機械学習入門

Neural Network/Deep Learning

Neural Network(復習)

入力層、中間層、出力層で構成される。

前回のモデルをまとめると

1. 入力データが入力層に入る。
2. 中間層内部では全結合層(全ての変数の重み付き和を取る層)を経て(Affine変換)、活性化関数を通る。
3. 2を繰り返す。
4. 出力層で出力関数(前回はシグモイド関数を用いて、２値の出力を得た)を通り、出力される。

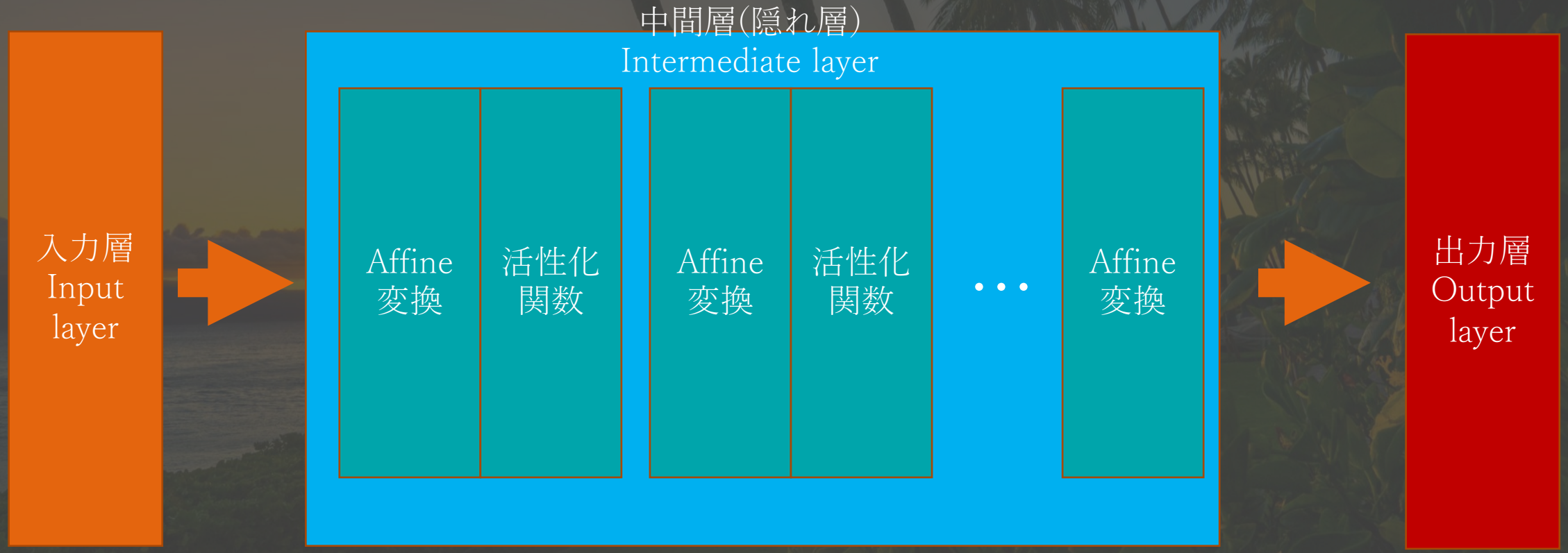
のように書ける。

Neural Networkとは(復習)

