実装演習レポート（深層学習Day3&Day4）

owadatk edited this page now · 4 revisions

深層学習Day3

再帰型NNの概念

要点のまとめ

- RNNとは時系列データに対応可能なNNである
- RNNの全体像
  - 数学的記述

$$u^t = W_{(in)}x^t + W\,z^{t-1} + b$$
$$z^t = f\left(W_{(in)}x^t + W\,z^{t-1} + b\right)$$
$$v^t = W_{(out)}z^t + c$$
$$y^t = g\left(W_{(out)}z^t + c\right)$$

- コード

```
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))
```

- RNNの特徴：再帰構造

- BPTT（Back Propagation Through Time）誤差逆伝播の一種
  - 数学的記述

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

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

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

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

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

  - コード

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

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

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

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

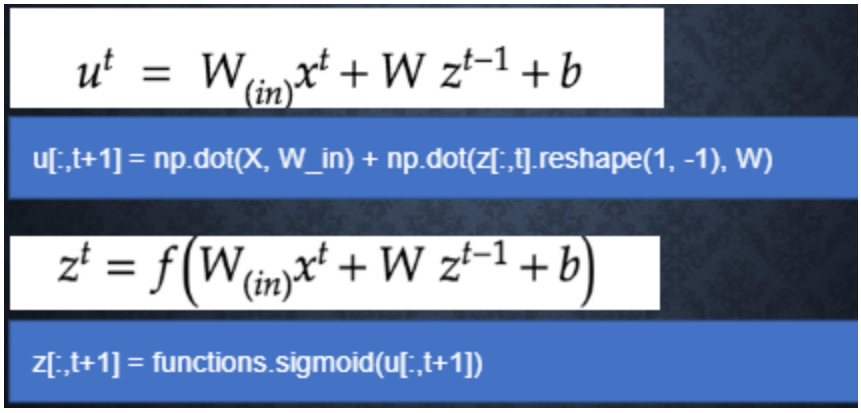
$$\frac{\partial E}{\partial W} = \frac{\partial E}{\partial u^t}\left[\frac{\partial u^t}{\partial W}\right]^T = \delta^t\left[z^{t-1}\right]^T$$

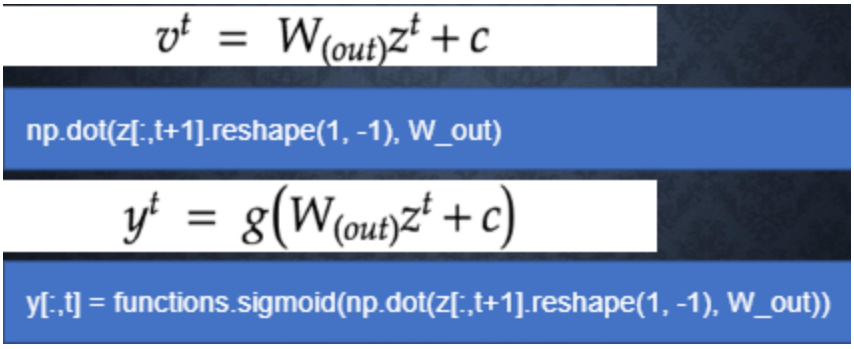
```
np.dot(z[:,t].reshape(-1,1), delta[:,t].reshape(1,-1))
```

- 連鎖律が長いのでdeltaを導入して簡略化。Tは転置ではなくて、時間を表す

$$u^t = W_{(in)}x^t + W\,z^{t-1} + b$$
$$z^t = f\left(W_{(in)}x^t + W\,z^{t-1} + b\right)$$
$$v^t = W_{(out)}z^t + c$$
$$y^t = g\left(W_{(out)}z^t + c\right)$$

- 数学的記述2
- コード2





- 数学的記述3

$$\frac{\partial E}{\partial u^t} = \frac{\partial E}{\partial v^t}\frac{\partial v^t}{\partial u^t} = \frac{\partial E}{\partial v^t}\frac{\partial \left\{W_{(out)} f\left(u^t\right) + c\right\}}{\partial u^t} = f'\left(u^t\right) W_{(out)}^T \delta^{out,t} = \delta^t$$

$$\delta^{t-1} = \frac{\partial E}{\partial u^{t-1}} = \frac{\partial E}{\partial u^t}\frac{\partial u^t}{\partial u^{t-1}} = \delta^t \left\{\frac{\partial u^t}{\partial z^{t-1}}\frac{\partial z^{t-1}}{\partial u^{t-1}}\right\} = \delta^t \left\{W f'\left(u^{t-1}\right)\right\}$$

$$\delta^{t-z-1} = \delta^{t-z}\left\{W f'\left(u^{t-z-1}\right)\right\}$$

```
delta[:,t] = (np.dot(delta[:,t+1].T, W.T) + np.dot(delta_out[:,t].T, W_out.T)) *
functions.d_sigmoid(u[:,t+1])
```

