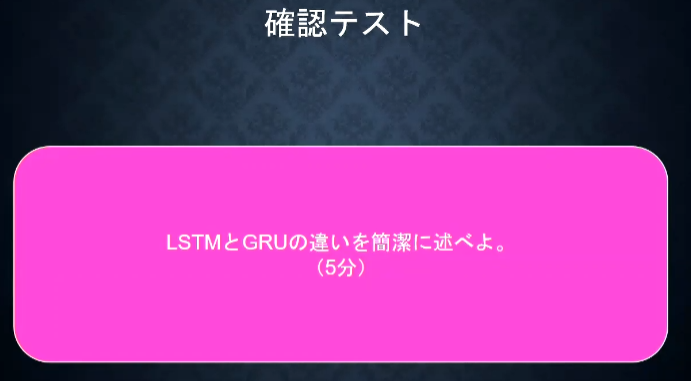
tensor\_flowによるRNNのモデル実装。tensor\_flowは機械学習用のPython高級ライブラリ

チャレンジ演習



前述のGRUの全体構造の図と見比べると正解は（4）となる。

確認テスト



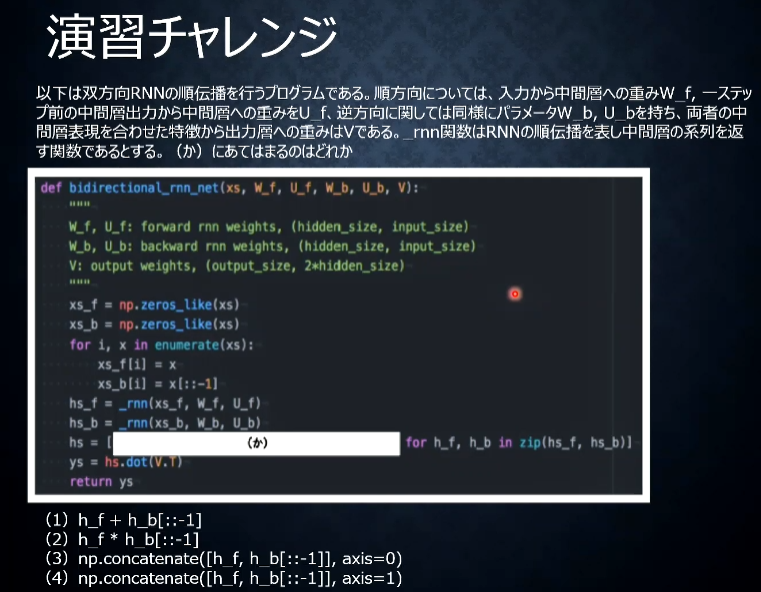
LSTM：入力ゲート出力ゲート忘却ゲートの3つのゲートに加え、CECがある　パラメータが多く計算量が多い

GRU：CECがない。デートはリセットゲートと更新ゲートのみ。パラメータが少なく計算量が少ない

【双方向RNN】

双方向RNNとは、過去の情報だけでなく、未来の情報も加味することで、精度を向上させるモデル。実用例としては、文章の推敲や機械翻訳等がある。文章では過去未来の情報が一度にすべて入力されるので双方向RNNが適用できる。過去から未来の方向の学習に加え、未来から過去への方向での学習も新たに加えより精度を上げることに成功した。

チャレンジ演習



足したりかけたりすると情報が消えてしまうので、（1）と（2）は選択肢から消える。

np.concatnateは配列結合を行うが、axis＝0の場合、同じ行の次元に結合し、axis＝1の場合、新たな次元（列）に結合する。

同じ時刻の重みは同じ次元に入るようにしたいため、axis=1で結合する。正解は（4）である後のhs.dot(V.T)の計算と、Vの次元が（output\_size,2\*hiddensize）であることからも次元の整合性をとるように考えても（4）の選択肢を選べる。

【RNNの応用例】

・seq2seq

二つのRNNがドッキングして構成されている。機械翻訳によく用いられるモデルである。一つ目のRNNでは、単語の列が順々に入力される。RNNでは隠れ層にデータが蓄積される。この蓄積されたデータは、これまでに入力されたデータの記憶がされる。隠れ層には、文章の意味がベクトルとして、保存される。これを別のRNNの入力にする。もう一つのRNNではベクトルから、別の出力（翻訳したい言語）へ返還される。

前者のRNNはエンコーダ、後者をデコーダと呼ぶ。

・Encoder RNN

文の意味を集約する役割がある。ユーザ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となっている。文の意味が1つのベクトルとして抽出され保存される。自然言語のベクトル化について、

Talking：文章を単語等のトークンごとに分割し、トークンごとのIDに分割する。文章の各単語に番号（ID）をつけ、one-hotベクトルとして単語の並びを表現する。

Embedding：IDから、そのトークンを表す分散表現ベクトルに変換する。One-hotベクトルはあまりにも0が多く表現として無駄があるので、情報を削減する。単語の意味が似ているものは、似通ったベクトル表現となるように機械学習を行う。このときVertや

MLM（Masked　Language Model）などが用いられる。MLMは、モデルを学習するときに、ある単語を隠し、その隠した単語を周囲の単語から予測できるように学習する。これにより、単語の意味ベクトルを自力で獲得することが可能となる。教師なし学習であり、大量な文章に対して学習が可能であり、精度向上につながっている。

Encorder RNN：ベクトルを順番にRNNに入力していく。

実際の処理

・vec1（単語）をRNNに入力し、hidden state（隠れ層）を出力する。このhidden state と次の入力vec2をまたRNNに入力してきたhidden stateを出力するという流れを繰り返す。

・最後のvecを入れたときのhidden stateをfinal stateとして取っておく。このfinal stateがthought vectorと呼ばれ、入力した文章の意味を表すベクトルとなる。

・Decoder RNN

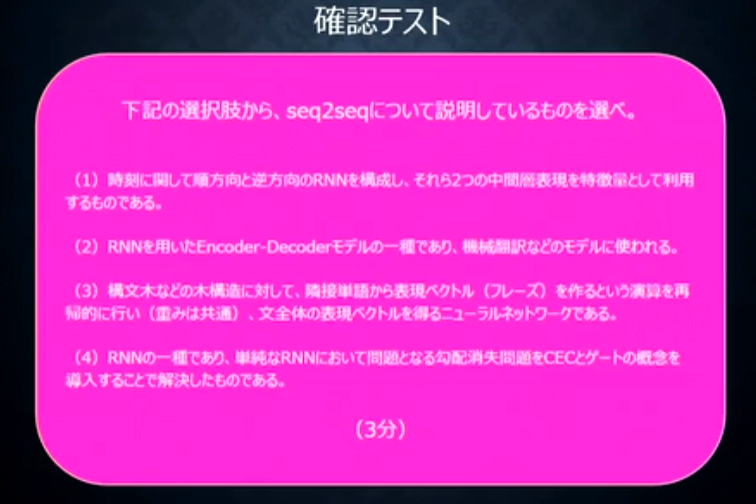
システムがアウトプットデータを、単語等のトークンごとに生成する構造となっている。Encorder RNNが出力した隠れ層（意味ベクトル）をもとに文脈を構成する。