一般的なMLP(NN)の構造

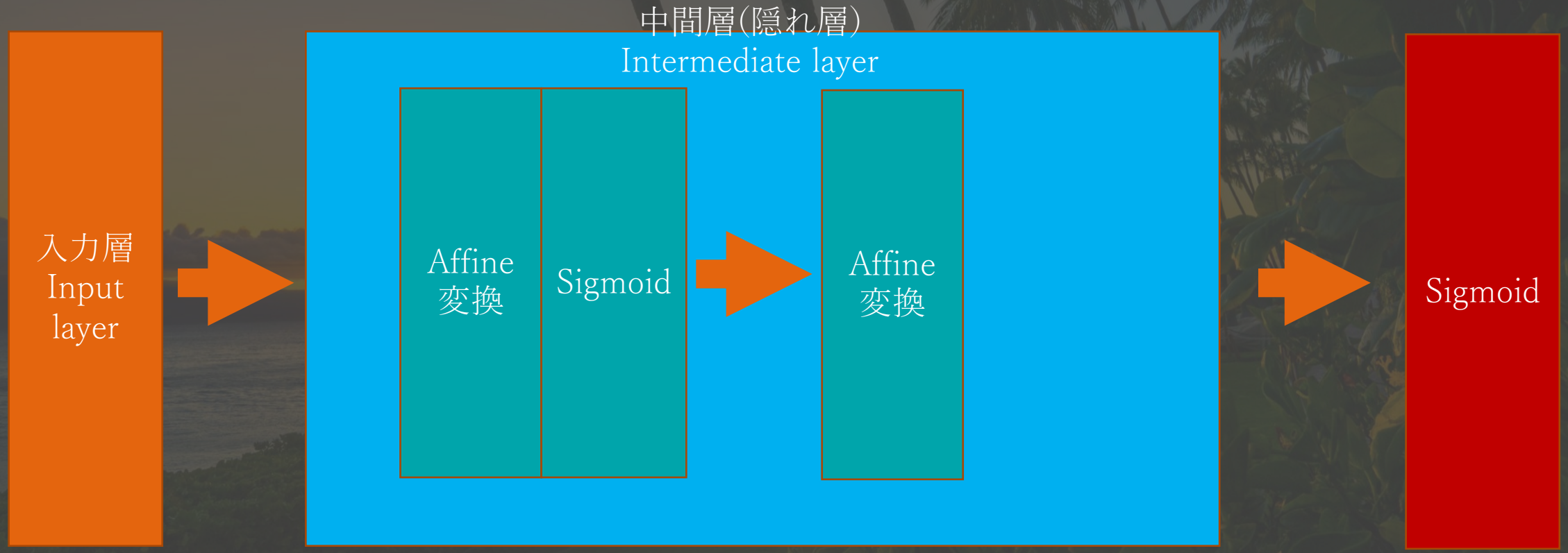
Neural Networkとは(復習)

もう少し詳しく書く。



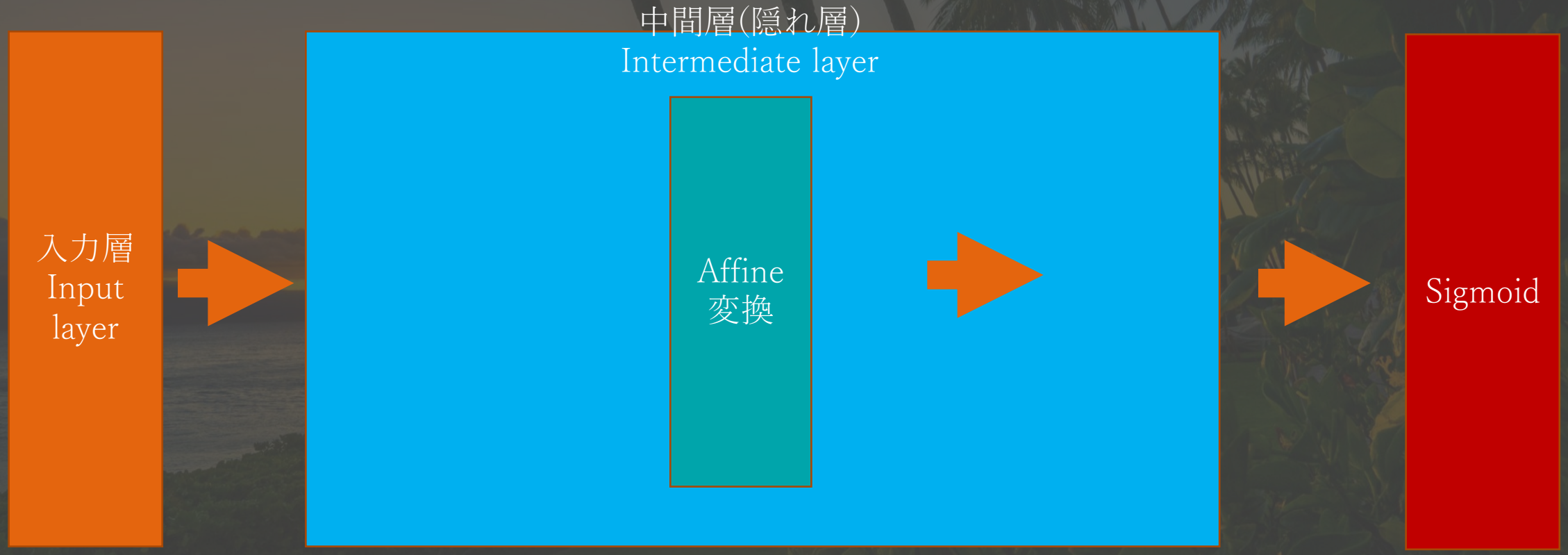
Neural Networkとは(復習)