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

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

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

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

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

- 数学的記述4

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

- コード

```
W_in -= learning_rate * W_in_grad
```

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

```
W_out -= learning_rate * W_out_grad
```

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

```
W -= learning_rate * W_grad
```

- BPTTの全体像

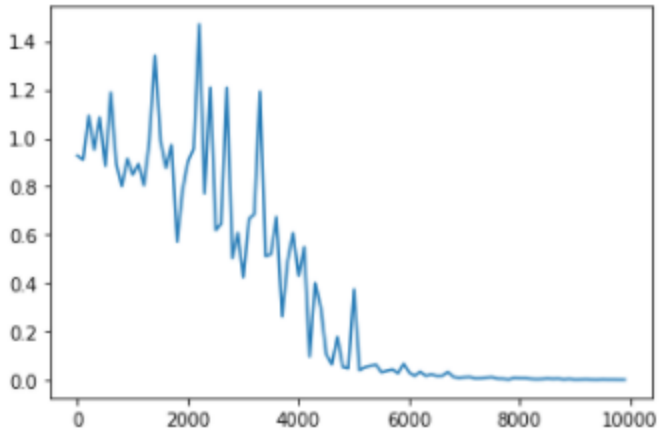
$$
\begin{aligned}
E^t &= loss\left(y^t, d^t\right) \\
&= loss\left(g\left(W_{(out)}z^t + c\right), d^t\right) \\
&= loss\left(g\left(W_{(out)}f\left(W_{(in)}x^t + W\ z^{t-1} + b\right) + c\right), d^t\right)
\end{aligned}
$$

$$
\begin{aligned}
&W_{(in)}x^t + W\ z^{t-1} + b \\
&W_{(in)}x^t + W\ f\left(u^{t-1}\right) + b \\
&W_{(in)}x^t + W\ f\left(W_{(in)}x^{t-1} + W\ z^{t-2} + b\right) + b
\end{aligned}
$$

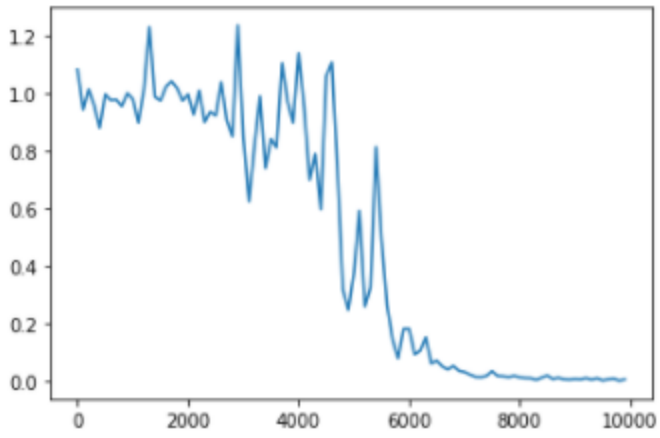
実装演習

- バイナリ加算

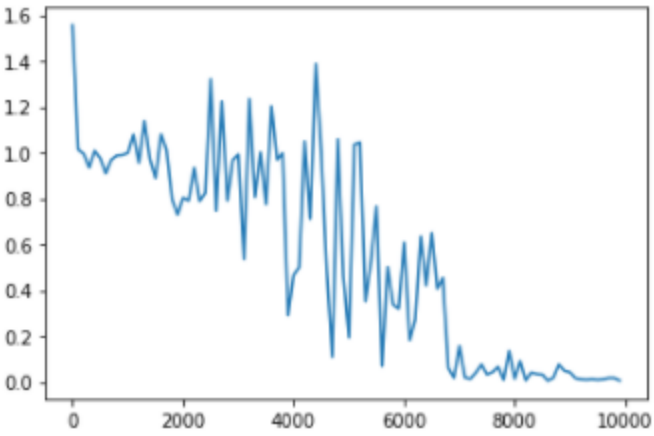
```
------------
iters:9900
Loss:0.0022435686175360675
Pred:[0 1 1 0 1 0 0 0]
True:[0 1 1 0 1 0 0 0]
11 + 93 = 104
------------
```