処理手順

1. Decoder RNN:Encorder RNN のfinal state(throught vector)から、各トークンの生成確立を出力する。Final stateをDecoder RNNのinitial stateとして設定し、Embeddingを入力する。
2. Sampling：生成確立に基づいてトークンをランダムに選ぶ
3. Embedding：2で選ばれたトークンをEmbeddingしてDecoder RNNへの次の入力とする。
4. Detokenize：1-3を繰り返し2で得られたトークンを文字列に直す。

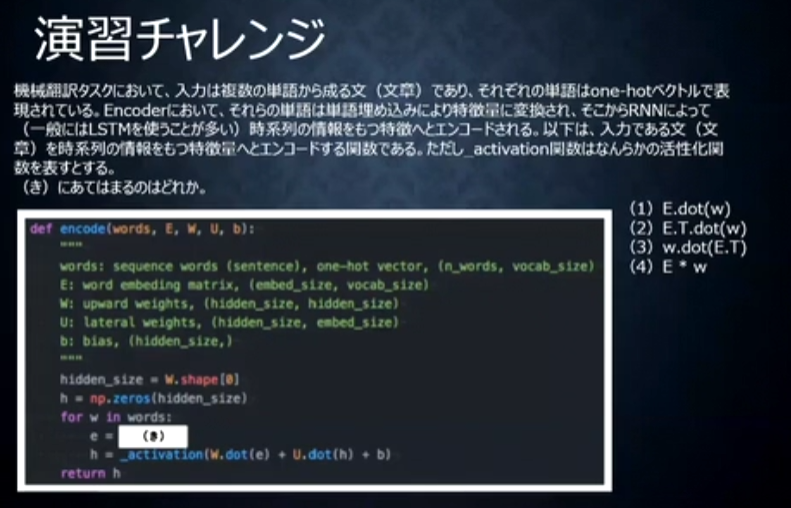
単語の並びから別の単語の並びが出力される（seq2seq）

確認テスト



1. 双方向RNNの説明である
2. 正解
3. 構文木の説明である
4. LSTMの説明である。

チャレンジ演習



E:embedding表現のベクトルにベクトルドット積をすることで、単語を取り出すことが出来る。正解は（1）である

【HRED】

Seq2seqの課題として、1問1答しか処理できないことがあげられる。問いに対して文脈も何もなく、ただ応答が行われ続ける。これまでの文章の文脈を何かしらの意味として取れないかと考案された手法がHREDである。HREDでは、過去n-1個の発話から次の発話を生成する。Seq2seq（単語から単語）では、会話の文脈無視で応答がなされていたが、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。Seq2seqのエンコーダから文の意味ベクトルを次のエンコーダに引き継ぐ。Seq2seqを一塊とみて、これをＲＮＮのように層を連ねることで、文章間の意味を学習させる。HREDの構造はseq2seqとＣontext RNNから構成される。Ｃontext RNNでは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となっている。これにより過去の発話の履歴を加味した返答ができる。しかしＨＥＲＤは確率的な多様性が字面にしかなく、会話の「流れ」のような多様性がない。すなわち、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。また、ＨＲＥＤは短く情報量に乏しい答えを出力する傾向がある。すなわち短いよくある答えを学ぶ傾向がある。

【ＶＨＲＥＤ】

ＨＲＥＤに、ＶＡＥの潜在変数の概念の加味したものであり、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となっている。

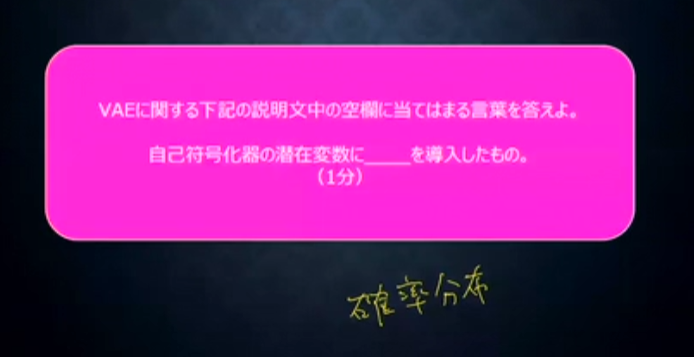
【オートエンコーダ】

教師なし学習の1つ。人間が正解不正解のラベル付けをする必要なく学習可能なモデルである。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を必要としない。オートエンコーダの具体例としては、MNISTの場合、28×28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がある。構造としては、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と呼ぶ。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と呼ぶ。zは入出力の画像データに比べ、圧倒的にサイズが小さい。入力と出力が同じになるように学習を進める。Encoderだけを見れば、zの次元が入力データより小さい場合、次元削減をしているみなすことが出来ることができメリットの1つである。

【ＶＡＥ】

通常のオートエンコーダの場合、潜在変数zにデータ圧縮をしているものの、その構造がどのような状態かわからない。このため、ＶＡＥではこの潜在変数zに確率分布z-N(0,1)を仮定したものを用いる（正則化）。これにより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が可能になる。元のデータがどれだけ似通っているかという情報の欠落を回避できる。学習では、ノイズを加えてデコータ処理を行う。これにより、汎用性の高い特徴の獲得が可能である。

確認テスト



「確率分布」という言葉が入る。

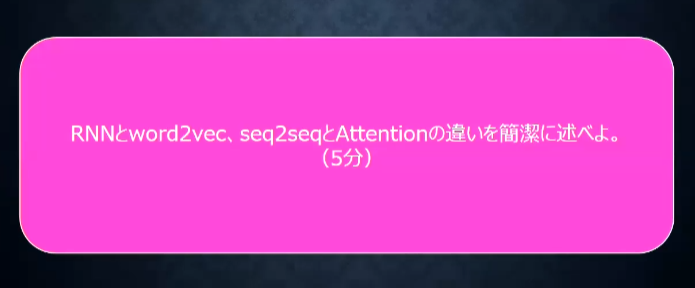
【word2vec】

word2vecは単語をベクトル表現する手法(Embedding)の一つである。ＲＮＮでは、単語のような可変長の文字列をニューラルネットワークに与えることはできないという課題がある、これを解決するためには、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し、各ボキャブラリに番号を振って、one-hot-ベクトルを作る。このベクトルから、Embedding表現をする行列を生成する。単語同士の意味を保ちながら、次元の少ないベクトルを生成するように学習を進める。メリットとしては、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となることである。

【Attention Mechanism】

seq2seqでは長い文章への対応が難しく、2単語でも、100単語でも、固定次元ベクトルに入力しなければならないことが課題である。これに対し、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を作る必要がある。これがAttention Mechanismであり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となっている。