XORの例



Neural Networkとは(復習)

Logistic回帰の例



Neural Network

畳み込みニューラルネットワークとは

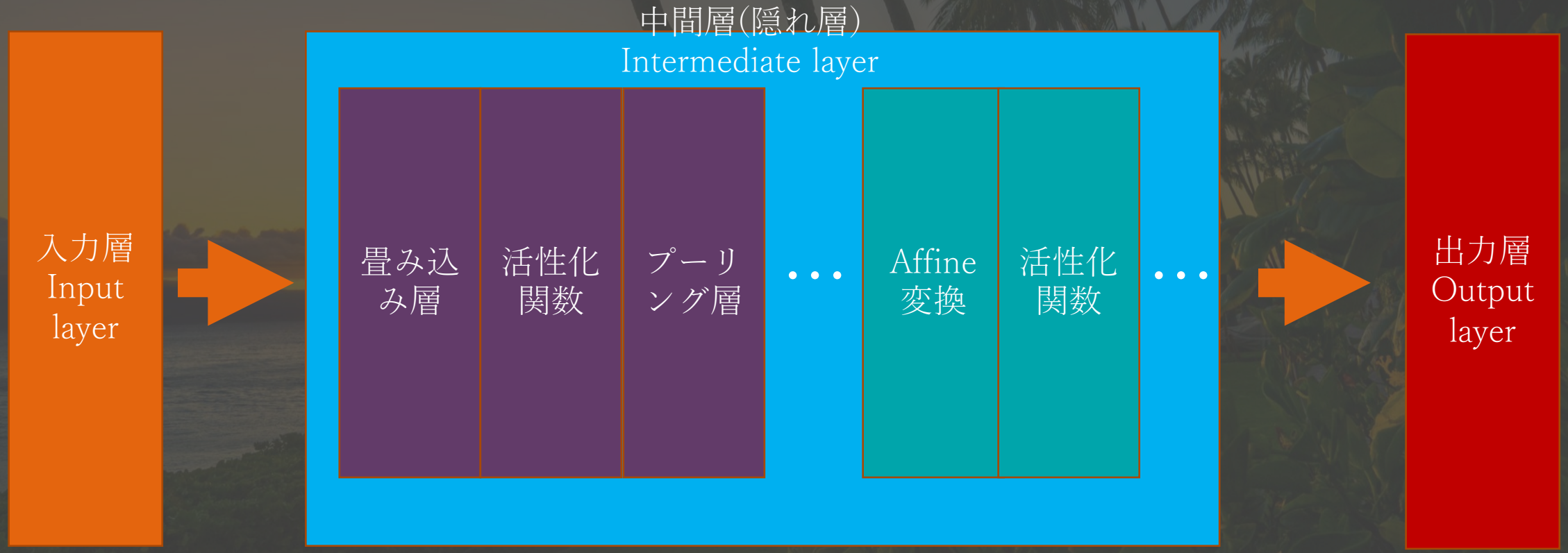
(Convolutional Neural network=CNN)