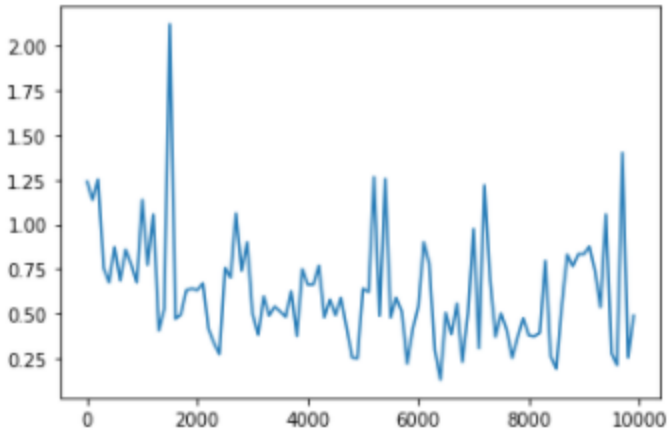
- Xavierの初期値



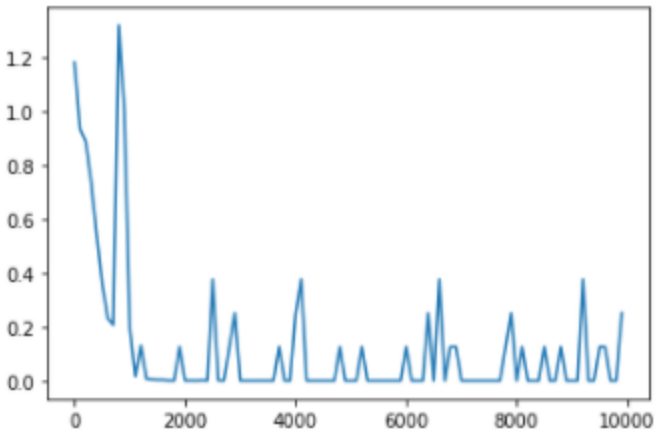
- Heの初期値



- Heの初期値＋ReLU関数



- Heの初期値＋tanh関数



確認テスト等考察

- 問「RNNのネットワークには大きく分けて3つの重みがある。ひと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、ひと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。残りひとつの重みについて説明せよ」
  - 前の中間層からの入力に対する重み
- 演習チャレンジ「構文木」の問題
  - 単語同士の特徴量を合体させるとき、それぞれの特徴を維持するにはどうするか？を考える。
- RNNの図の中のy(1)を数式で表す（中間層の出力はシグモイド関数g(x)を用いる）
  - y(1) = g(W(out)*S(1) + C), s(1) = f(w(in)*x(1)* + *w*s(0) +b)
- コード演習問題：中間層から中間層への出力（チャレンジ問題）
  - 上記の「数学的記述3」を参照

LSTM

要点のまとめ

- 時系列を遡るほど勾配が消失していく問題への解決策として、構造自体を変えることで対応したのがLSTM
- 勾配消失問題と勾配爆発問題
- LSTMはRNNの一種。CEC (Constant Error Carousel) を取り囲むように様々な要素が配置されている

- CEC：記憶する部分。学習機能を分離してCECには記憶する機能だけをもたせている
- 入力ゲート：CECに覚え方を渡す
- 出力ゲート：CECに記憶を使う方法を渡す
- 忘却ゲート：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
- 覗き穴結合：CEC自身の値に、重み行列を介して伝播可能にした構造⇒あまり性能の改善はみられなかった

実装演習

（LSTMとGRUをまとめてGRUの項参照）

確認テスト等考察

- 演習チャレンジ：勾配爆発を防ぐ方法⇒**勾配クリッピング**（勾配のノルムが閾値を超えたら勾配のノルムを閾値に正規化する）
- プログラムは基本文法は覚える必要あるが、他は実装しながら必要に応じて調べればよい
- E資格で知らないものが出てきたら周りの情報から推測するしかない
- 問「…なくなても影響を及ぼさない単語。どのゲートが作用するか」⇒忘却ゲート
- 演習チャレンジ：CECに覚えさせる情報　⇒モデルを見ながら考える（図を見比べてコードを考える問題は頻出）

GRU

要点のまとめ

- LSTMの改良版。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。
- パラメータを大幅に削減。計算負荷が低いというメリット。
- 内積演算ユニット、更新ゲート、リセットゲート

