Chainerノート

Chainerを使う上では、ニューラルネットワークの知識必要。そのため、まず、ニューラルネットワーク仕組について見ていく。

ニューラルネットワークとは、機械学習の手法の１つ。人間の脳内にある神経細胞（ニューロン）とその繋がりを、人口ニューロンという数式的なモデルで表現したもの。

簡単なニューラルネットワークの構成図

※２層のニューラルネットワーク



ニューラルネットワークは、「入力層（input）」、「出力層（output）」、「隠れ層（hidden）」からなっている。層と層の間にはニューロン同士の繋がりの強さを示す重み「」がある。

※隠れ層の数は任意である。ディープラーニングとは、隠れ層が多数存在する多層構造のニューラルネットワークのことを指す。

各層は、「ノード」と呼ばれる任意の変数から構成されている。

※ノードは入力データの次元を指す。

例えば、人口調査の入力データが（気温、二酸化炭素量、日光量）だった場合、この１つ１つ（気温、二酸化炭素、日光量）がノード。

入力に関しては、実際の入力とは独立した値を入れることがある。これを「バイアスノード」という。

※ののようなもの。下の図ではバイアスノードをとした。

インプットからアウトプットまでどのようにたどり着つくのか、伝播について見ていく。



上記の図は、隠れ層の第一ノードにインプットから入力が行われる様子をオレンジで示したものである。

入力に対して、のようにという重みがかけられている。

ここで、は、という値を受けとったが、これをそのまま、横流しにするわけではない。

しきい値を受けて、次にどの層に割り振るのかを決めるのである。入力を次の層の出力に変換する関数をとし、その出力値は、で表現できる。この関数を活性化関数という。



また、同様に、先の活性化関数 で振り分けられた値に重みをかける。そして、活性化関数によって次の出力層に振り分けられる。

※一層のニューラルネットワークは、ロジスティック回帰とほぼ同じである。ただし、上記は、２層のニューラルネットワーク。

ニューラルネットワークを学習させる方法を見て行く。

学習させるためには、「誤差逆伝播法」を用いる。

誤差伝播法とは、出力層から伝播させていき、各層の重み（パラメータ）を調整するという手法である。



誤差の計算方法は、【cost function】【loss function】【error function】などが定義される。

僕は、【error function】を用いてきた。

そして、誤差を最小にするように、重みを調整する方法として、【勾配法】を用いてきた。

参考URL：①<https://qiita.com/icoxfog417/items/96ecaff323434c8d677b>

②<https://www.sbbit.jp/article/cont1/33345>

一層のChainerにチャレンジ！！

※データはMNISTでやる！！

イメージ的には、こんな感じになる。



まず、言葉、

メソッド（＝クラス内関数）

オブジェクト（変数や関数をひとまとまりにしたもの）

インスタンス（）

クラス：設計図　　　→ 「それってどんなもの？？」

インスタンス：実際に作ったもの　→ 「クラス（設計図）に書かれた‘それってどんなもの？’を形にしたもの」

オブジェクト：もの（クラス・インスタンスなどをふんわり表現したもの）

簡単なクラスの説明

|  |  |
| --- | --- |
| **class Person(object):** **def \_\_init\_\_(self, name):** **self.name = name** **def greet(self):** **print ("my name is %s " % self.name)****実行結果**  |  | | --- | | **rasu = Person("rasukaru")** **print(rasu.name) → rasukaru** **rasu.greet() → my name is rasukaru** |   **※ここではインスタンス変数を‘name’と定義。**  **\_\_init\_\_は、‘コンストラクタ’** |

Read this URL↓↓

<http://docs.chainer.org/en/stable/tutorial/basic.html#optimizer>

<http://i101330.hatenablog.com/entry/2015/12/13/012736> （chainerチュートリアル）