人間の視覚情報からパターンを認識している、という神経ネットワークを模倣して作られたNN

最初は画像認識の分野で広く使われていたが、今は自然言語や音声認識の分野でも用いられる。

Convolutional Neural Network

CNNの構造：畳み込み層とプーリング層が追加されている



Convolutional Neural Network

ざっくりと人の視神経のモデル化を説明する。

人が見たモノを細かく分解すると線分になる。

その線分を見分けているのが単純型細胞という細胞。

そしてその線分はモノによってずれる(例えば、目の位置は人によって違う)

そのズレを吸収して目だと判別するのが複雑型細胞という細胞。

この二つの細胞で人はモノを認識しているという仮説に基づいたニューラルネットワークモデルがCNNである。

Convolutional Neural Network

もう少しわかりやすくいうと。

単純型細胞は画像の濃淡パターンを検出(特徴抽出)の働き、

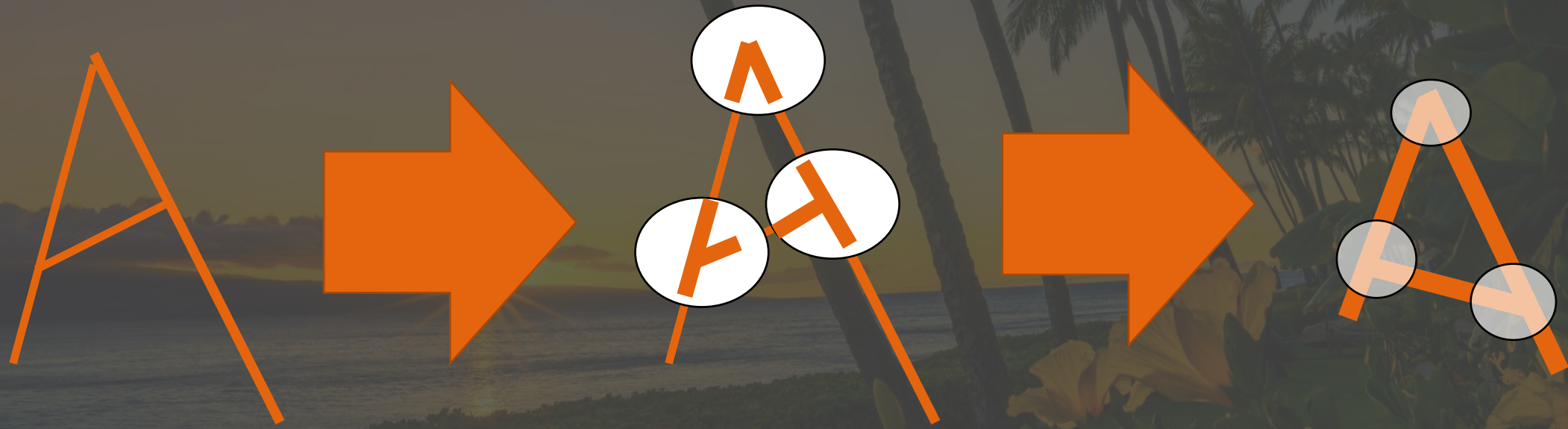
複雑型細胞はその抽出した特徴したモノの位置が変動しても同一の物体であるとみなす、という働きを持つ。