実装演習

- %tensorflow_version 1.x　追加
- ライブラリに用意されている

```
187         # RNN のセルを定義する。RNN Cell の他に LSTM のセルや GRU のセルなどが利用できる。
188         cell = tf.nn.rnn_cell.BasicRNNCell(self.hidden_layer_size)
189         outputs, states = tf.nn.static_rnn(cell, input_data, initial_state=initial_state)
190
```

```
purified : 1.3827813e-14
nucleotidase : 1.4714438e-14
scd : 1.390414e-14
adpnp : 1.3065119e-14
spectra : 1.4082694e-14
mhz : 1.5197516e-14
carboxyl : 1.5074157e-14
oxygens : 1.4567606e-14
hydroxyl : 1.40896545e-14
hydrolysis : 1.4591049e-14
phosphates : 1.5364786e-14
conformations : 1.461333e-14
conformation : 1.3528324e-14
annan : 1.4627048e-14
unesco : 1.3770172e-14
sci : 1.37949895e-14
ricyt : 1.450121e-14
averaged : 5.898548e-14
ecological : 1.4461878e-14
statins : 1.4468058e-14
coronary : 1.24290735e-14
myocardial : 1.3683982e-14
infarction : 1.4459039e-14
revascularization : 1.41071855e-14
lipid-lowering : 1.568556e-14
incremental : 1.39636185e-14
ischemia : 1.4152355e-14
miracl : 1.5107144e-14
atorvastatin : 1.3983208e-14
mgdl : 1.5000276e-14
rehospitalization : 1.4568384e-14
worsening : 1.3476893e-14
endpoints : 1.4479183e-14
lipid : 1.5310432e-14
lipoprotein : 1.3668956e-14
ldl : 1.4424414e-14
statin : 1.378644e-14
a--z : 1.3740787e-14
simvastatin : 1.413377e-14
bmi : 3.1720919e-07
covariates : 2.8344898e-06
yhl : 2.2330354e-07
yol : 9.992968e-11
obesity : 1.3501551e-09
evgfp : 6.1829963e-09
unintended : 4.6785074e-09
sizes : 2.569963e-07
obese : 1.9368042e-07
<???> : 2.9195742e-05
Prediction: some of them looks like et
```

- プログラムの実行結果

確認テスト等考察

- 問「LSTMとCECが抱える課題について、それぞれ簡潔に述べよ」

    - LSTMはパラメータが多いために計算量が大きくなるという課題がある。CECは勾配が1で学習能力がないという課題があり、入力ゲート、出力ゲート、忘却ゲートというパラメータが必要となっている。
- 演習チャレンジ：概念図と見比べながら解けばできる
- 問「LSTMとGRUの違いを簡潔に述べよ」
    - LSTMよりGRUの方が計算量が少ない

双方向RNN

要点のまとめ

- 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

実装演習

（確認テスト参照）

確認テスト等考察

- 演習チャレンジ（双方向RNNの順伝播を行うプログラム）⇒加算や乗算をしてしまうと元の情報が失われる。どの軸方向に情報をつなげたらよいか？ axis=0は行方向につなげる（行が長くなる＝1次元のまま）、 axis=1は列方向につなげる（2次元になる）

Seq2Seq

要点のまとめ

- RNNの応用例
- 2つのネットワークを組み合わせた自然言語処理用のネットワーク（機械翻訳に用いられている）
- 入力側のネットワークをEncoder、出力側のネットワークをDecoderと呼ぶ

- 意味の近い表現が似たようなベクトル表現に変換できることがモデル開発の肝（特徴量抽出）
- BERTが優秀な結果を出している（MLM:Masked Language Modelという手法：学習時に一部を隠して前後の文脈から推測させる）
- HRED：Seq2Seqは一問一答しか対応できないところ、文脈に即した応答ができる。HRED=Seq2Seq + Context RNN
- 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ない。短く情報量に乏しい答えをしがち
- VHRED：HREDにVAEの潜在変数の概念を追加したもの。
- VAE：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。Decoderはノイズ付きのデータを受け取るため、より汎用性が高まる。

実装演習

- （該当実習なし）

確認テスト等考察

- Seq2Seqの説明
  - (1) 双方向RNN
  - (2) Seq2Seq
  - (3) 構文木
  - (4) LSTM
- 演習チャレンジ：文の意味を表すベクトルを作ろうとしている。Word embedding matrixというのは、単語とembedding表現との対応表のこと。解答は(1) E.dot(w)
- VAEは、自己符号化器の潜在変数に「確率分布」を導入したもの
- 問「seq2seqとHRED、HREDとVHREDの違いを説明せよ」
  - seq2seqは一文の一問一答に対して処理ができるある時系列データからある時系列データを作り出すネットワーク
  - HREDはseq2seqの機構にそれまでの文脈の意味ベクトルを解釈に加えることで文脈の意味をくみ取った文の変換ができるようにしたもの