確認テスト



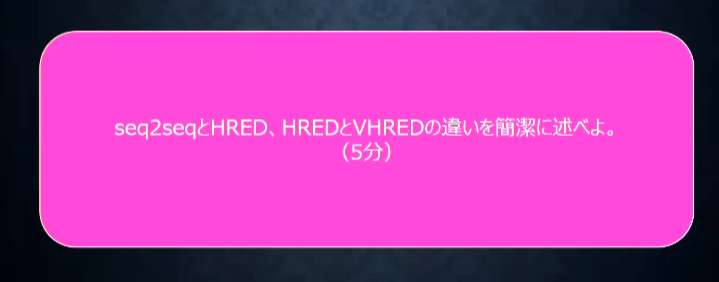
RNNは時系列データを処理する場合に適したニューラルネットワークである。

Word2vecは単語の分散表現ベクトルを得る手法である

Seq2seqは一つの時系列データから別の時系列データを得るニューラルネットワークである。

Attentionは時系列データの中身のそれぞれの単語に対して、関連性に重みをつける手法である。

確認テスト



Seq2seqは一問一答に対して処理が可能なネットワークである。

HREDはSeq2seqの機構にこれまでの文脈の意味ベクトルを解釈に加えられるようにすることで、文脈の意味をくみ取った文の変換を可能としたモデルである。

VHREDはHREDの文脈に対する回答の多様性が乏しいという課題を解決する手法であり、オートエンコーダ（VAE）の考え方を取り入れて、改良を施したモデルである。

【強化学習】

これまでの教師あり/教師なし学習では、データの特徴を見つけることが目標だった。これに対し強化学習では、目的を達成するための行動を自力で何とか見つけ出すという人間が一般に行っている行動を模倣している。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中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である。下図が強化学習の概念図である。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を採用している。強化学習では、環境とエージェント（モデル）があり、エージェントは環境において方策（Π）を訓練、決定する。そして方策に基づいて行動し、高い報酬がもらえるように、すなわち価値（V）が高くなるように行動を決定する。環境の状態（S）は変化するものであり、これに対応するようにモデルの学習を進める。



【強化学習の応用例】

マーケティングの場合は環境が会社の販売促進部であり、エージェントがプロフィールと入力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。そして行動は顧客ごとに送信する/しないの二つの行動を選ぶことが出来る。報酬（目標）では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り上げという正の報酬を受ける。キャンペーンを買ってくれる人をうまく選んで、メールを送ることを強化学習のモデルで実行する。最初のうちは無差別にメールを送るが、次第にキャンペーンを買ってくれる傾向がある顧客に対してメールを選択的に送ることが出来るようになる。

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が可能である。どのような顧客にキャンペーンメールを返信するとどのような行動を送るかが既知であるためである。しかし強化学習の場合、上記仮定は成り立たないとする。不完全な知識をもとに行動しながら、データを収集し最適な行動を見つけていく。どれだけ探索するかとどれだけ既存の知識を活用するかはトレードオフの関係である。過去のデータベースで、ベストとされる行動のみを常に続けていれば、他にもっとベストな行動を見つけることはできない、すなわち探索が足りない。かといって、未知の行動のみを常にとり続ければ過去の経験が生かせない。すなわち利用が足りない。このトレードオフに対して強化学習では適切に調整する機能を持つ。

方策（Π）と価値（V）について、どういう条件が最も良いかを設定し学習させることが強化学習の運用上重要であり、方策関数と行動価値関数を数学的に定義し、学習させる。

確認テスト

Q強化学習は教師あり学習と教師なし学習と何が違うか？

A目的が違う。教師なし/あり学習では、データに含まれるパターン（特徴量）を見つけ出し、そのデータ、特徴量から予測することが目標である。一方強化学習では優れた方策（行動指針）を見つけることが目標である。

【強化学習の歴史】

・昔のコンピュータでは計算能力が追い付かず、機械学習が扱えない冬の時代があったが、計算速度の向上により大規模な状態を持つ場合の強化学習が可能となりつつある。

・強化学習は、関数近似法とQ学習を組み合わせる手法の登場によりブレイクスルーが起き、注目を集めるきっかけとなった。Q学習とは行動価値関数を、行動するごとに更新することにより学習を進める方法のことである。関数近似法とは、当時の値のルックアップによる表検索に代わり、価値関数や方策関数を近似する手法のことである。

【価値関数】

価値を表す関数（価値関数V）としては、状態価値関数と行動価値関数の2種類がある。

ある環境の状態の価値にのみ注目する場合には状態価値関数、状態と価値（エージェントがとった行動）を組み合わせた価値に注目す場合は行動価値関数を定義する。近年よく使われているのは行動価値関数である。

【方策関数】

方策関数Π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探るのかの確率を与える関数のことである。方策関数が出した結果に基づいてエージェントは環境中で行動をとる。価値関数が出す価値が最大化するように方策関数が決定される。

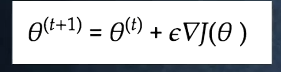
方策関数はΠ(s,a)であらわす。Π(s,a)：VやＱをもとにどういう行動をとるかを表す。例えば経験を生かすのかチャレンジするかなど。その瞬間その瞬間の行動をどうするかを決定する。

状態関数はＶΠ（s）であらわし、状態+行動関数（行動価値関数）はQΠ(s,a)であらわす。

ゴールまで今の方策を続けたときの報酬の予測値が得られる。すなわちやり続けたら最終的に結果はどうなるかということを計算する。。

方策勾配法について

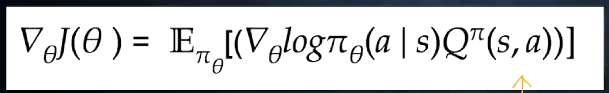
方策間数の学習方法に方策反復法がある。方策をモデル化して最適化する手法が方策勾配法である。以下に方策勾配法の理論式を示す。

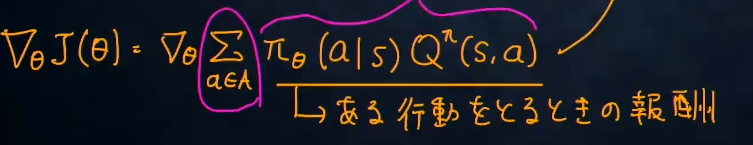


Θが方策関数の重みパラメータである。ｔは時刻、εは学習率である。ニューラルネットワークの重みの更新式に似ている。Jとは方策の良さを表し、定義しなければならないＪはニューラルネットワークでは誤差関数に相当し期待収益を表す。ニューラルネットワークでは誤差関数を小さくするように学習を進めたが、強化学習ではＪが大きくなるように学習を進める。

方策関数Q(s,a|Θ)の定義について

定義方法には、平均法週報と割引報酬和がある。上式でε∇Ｊ(Θ)の部分を計算する。定義に対応して行動価値関数:Q(s,a)の定義をおこない以下の方策勾配定理が成り立つ。