例えば多数の単純型細胞で「目」というものを認識し、それが多少ずれたとしても複雑型細胞によって「目」であると判別できるということである。

このようなメカニズムを利用したNNモデルがCNNである

Convolutional Neural Network

Aという文字を認識することを考える。

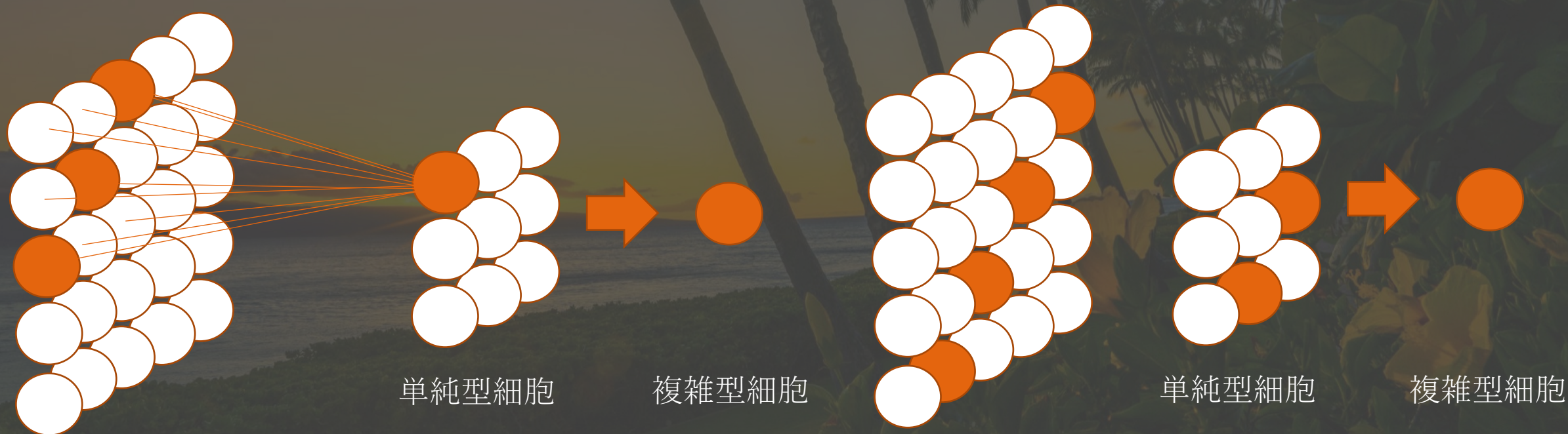


単純型細胞によって形を認識

複雑型細胞によって多少位置がずれてても認識できる

Convolutional Neural Network

モデル化(斜めの線分に反応する単純型細胞の例)



Convolutional Neural Network

CNNを構成する中間層

中間層として新たに畳み込み層とプーリング層を導入する

畳み込み層は単純型細胞に対応し、複数のピクセルをまとめて特徴抽出を行う役割を持つ。

プーリング層は複雑型細胞に対応し、鈍感な位置応答の役割を持つ。

それは次のように書ける。

Convolutional Neural Network

畳み込み層：複数のピクセルをまとめて、重要な部分を抜き出す
ずらす数：ストライド

10	10	10	0	0	0
0	10	10	10	0	0
0	0	10	10	10	0
0	0	10	10	10	0
0	10	10	10	0	0
10	10	10	0	0	0