    - VHREDはオートエンコーダー、VAEの考え方を取り入れて、短い単語以上の出力ができるように改良を施したモデル

Word2vec

要点のまとめ

- 単語をベクトル表現する手法
- 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。

実装演習

- （該当実習なし）

確認テスト等考察

- （Attentinoの項参照）

Attention Mechanism

要点のまとめ

- 長い文章に対応するため、「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を導入⇒Attention Mechanism

実装演習

- （該当実習なし）

確認テスト等考察

- 問「RNNとword2vec、seq2seqとattentionの違いを説明せよ」
    - RNNは時系列データを処理するのに適したNN。
    - word2vecは単語の分散ベクトルを得る手法。
    - seq2seqはひとつの時系列データから別の時系列データを得るネットワーク。

    - attentionは時系列データの中身の関連性に重みをつける手法。

深層学習Day4

強化学習

- 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
- 方策、行動、報酬、価値。エージェントと環境の状態とのインタラクション
- 探索と利用のトレードオフを調整する必要がある
- 学習させるのは方策関数と行動価値関数
- 関数近似法とQ学習を組み合わせる手法の登場が重要
- 方策関数とは、ある状態でどのような行動をとるのかの確率を与える関数である
- 方策勾配法：方策をモデル化して最適化する手法

AlphaGo

- 強化学習が注目を集めるきっかけとなった
- AlphaGo Lee
    - ValueNet（価値関数に相当）とPolicyNet（方策関数に相当） どちらもCNN
- AlphaGo Zero
    - Residual Network（ショート化とを追加）勾配爆発、消失を抑制

軽量化・高速化技術

- モデル並列：分散深層学習（最も重要な技術）、分岐部分でモデルを分割できる、モデルが大きいときはモデル並列化を、データ大きいときはデータ並列化をするとよい
- データ並列：同期型、非同期型

- GPU：GPGPU開発：CUDA、OpenCL⇒ほとんどCUDAしか使われていない。Tensorflowで使える
- 量子化：通常のパラメータの64bit浮動小数点を32bitなど下位の精度に落とすことでメモリーと演算処理を削減する
- 蒸留：規模の大きなモデルの知識を使い軽量なモデルを作成する（学習済みの精度の高いモデルの知識を軽量なモデルへ承継させる）
- プルーニング：モデルの精度に寄与が少ないニューロンを削減することでモデルの軽量化、高速化が見込まれる（想像以上に消せるし、性能も変わらない）

応用モデル

- MobileNet：画像認識のネットワーク。画像認識は2017年頃に完成。精度を維持しつつ軽量化、高速化の方向へ。MobileNetは畳み込み演算に工夫
- DenseNet：画像認識のネットワーク。Dense Blockに特徴。成長率というハイパーパラメータ
- Layer正規化／Instance正規化：Batch Normalization（ミニバッチ単位で平均0、分散1に正規化）メモリーの制約を受けバッチを大きくできない問題に対応
- WeveNet：音声生成モデル。Delated causal convolution

Transformer

- BERTを理解するにはEncoder-Decoder Model、Transformer（Encoder-Decoder + Attention）の理解が必要
- Encoder-Decoderモデルは固定長ベクトルで表現するため、文章が長くなると表現力が足りなくなる　⇒Attentionの導入
- Attentionは辞書オブジェクトと同等の機能
- Attentionには二種類ある：ソース・ターゲット注意機構と自己注意機構
- 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
- Attention - 全領域に応用され最高精度を叩き出す注意機構の仕組み

物体検知・セグメンテーション

- 物体検知（Bounding Box）、セグメンテーション（意味領域分割、個体領域分割）
- Box/画像比率：画像の中に検出対称が多いほど識別の難易度が高い（現実の問題に近い）
- 目的に応じたBox/画像比率の選択
- 評価指標：Precision/Recall
- 応用上は検出精度に加えて検出速度も問題となる　YOLOv3、RetinaNet-101
- 一段階検出器（DetectorNet, SSD, YOLO, RetinaNet, CornerNet）
  - 候補領域の検出とクラス推定を同時に行う
  - 相対的に精度が低い傾向
  - 相対的に計算量が小さく推論も早い傾向
- 二段階検出器（RCNN, SPPNet, Fast RCNN, Faster RCNN, RFCN, FPN, Mask RCNN）
  - 候補領域の検出とクラス推定を別々に行う
  - 相対的に精度が高い傾向
  - 相対的に計算量が大きく推論も遅い傾向

Clone this wiki locally

```
https://github.com/owadatk/cautious-octo-bassoon.wiki.git
```