FUYUTSUKI 概要

ライブラリ Fuyutsuki は C#用に作成している深層学習・機械学習用ライブラリです。深層学習・人工知能の本質的な理解のため、そして Python で学習を行い、それをデスクトップアプリの開発言語Ｃ＃で読み込む手間を省く、そしてＣ＃のみで強化学習ができるように開発を行いました。また pytorch のように簡単に計算できるように実装を行いました。

名前は現在開発、テスト中の自作サポートプログラム Yoidsuki が冬月型であることにちなみ、Yoiduki に搭載する人工知能や深層学習プログラムの基礎としてのライブラリとして Fuyutsuki(冬月)と名付けました。

なお使用例としてサンプルとしていくつかのプログラムを Fuyutsuki.Test の中に入れております。

ＭＡＴＲＩＸ

使用法

using static Fuyutsuki.Matrix;でインポート

説明

Matrix クラスは２次元行列を計算できる関数と Matrix 構造体を作成するプログラムを入れたクラスです。

Python の ndarray のように行列同士を四則演算、サイズ変更、対角化など様々な計算が int や double 型などと同様に計算ができます。

ひとつ注意点としては、変数型の仕様とことなり、Matrix 同士の計算では、割り算や掛け算がほかの演算よりも優先されることはありません。
よって a+b*c-d を行う場合は、a+(b*c)-d のように記述してください。

またアダマール積(行列成分同士の掛け算)は*もしくは MatMul 関数で、
行列の内積(Dot 算)は MatDot を用いて計算を行ってください。

VARIABLE

使用法

using static Fuyutsuki.Variable;でインポート

説明

Variable クラスは 2 つの Matrix　Weights と Grads を持ちます。

Weights は Variable で行われる計算の結果でメインに使用されるデータです。一方 Grads は誤差逆伝搬や Optimizer など Weights データの修正に用いられます。四則演算の仕様は Matrix と一緒ですが、アダマール積は Mul もしくは＊、Dot 算は Dot を使用して計算が可能です。

主な特徴としては BackProp というリストを持ちます。これは誤差逆伝搬で使用されるもので、計算を積み重ねていくうえで、前処理の情報を保持し、誤差逆伝搬の時にさかのぼって誤差を伝搬します。また誤差逆伝搬の上で、後述する Layer クラスの勾配やバイアスなどのパラメータや出力結果、入力以外の一時処理として生み出される変数は参照されなくなり、メモリから自動的に消去されるため、メモリリークを防げます。
よって誤差逆伝搬を伴う学習でメモリが増えていく場合、どこかでつながりが切れているか確認を行うことができます。

ＦＵＮＣＴＩＯＮ

使用法

Using Fuyutsuki.Function；でインポートしたうえで、コード上で
Function F=new Function();と書き、F.(関数名)で使用

説明

Function クラスは、CrossEntropy や MeansSquareError など loss 計算や形を変える Transpose, 行列を Dropout させる Dropout など、深層学習で用いられる関数群です。

Variable 内の同名関数と同じ動きをします。

LAYER

使用法

using Fuyutsuki.model;でインポート

説明

深層学習で用いられる Layer クラスを基底クラスにしたニューラルネットワークセルが入っています。RNN で用いられる RNN や LSTM,GRU、結合層として用いられる Linear クラスなどです。なお後述する CNN クラスは開発中です。

共通する関数は Forward と Getparam、Cleargrads です。

Forward(call):セルの計算を進める関数

Getparam:パラメータを参照渡しで出力します。

ClearGrads:パラメータの勾配を 0 にします。

MODEL

使用法

using Fuyutsuki.model;でインポート

説明

深層学習で使用される Model が入っています。
TwoLayerNetwork や Encorder,MLP(多層パーセプトロン)です。
Decorder は現在調整中です。

自分で Model を作る際は、Parameter はリスト Params へ、Layer は AddLayer 関数などを用いてリスト Layers に登録してください。Optimizer による Parameter 更新が自動化されます。

基本の関数は Layer とおなじです。各 Model によって多少の違いがあります。

OPTIMIZER

使用法

using Fuyutsuki.model;でインポート

Optimizers.(optimizer 名)で呼び出し

説明

Model パラメータ更新に用いられるレイヤーOptimizer のクラス群です。

Var opt=new Optimizer.Adam();

などで呼び出したうえで、Ｍｏｄｅｌクラスを

Opt.SetUp(model);のようにしてセットしてください。

パラメータ更新を行う際は opt.Update();で行えます。

Optimizer は今後増やしていく予定です。

ＣＮＮ及び VARIABLE4D

現在開発中のコードです。ＣＮＮはＤＮＮやＲＮＮと違い、3~4 次元配列を用いるため、Variable 4 D クラスを用います。現在 CNN は python では numpy を用い、 6 次元配列を用いて処理するところを 4 次元以下で処理できるように、im2col 関数を作成しました。

現在 col2im 関数の実装を行っています。また im2col,col2im ではともに重みを 4,3 次元から 2 次元への変換処理を行っている部分があったため、２次元で重みを保持できないか試行錯誤を行っております。col2im 関数の作成完了に伴って、Pooling 層及び畳み込み層の実装を行い、CNN の実装を完了する予定です。

GRU 対応について

現在 Matrix での計算は CPU のみを用いて行っていますが、NvidiaGPU の入手次第、計算の高速化を行うために並列処理、GPU による計算機能の追加を付与する予定です。