*

filter/kernel

1	1	1
0	1	0
1	0	1

30	29	25	33
27	19	15	22
37	24	28	39
26	38	34	24

Convolutional Neural Network

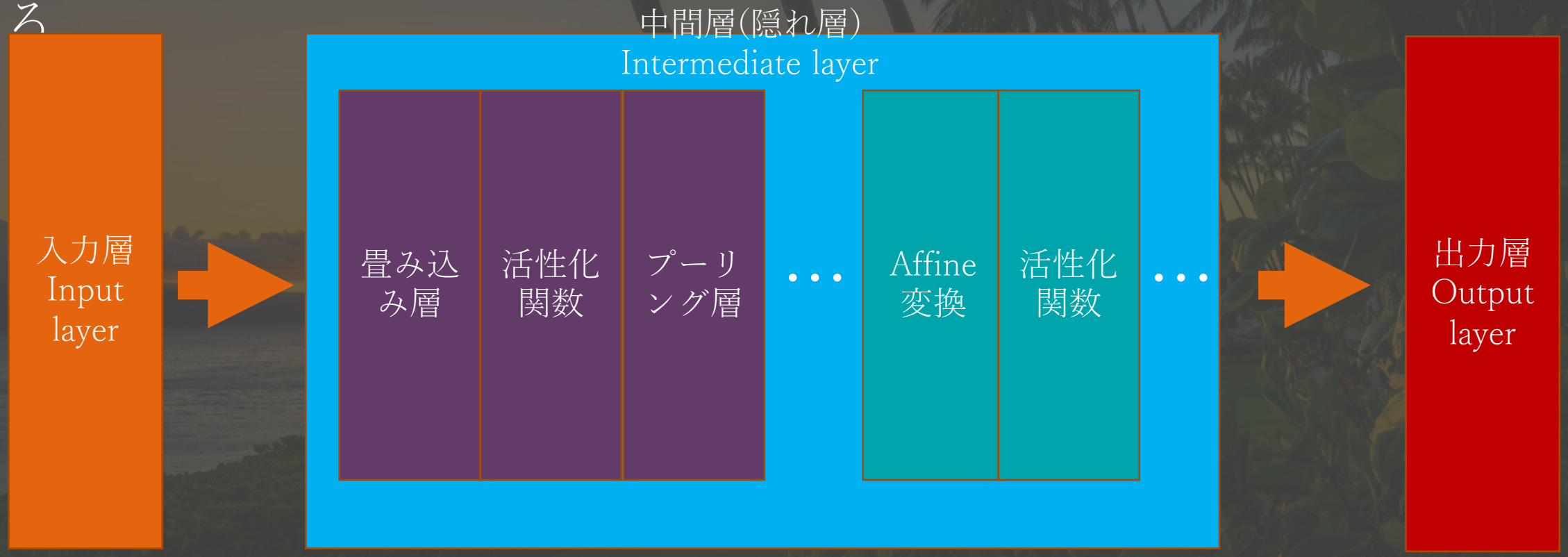
プーリング層：位置応答を鈍化させる(例：Maxプーリング)