Π（）は方策関数を示す。

Ｑ（）は価値関数を示す。

Π（）Ｑ（）である行動をとるときの報酬を表す。すべての行動パターンに対して報酬を足し合わせる。

【Alpha Go】

強化学習に注目を集めたモデルの一つ。Alpha Go LeeとAlpha Go zeroの2種類が存在する。

【Alpha Go Lee】

方価値関数に関するネットワーク(Valueネット)、方策関数に関するネットワーク(PolicyNET)の2つネットワークで構成される。ともに畳み込みニューラルネットワークである。

PolicyNET

19×19の2次元データサイズ（この入力サイズは囲碁の盤面に対応している）、48チャンネルのデータを扱う。例えば、石（白石、黒石、空白の3チャンネル）やオール1、着手履歴（8手前までに石が打たれた場所）、呼吸点、取る石、取られる石の数等がチャンネルに含まれる情報である。CNNなので畳み込み層で畳み込んで、プーリングする。出力としてはSoftMAX関数を使いどこに打てばよいかという確率を算出する。

ValueNET

入力は19×19×49のサイズ（PolicyNETと比べ手番のチャンネルが増える）であり、CNNなので畳み込み層で畳み込みを行う。出力として得たいのはこのままの方策で勝てそうかどうかである。1～-1の値を出力する（1が勝ち、-1が負けに対応）。全結合層を使い出力を1次元にする。

Alpha Goの学習方法

1. 最初に教師あり学習を行う

2. 強化学習によるPolicyNETの学習を行う

3. 強化学習によるValueNETの学習を行う

PolicyNETの強化学習

現在の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う。Policypoolとは、PollcyNetの強化学習の過程を500イタレーションごとに記憶し保存しておいたもの。現状のPollcyNet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、対局のバリエーションに幅を持たせて、過学習を防ごうとするためである。この学習をミニバッチサイズ128で行う。

ValueNETの学習

PollcyNet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する。教師データの作成手順は

1. 教師あり学習で作成したPollcyNetでN手まで打つ
2. N+1手目の手をランダムに選択し、その手で進めた局をS(N+1)とする
3. S(N+1)から強化学習で作成したPollcyNetで終局までうち、その勝敗報酬をRとする。

である。

S(N+1)とRを教師データ対とし、損失関数を平均2乗誤差を用いて、回帰問題として学習する。この学習をミニバッチサイズ32で5000万回行う。N手までとN+1手からのPollcyNetを別々にしてある理由は、過学習を防ぐためであると原著論文では結論づけられている。

RollOutPolicy

ニューラルネットワークではなく線形の方策関数である。探索中に拘束に着手確率を出すために使用される。畳み込みニューラルネットワークなので計算量が多い。方策のPolicyNETで、盤面を見て1つの石をどこに置くかという計算は3msかかる。これを学習のために何千回とやると膨大な学習時間となる。しかしRollOutPolicyは3usで計算が可能である。教師あり学習の時にRollOutPolicyを使う。人間同士の対戦を学習させる際、KGS GO Server(ネット囲碁対局サイト)の棋譜データから3000万局分の教師を用意し、教師と同じ着手を予測できるように学習を行う。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Pollcynetは57％ほどの精度である。同じことをRollOutPolicyにも学習させるが24パーセントにとどまる。

モンテカルロ木探索

強化学習の学習手法の1つであり、価値関数の更新方法である。コンピュータ囲碁ソフトでは現在最も有効とされている探索法である。ほかのボードゲームではminmax探索法やsin波形のαβ探索を使うことが多いが、盤面の価値や勝率予測値が必要となる。しかし囲碁では、盤面の価値や勝率予測値を足すのが困難であるとされてきた。そこで、盤面評価値に頼らず末端評価価値すなわち勝敗のみを使って探索を行うことが出来ないかという発想でモンテカルロ木探索は発明された。囲碁の場合、ほかのボードゲームとは違い最大手数はマス目数でほぼ限定されるため、末端局面に到達しやすい。具体的には、元局面から末端局面までPlayoutと呼ばれるランダムシミュレーションを多数回行い、その勝敗を集計して着手の優劣を決定する。また、該当手のシミュレーション回数が一定数を超えたら、その手の局面をシミュレーション開始局面とするように探索木を成長させる。この探索木の成長を行うというのがモンテカルロ木探索の特徴である。モンテカルロ木探索はこの木の成長を行うことで、一定条件化において探索結果は最善手を返すということが理論的に証明されている。

AlphaGo LeeとAlphaGo zeroの違い（AlphaGo zeroの特徴）

・教師あり学習を一切行わず、強化学習のみでモデルを作成する

・特徴入力からヒューリスティックな要素（人間が決めた特徴量選択）を排除し、石の配置のみを特徴の入力として使用している

・PollcyNetとvalueネットを1つのネットワークにしている

・ResidualNetを導入している。

・モンテカルロ木探索からRollOutシミュレーションをなくしている。

ネットワークは1つになったが　方策関数と価値関数の2つを出力する。出力を2つに分けるネットワークがResidualNetworkである。

ResidualNetworkは層のショートカット構造を持つネットワークであり。層が深い部分を回避し、勾配消失や勾配爆発を抑える効果を狙っている。ResidualNetworkを使うことにより、100層を超えるネットワークでも安定した学習が可能となる。基本構造は、

Convolution→BatchNorm→ReLU→Convolution→BatchNorm→Add→ReLUのブロックを1単位として積み重ねる形（39個）となる。ResidualNetwork自体深いネットワークだが、ショートカット構造で、勾配消失を回避している。また、ResidualNetworkを使うことにより、総数の違うネットワークのアンサンブル効果が得られるという説もある。

ResidualNetworkの派生形として、BottleneckやPreActivationがある。Network構造の工夫としてはWideResNetやPyramidNetがある。

【軽量化・高速化技術】

軽量化・高速化技術のアプローチとしては以下の2種類ある。

1. モデルをいかに早く学習させるかというアプローチ（モデル・データの並列化やGPUの利用）
2. 性能が悪いPC上でもモデルが動くようにしようとするアプローチ（量子化、蒸留、ブルーニング）

【分散深層学習】

コンピュータは18か月で2倍の計算能力になるのに対して、機械学習モデルは毎年10倍の計算量が必要になっている。このため、深層学習は多くのデータを使用したり、パラメータ調整のために多くの時間を使用したりするため、高速な計算が求められる。高速な計算を実現するためには、複数の計算資源（ワーカー）を使用し、並列的にニューラルネットを構成することで、効率の良い学習を行う必要がある。また、データ並列化、モデル並列化、GPUによる高速化技術は不可欠である。

【データ並列化】

親モデルを各ワーカーに子モデルとしてコピーし、データを分割後、各ワーカーごとに計算させる。コンピュータ内の演算機を増やす場合もある。スマートフォンをワーカーとして使う研究例もある。データ並列化は各モデルのパラメータの合わせ方で、同期型、非同期型が決まる。

同期型

各ワーカーの計算が終わるのを待ち、全ワーカーの勾配が出たところで勾配の平均を計算し、親モデルのパラメータを更新する。パラメータ更新後のモデルを再び子モデルとしてコピーする。これを繰り返す。

非同期型

各ワーカーはお互いの計算を待たずに、各子モデルごとに更新を行う。学習が終わった子モデルはパラメータサーバにpushされる。新たに学習を始めるときは、パラメータサーバからPopしたモデル（サーバの中で最も最近保存されたモデル）に対して学習していく。

同期型と非同期型の比較

・処理スピードは、お互いのワーカーの計算を待たない非同期型の方が速い。

・非同期型は最新のモデルパラメータを利用できないので学習が不安定になりやすい(StaleGradientPtoblem)。

・現在は同期型の方が精度が良いことが多いので、主流となっている。

・分散PCが完全に管理できる場合は同期型、そうでない場合は非同期型というような使い分けがなされる。

【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。すべてのデータで学習が終わった後で、一つのモデルに復元する。大きなニューラルネットワークでは、各層（畳み込み層や、出力層等）の複数の塊、ブロックごとに分割し、各々のブロックで各ワーカーごとに学習させることが可能である。枝分かれ構造のニューラルネットワークも分割可能でありこちらの方が主流である。モデル並列化ではコンピュータが1台で、その中の何台かのPGUで分割することが多い。各々のワーカーの計算結果を収集するときにPCが別々だと通信速度が遅く大変であることが理由である。モデルが大きいときはモデル並列化を、データが大きいときはデータ並列化をすると効果的である。モデル並列化を行うとき、モデルが大きいほど、パラメータ数が多いほど高速化の効果も向上する。小さいモデルに対し並列化しても、各PCで計算した結果を収集するデータ通信の遅延もあってあまり効果が見られない。

【Large Scale Distributed Deep Networks】

・Google社 が2016年に出したモデルであり、Tensorflowの前身といわれている。

・並列コンピューティングを用いることで大規模なネットワークを高速に学習させる仕組みを提案した。

・主にモデル並列をデータ並列（非同期型）の提案をしている。

【GPUによる高速化】

コンピュータの計算機は大きく3つ存在する。

1. GPGPU（General-purpose on GPU）：もともとの使用目的であるグラフィック以外の用途で使用されるGPUの総称である。
2. CPU：PCに1つはついており、高性能なコアが小数存在する。いろいろな種類の演算を処理可能である。また、複雑で連続的な処理を得意とする
3. GPU：比較的低性能なコアが多数存在し、簡単な並列処理を得意とする。ニューラルネットワークの学習では単純な行列計算が多いので、GPUによる高速化が効果的である。

【GPGPUの開発環境】

GPGPUの開発環境には大きくCUDAとOpenCLの2種類が存在する。

・CUDA

GPU上で並列コンピューティングを行うためのプラットフォームであり、NVIDIA社が開発しているGPUのみで使用可能である。ディープラーニング用に提供されているので使いやすいく、機械学習分野では現在CUDAが主流である。

・OpenCL

オープンな並列コンピューティングのプラットフォームであり、NVIDIA社以外の会社(Intel,AMD,ARMなど)のGPUからでも使用可能である。ディープラーニング用の計算に特化しているわけではない。ディープラーニングフレームワーク(Tensorflow,Pytorch)内で実装されているので、使用する際はフレームワークを指定すればよい。

【量子化】

ネットワークが大きくなると大量のパラメータが必要になり、学習や推論に多くのメモリと演算処理が必要になる。パラメータ、重みの数が多いと、多くの情報を保存しておくために多くのメモリを必要とする。通常のパラメータの64bit浮動小数点を32bitなどの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。これを量子化と呼び、一つの数字を表す桁数を落とすことで、保存しておくパラメータの情報量を節約する。

【省メモリ化】

ニューロンの重みを表現する浮動小数点のbit数を少なくし有効桁数を下げることで、ニューロンのメモリサイズを小さくすることが出来、多くのメモリを消費するモデルのメモリ容量を抑えることが出来る。しかし量子化を過度に適用しすぎると、表現できる最高数字と有効桁数が小さくなる。

【量子化の利点と欠点】

利点は計算の高速化、省メモリ化　計算量が少なくて済むということである。

欠点は精度の低下である。

【精度の低下】

量子化により、ニューロンが評価できる小数の有効桁が小さくなり、モデルの表現力が低下する。（32bitの場合は下限　1.175494＾E-38,64bitの場合は下限2.225074E＾-308）

よって、学習した結果ニューロンが1.175494＾E-38未満の値になる場合など重みを表現できなくなる。ただし実際の問題では64bitを32bitにしてもほぼ精度は変わらない。現在では16bitでも十分な場合も多い。

【極端な量子化】

極端な量子化を考察する。表現できる値が0,1の1bit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。学習によってaが0,1を得る必要があるが、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のようになり誤差が大きな式になってしまう。よって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。

【計算の高速化】

倍精度演算(64bit)と単精度演算(32bit)は演算性能が大きく違うため、量子化により精度を落とすことにより、より多くの計算をすることが出来る。深層学習で用いられるNVIDIA社製のGPUの性能は以下のようになる。ここでFROPSは小数の計算を1秒間に何回行えるかを表す。



速度の実験結果について

32bitと6bitで比較すると、0.5sから0.1sまで処理時間を短縮可能である。精度については画像のふちに若干ひずみが見えるか見えないか程度の違いである。

【蒸留】

精度の高いモデルはニューロンの規模が大きなモデルになっている。そのため、推論に多くのメモリと演算処理が必要とされる。規模の大きなモデルの知識を使い軽量なモデル作成を行う。これを蒸留と呼ぶ。

【モデルの簡略化】

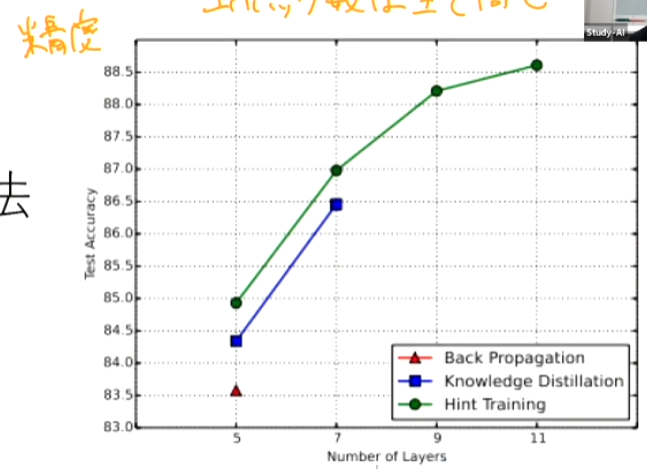
学習済みの精度の高いモデルの知識を軽量なモデルへ継承させる。知識の継承により、軽量でありながら複雑なモデルに匹敵する精度のモデルを得ることが出来る。