30	29	25	33
27	19	15	22
37	24	28	39
26	38	34	24

30	33
38	39

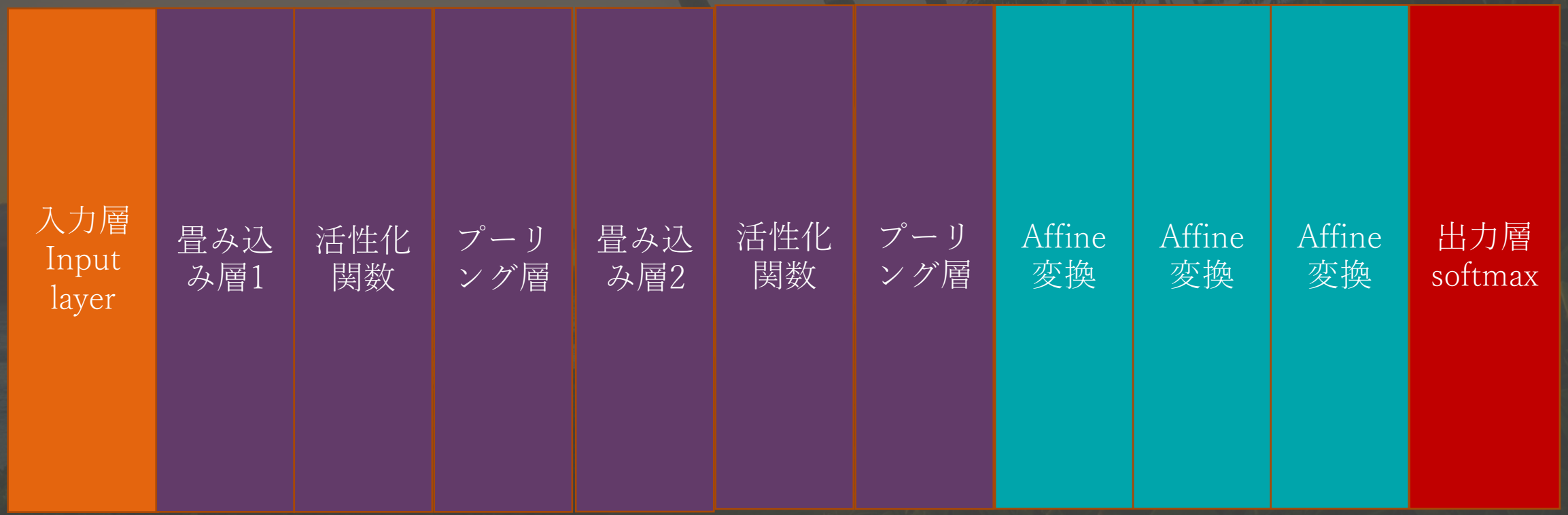
Convolutional Neural Network

CNNの構造(再掲)：畳み込み層とプーリング層が追加されている



Convolutional Neural Network

今回作るモデルの概要図



Neural Network (subject1)

まずはデータを入手する。

pytorchのMNIST（手書き数字のデータセット）を使う。

実装

データセットの入手

torchvision.datasets.MNIST ()でデータを取ってくる

root:どこに保存するか

train:Trueなら、訓練データを、Falseならテストデータを取ってくる

transform:取ってくるタイミングでデータを加工できる

download:データをダウンロードするか否か(次に使う時にダウンロードを省略できる)

実装

データセットの入手

torchvision.transforms.Compose([])

データの加工の際にさまざまな加工を行える。

今回は正規化処理のみを行うので、

..MNIST(transform=transforms.ToTensor())でも良いし、サンプルコードのようにしてもよい。

Neural Network (subject2)

train_data, test_dataという変数にtorchvision.datasets.MNIST()で取ってきたデータを格納せよ

https://pytorch.org/vision/stable/generated/torchvision.datasets.MNIST.html#torchvision.datasets.MNIST

実装

過学習を抑えるためにミニバッチ学習を行う。
ミニバッチ学習とは、データの一部分を用いて学習させること
(反対にバッチ学習とは全てのデータを用いて学習させること)
(ミニバッチ学習は勾配が敏感に反応するため、局所解に陥りづらくなる)

実装

torch.utils.DataLoader(data, batch_size, shuffle=bool, num_workers)

data:訓練データとかテストデータ

batch_size:ミニバッチのサイズ

shuffle:データを混ぜ混ぜする

num_workers:計算速度に依存するパラメータ(並列処理の数)

Neural Network (subject3)

train_loader, test_loaderという変数にミニバッチ(batch_size=100)に区切ったデータを格納せよ

https://pytorch.org/docs/stable/data.html#torch.utils.data.DataLoader

実装

学習のイメージ

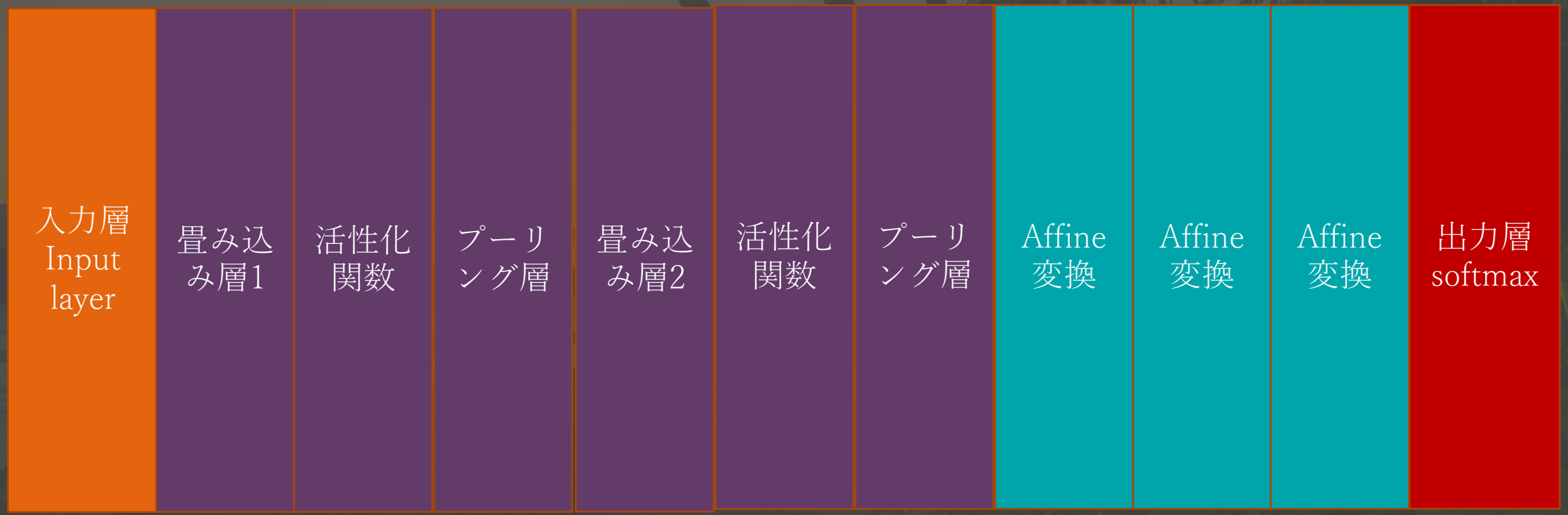
```
for epoch in epocchs:
    for batch in batches:
        勾配の初期化
        順伝播
        誤差計算
        誤差逆伝播
        重みの更新
```