【教師モデルと生徒モデル】

蒸留は教師モデルと生徒モデルの二つで構成される教師モデルとは、予測精度の高い、層の多い複雑なモデルやアンサンブルされたモデルである。生徒モデルとは、教師モデルをもとに作られる軽量なモデルである。学習方法については、教師モデルの重みを固定し、生徒モデルの重みを更新していく。誤差は教師モデルと生徒モデルのそれぞれの誤差を使い重みを更新していく。

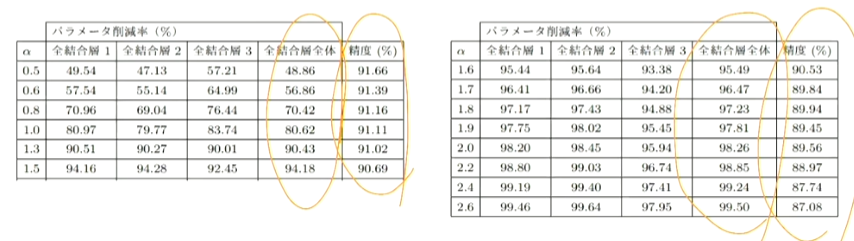
【蒸留の利点】

下のグラフはCifar10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である。図のBack propagationは通常の学習、Knowledge Distillationは先に説明した蒸留手法を適用した場合。Hint Taraingは引用論文で提案された蒸留手法を適用した場合である。図から蒸留によって少ない学習でより精度の良いモデルを作成することが出来ている。



【プルーニング】

ネットワークが大きくなるとパラメータ数も増大するが、すべてのパラメータ及びニューロンの計算が精度に寄与しているわけではない。モデル精度への寄与が少ないニューロンを削除することでモデルの軽量化、高速化が見込まれる。寄与の少ないニューロンの削除を行いモデルの圧縮を行うことで高速に計算を行うことが出来る。ニューロンの削除方法としては、重みが閾値以下の場合、ニューロンを削除し再学習を行う方法がある。下の表はOxford102 category ower datasetをCaffeNetで学習したモデルのプルーニングの閾値による各層と全体のニューロンの削除率と精度をまとめたものである。αは閾値のパラメータ、閾値は各層の標準偏差σとパラメータの積である。閾値を高くするとニューロンは大きく削減できるが精度も減少する。パラメータを半分も削減したにも関わらす精度は90％以上を維持できている。



【モデル軽量化のまとめ】

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：制度への寄与の少ないニューロンをモデルから削除し高速化と省メモリ化を行う技術

【応用技術】

【MobileNet】

MobileNetは画像の認識モデルで軽量化したモデルである。ディープラーニングモデルでは精度は良いが、その分ネットワークが深くなり計算量が増える。計算量が増えると、多くの計算リソースが必要となりコストが高くなる。MobileNetによりディープラーニングモデルの軽量化・高速化・高精度化を実現した。その名の通りモバイルなネットワークとなっている。近年の画像認識タスクに用いられる最新のニューラルネットワークアーキテクチャは、多くのモバイル及び組み込みアプリケーションの実行環境を上回る高い計算資源を必要とされている。一般的な畳み込みレイヤーは入力マップ（H×W×C）、畳み込みカーネルのサイズ（K×K×C）、出力チャンネル数（M）、スライド1でパディングを適用した場合の畳み込み計算の計算量はK×K×C×Mである（1画素あたり）。よって出力画像の計算量はH×W×K×K×C×Mである。一般的な畳み込みレイヤでは計算量が大きいことが課題である。MobileNetではDepthwise ConvolutionとPointwize Convolutionの二つの新たな畳み込み演算方法を使って学習が行われる。

【Depthwise Convolution】

Depthwise Convolutionでは、入力マップのチャンネルごとに畳み込みを実施する。出力マップとそれらを結合する(入力マップのチャンネル数と同じになる)。これまでの畳み込み演算と演算方法は同じであるが、フィルター数は1個で固定する（先ほどはM個）。また、3チャンネルではなく1チャンネルごとにしか畳み込み演算しない。このようにすることで、出力画像の計算量はH×W×C×K×KとなりM次元分計算量が削減される。

【Pointwize Convolution】

Pointwize Convolution は1×1コンボリューションとも呼ばれる。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（チャンネル数）はフィルタ数分だけ作成可能である（任意のサイズが指定可能である）。Depthwise Convolutionとは逆の変数削減である。このときの計算量はH×W×C×MとなりK×K分の計算量を削減可能である。

【MobileNetのアーキテクチャ確認問題（太字部分が回答）】

Depthwise Separable Convolution問手法を用いて計算量を削減している。通常の畳み込みが空間方向とチャンネル方向の計算を同時に行うのに対して、Depthwise Separable ConvolutionではそれらDepthwise Convolution とPointwize Convolutionと呼ばれる演算によって個別に行う。

Depthwise Convolutionはチャンネルごとに空間方向へ畳み込み。すなわち、チャンネルごとにK×K×1のサイズのフィルターをそれぞれ用いて計算を行うがその計算量は**K×K×H×W×C**となる。

次にDepthwise Convolutionの出力をPointwize Convolutionによってチャンネル方向に畳み込むすなわち、出力チャンネルごとに1×1×Mサイズのフィルターをそれぞれ用いて計算を行うため。その計算量は**C×M×H×W**となる。

【DenseNet】

DenseNetは画像認識のためのネットワークである。Denae Convolutional Network （DenseNet）は畳み込みニューラルネットワーク（CNN）アーキテクチャの一種である。ニューラルネットワークでは層が深くなるにつれて学習が難しくなるという問題があった。この問題に対しResidual Network（ResNet）などのCNNアーキテクチャでは前方の層から後方の層へアイデンティティ接続を介してパスを作ることで問題に対処した。DenseNetもそのようなアーキテクチャの一つである。DenseNet ではDenseBlockと呼ばれるモジュールを用いて問題に対処している。DenseNetで特徴的なのはDenseBlockであり出力層に前の入力層を足し合わせる構造である。層間の情報の伝達を最大にするためにすべての同じ特徴量サイズの層を結合する。チャンネルが前層分増えるような仕組みとなっている。畳み込みのチャンネル数分、出力のチャンネルが増えていく。Transition Layerと呼ばれる層で特徴マップのサイズを変更し元の入力サイズに戻るようにダウンサンプリングが行われる。DenseNet はDenseBlockとTransition Layerを交互に繰り返す構造になっている。

DenseNetとResNetの違い

DenseBlockでは前方の各層からの出力全部が後方の層への入力として用いられるが、ResidualBlockでは前1層の入力のみ工法の層へ入力している。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。ここで、DenseBlock内の各ブロックごとにk個づつ特徴マップのチャンネル数が増加していくとき、kを成長率と呼ぶ。

【BatchNorm】

レイヤー間を流れるデータの分布を、ミニバッチ単位で平均が0、分散が1になるように正規化する。これをBatch Normalization（BatchNorm）と呼ぶ。BatchNormはニューラルネットワークにおいて学習時間の短縮や初期値への依存低減、過学習の抑制などの効果がある。問題点としてはBatchSizeが小さい条件下では、学習が収束しないことがあり、代わりにLayer Normalizationなどの正規化手法が使われることが多い。ミニバッチの大きさはデバイス、PCによって変わるので、BatchNormによる効果を確認するのが難しく、使い勝手が悪い。

Batch Norm以外の正規化としてはそれぞれのサンプルのすべてのピクセルが同一分布に従うように正規化するLayer Normや、さらにチャンネルも同一分布に従うように正規化するInstance Normなどがある。

H×W×CのサンプルがN個あった場合に、N個の同一チャンネルが正規化の単位である。

RGBの3チャンネルのサンプルがN個の場合には、それぞれのチャンネルの平均と分散を求め正規化を実施し、チャンネルごとに正規化された特徴マップを出力する。

このとき、ミニバッチのサイズが大きく取れない場合には、効果が薄くなってしまう。

【Layer Norm】

N個のサンプルのうち1つに注目し、H×W×Cのすべてのピクセルが正規化の単位となる。RGBの3チャンネルのサンプルがN個の場合は、あるサンプルを取り出し、すべてのチャンネルの平均と分散を求め、正規化を実施し、特徴マップごとに正規化された特徴マップを出力する。ミニバッチの数に依存しないので、上記のBatch Norm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、入力データのスケール操作や、重み行列のスケールやシフト操作を施しても出力が変わらないことが知られている。

【Instance Norm】

各サンプルの各チャンネルごとに正規化する。コントラストの正規化により、画像のスタイル転送やテクスチャ合成タスクなどで利用される。

【Wavenet】

Aaron van den Oord et. al.らにより提案されたモデルである。AlphaGoのプログラムを開発しており、2014年にgoogleに買収されている。生の音声波形を生成する深層学習モデルであり、Pixel CNN(高解像度の画像を精密に生成できる手法)を音声に応用したものとなっている。Wavenetでは、時系列データに対して畳み込みを適用する。Dilated convolution 層が深くなるにつれて畳み込むリンクを離す（飛び飛びに畳み込みを行う）構造を採用しており、受容野を簡単に増やすことが出来るという利点がある。

【確認問題（太字が回答）】

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者を提案したことがWavenetの大きな貢献の一つである。提案された新しいConvolution型アーキテクチャは**Dilated causal convolution**と呼ばれ、結合確率を効率的に学習できるようになっている。

**Dilated causal convolution**を用いた際の大きな利点は、単純なConvolution layerと比べて、**パラメータ数に対する受容野が広い**ことである。

【物体検知とSS（セマンテッィクセグメンテーション）】

【物体認識タスク】

物体検知タスクは大きく分けて以下の4種類である。

・Classification（分類） ：出力が（入力画像に対し単一または複数の）クラスレベル

・Object Detection（物体検知）：出力がBounding Box[box/BB]

・Semantic Segmentation（意味領域分割）：出力が(各ピクセルに対し単一の)クラスラベル

・Instance Segmentation（個体領域分割）：出力が（各ピクセルに対し単一の）クラスラベル

タスクの難易度はClassification（分類）→Object Detection（物体検知）→Semantic Segmentation（意味領域分割）→Instance Segmentation（個体領域分割）の順で難しくなる。Classification（分類）では物体の位置がどこにあるかという認識は不要である。また、Instance Segmentation（個体領域分割）のみインスタンス（物体個々）の区別まで行う。

【データセット】

データセットではクラス数と訓練・検証データ数とBox/画像の3項目が注目すべき特徴である。以下に代表できなデータセットとその特徴を挙げる。

・ＶＯＣ12：Instance Annotationあり。クラス数は20　訓練・検証データ数は11540　Box/画像は2.4

・ＩＬＳＶＲＣ１７：クラス数は200　訓練・検証データ数は476668　 Box/画像は1.1

・ＭＳ ＣＯＣＯ１８：Instance Annotationあり。クラス数は80　訓練・検証データ数は123287　Box/画像は7.3

・ＯＩＣＯＤ１８：Instance Annotationあり。クラス数は500　訓練・検証データ数は1743042　Box/画像は7.0

特にBox/画像（1画像あたりにBounding Boxがいくつあるか）については、小さいほどアイコン的な映り、日常感とはかけ離れやすい画像を示しており、大きいほど部分的な重なり等も見られる日常生活のコンテキストに近い画像となっている。目的に応じたBox/画像の選択をする必要がある。クラス数が大きいほど、同じものなのに違うラベルが付与されている可能性があり有用ではない場合がある。

【評価指標】

まず分類問題における評価指標である混同行列を復習する

真値：Positive, Negative　と　予測：Positive, Negative　に対して、以下の評価指標を定義する。

・Precision = True Positive/(True Positive + False Positive)

・Recall = True Positive/(True Positive + False Negative)

判定閾値を変えることで、Precision、Recallも変化する。物体検知では、信頼度とラベルBounding Boxのセットで与えられる。閾値を変えると、閾値判定を満足する候補の総数が変化する。物体検出においてはクラスラベルだけではなく、物体位置の予測精度も評価する必要がある。その方法としてIoUが提案されている。混同行列の要素であらわすと、

IoU=TP/(TP+FP+FN)となる。よくあるIoUの誤りとしては正解面積/全予測面積　という解釈をしがちだがこの直感的な解釈は誤りである。

物体検知では、信頼度だけでなくIoUに対しても閾値を設定する。二つの閾値判定を満足しているものはTrue Positive、どちらか一方でも閾値判定を満足しない場合False Positiveと判断する。同じ物体に対して複数のBounding Boxが出てきた場合最も信頼度が高いものを選択する。本来検出するべきもので検出できなかった場合はFalse Negativeと判断する。各クラスに関してPrecision、Recallを算出することでクラスごとに評価する。信頼度の閾値を変えながらPrecision、Recallを計算しPR曲線(Precision-Recall curve)を求める。PR曲線から、Average Precision（AP）を求める。Average PrecisionはPR曲線の下側面積に相当する。クラスラベルを固定していたので、クラスラベルごとにAPが求まる。各APをクラス数で平均したものがmean Average Precision(mAP)である。

MS COCOで導入された指標

一般的には信頼度の閾値を変え、IoUの閾値は0.5で固定してmAPを求めていたが、IoUの閾値を0.5～0.95まで0.05刻みでAPとｍＡＰを計算し算術平均を計算するした指標がmAPcocoである。MS COCOの評価ではmAPcocoが用いられた。これにより物体検出結果の位置まで考慮した評価が可能となる。

物体検知では、応用上精度に加え速度も問題となる。検知速度の指標に関して、FPS(Flames per Second)が用いられる。FPS は1秒間あたりに物体検出処理可能な画像(flame)数である。1画像あたり何秒推定に時間がかったかという観点で評価されることもある。

【物体検知】

物体検知の歴史的な技術発展についてまとめる。

2012年ごろ、AlexNetの登場を皮切りに時代はＳＩＦＴからＤＣＮＮへモデルが改良される。2013年以降、代表的なネットワークとして、VGGNetやResNET、GooLeNetなどが登場する。これと同時ＲＣＮＮやYOROなどの物体検知モデルが登場する。物体検知モデルは、2段階検出器と1段階検出器に大きく分類される。2段階検出器は候補領域の検出とクラス推定を別々に行うモデルであり、RCNNやSPPNet、FPNなどが該当する。一方1段階分類器は候補領域の検出とクラス推定を同時に行うモデルであり、YOLOやDetectorNet、SSDなどが該当する。2段階検出器の方が1段階検出器に比べ精度が高くなる傾向がある。しかし計算量は1段階検出器の方が2段階検出器に比べ小さく推定が速い傾向がある。

【SSD(Single Shot Detector)】

1段階検出器の物体検知モデルであるSSD(Single Shot Detector)について説明する。まず、Bounding Boxを適当な位置に初期設定する。これをDefault boxと呼ぶ。SSDの学習によって、Default boxが修正されBounding Box、ラベル付け、信頼度が決定する。畳み込み層と全結合層が合計16層あるVGG16のモデルがベースとなっている。VGG16と異なるのは全結合層2層を畳み込み層に変更し、VGG16の第10層目に相当する畳み込み層を特徴マップとして出力している。ほかの層でも異なるサイズから特徴マップを出力している。特徴マップのサイズが大きい、すなわち解像度が大きい場合は小さな物体が、特徴マップのサイズが小さい、すなわち解像度がちいさい場合は大きい物体に対し、Bounding Boxを学習していることに対応する。マップの各特徴慮にｋ個のDefault boxを用意する場合、出力サイズはｋ（クラス数+4）となる。4はＳＳＤにおけるDefault box変化分のオフセット項である。

【Semantic Segmentation】

Semantic Segmentationでは入力画像の各ピクセルに対してラベル付けを行う必要があるが、このとき畳み込みニューラルネットワークで層を重ねるごとに解像度が小さくなることが不都合となる。そのため、元の入力画像の解像度に戻す、アップサンプリングを行う必要がある。ＶＧＧ１６のネットワークにおいても特徴マップが224×224から畳み込みを複数層にわたって行うことで最終的には7×7のサイズまで小さくなっている。受容野を大きくするために深い畳み込み層やプーリングが必要であった。よってプーリング層をむやみに削除した特徴マップのサイズを維持することはできない。そこで、アップサンプリングの手法として、Deconvolution(Transposed convolution)がある。通常の畳み込み層と同様にカーネルサイズとパテッィング、ストライドを指定する。演算手順としては

1. 特徴マップのピクセル間隔をストライド分だけ開ける

2. 特徴量マップの周りに(カーネルサイズ-1)-パッティング数 だけ余白を作る

3. 畳み込み演算を行う

となる。

実際には畳み込み演算の逆演算とはならないので注意が必要である。当然プーリングで失われた情報が復元されるわけではない。プーリングによりローカルな情報すなわち輪郭が失われていくが、低レイヤーのプーリング層を要素ごとに足し算しながらアップサンプリングすることで、ローカルな情報を補完しながらアップサンプリングを行う。例としてU-Netでは畳み込みのEncoderとDeconvolutionのＤecoder構造となっている。Ｄecoderではチャネル方向での結合で、低いレイヤーのEncoderの情報を付加しながらアップサンプリングを行っている。

プーリングしたときの位置情報を保持しておくことで、プーリング後元のサイズにアップサンプリングできるアンプーリングという手法がある。また、プーリング層を使わないで受容野を広げていく方法としてDilated Convolutionがある。

【Transformer】

【seq2seqの復習】

Transformer の要素モデルであるeq2seqを復習する。seq2seqはある時系列データから別の時系列データへ変換するモデルのことである。seq2seqにおいて重要な要素モデルはＲＮＮと言語モデルの2つである。ＲＮＮは系列情報を内部状態に変換することが出来る。文章中の各単語が現れる際の同時確率は、事後確率で分解できる。したがって、事後確率を求めることがＲＮＮの目標になる。言語モデルを再現するようにＲＮＮの重みが学習されていれば、ある時点の次の言語を予測することが出来る。これを繰り返すことで、先頭単語を与えれば文章を生成することが出来る。seq2seqは2つのＲＮＮが結合された構造を持つ。すなわち、入力を受け取るＲＮＮ(Encoder)と出力するＲＮＮ(Decoder)から構成される。Decoder側の構造は言語モデルＲＮＮとほぼ同じだが、隠れ層状態の初期値にEncoder側の内部情報を受け取る。Decoderの出力側に正解を当てれば教師あり学習が実施できる。seq2seqは長い文章に対して、表現力が不足することが課題である。この課題に対し、文章中のどの単語に注目するべきかをモデルに学習させる機構：Ａttentionが発明された。Ａttentionでは、各単語の隠れ状態の荷重平均を算出する。Attentionはキーに対応したクエリにどれだけ類似しているかを算出し、辞書オブジェクトとしている。

【Transformerの概要】

Transformerは2017年6月に登場した。TransformerはＲＮＮを使わない構造であり必要な構造はAttentionのみである。当時のSOTAをはるかに少ない計算量で実現している。EncoderとDecoderの構造を持ち、どちらもAttention機構を含んでいる。文字の位置情報を最初に付加する構造がある。Attentionはターゲットとなる単語に対して、最も近い値をソースから探し、注目（重みづけ）するソース・ターゲット注意機構と、自身の入力からどこに注目すべきか重みづけする自己注意機構の2つある。TransformerのEncoderで使われるのは自己注意機構である。入力をすべて同じにして学習的に注意個所を決めていく。文章の長さに対応したCNNと解釈することもできる。Encoderでは自己注意機構により文脈を考慮して各単語をエンコードする。そして、位置情報を保持したまま順伝播させる。その後、全単語に関するAttentionをまとめて計算する。この計算後の結果を結合する。Decoderでは、各層で2種類の注意機構（ソース・ターゲット注意機構と自己注目機構）を使っている。未来の情報を見ないようにマスクして自己注意機構の処理を行う。Transformerでは、入出力の差分を学習したり、バッチノーマライゼーションを行ったりすることで、学習効率を高めている。Transformer は言語構造を理解できるとされており、実際、Attention層を可視化すると言語構造をとらえていることが多い。