実装

とりあえず、データの入手ができたので、
順伝播と中間層/出力層を定義する
class MnistClassifier(nn.module):としてクラスを定義

Convolutional Neural Network

今回作るモデルの概要図



Neural Network (subject4)

上記のように中間層/出力層を定義せよ。

Conv2d(in_channels=1, out_channels=32, kernel_size=3, stride=1)

Conv2d(32, 64, 3, 1)

MaxPool2d(2, 2)

Linear(5*5*64, 128)

Linear(128, 64)

Linear(64, 10)

Softmax(dim=1)

https://pytorch.org/docs/stable/nn.html

実装

```
def __init__(self):
        super(MnistClassifier, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=1, out_channels=32,
        kernel_size=3, stride=1) #畳み込み層
        self.pool = nn.MaxPool2d(2, 2) #プーリング層
        self.conv2 = nn.Conv2d(32, 64, 3, 1) #畳み込み層
        self.fc1 = nn.Linear(5 * 5 * 64, 128) #以下は全結合層(Affine層)
        self.fc2 = nn.Linear(128, 64)
        self.fc3 = nn.Linear(64, 10)
        self.outputs = nn.Softmax(dim=1)
```

Neural Network (subject5)

順伝播を定義せよ

ただし、全結合層に移るタイミングで

x = x.view(-1, 5*5*64)

の一行を入れること。

https://pytorch.org/docs/stable/nn.html

実装

```
def forward(self, x):
        x = self.conv1(x)
        x = F.relu(x)
        x = self.pool(x)
        x = self.conv2(x)
        x = F.relu(x)
        x = self.pool(x)
        x = x.view(-1, 5*5*64)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        x = self.outputs(x)
        return x
```

実装

次に、損失関数と最適化アルゴリズムを定義する。

```
loss_fn = torch.nn.CrossEntropy()
optimizer = torch.optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
```

Neural Network (subject6)

損失関数と最適化アルゴリズムを定義せよ。

変数名それぞれは

loss_fn

optimizer

とすること

実装

そこまでできたらいよいよ学習。
基本は
1. 勾配初期化
2. 順伝播－＞予測値を得る
3. 予測値と本物の誤差を計算
4. 誤差逆伝播
5. 重みの更新

の流れで実装する。

Neural Network (subject7)

subject_MNISTのsubject6の空白を埋めよ

(勾配の初期化→順伝播→誤差を計算→誤差逆伝播→重みの更新を実装せよ)

Neural Network (subject8)

手書き数字のサンプルを用いてできたモデルで予測してみよ

model(data)->argmax()で予測された数字が得られる。

Neural Network (subject9)

FashsionMNISTを用いたCNN実装のコードを完成させよ。

subject_FashionMnist

まとめ

CNNは人間の視覚を元に作られた→画像分類のタスクによく使われる。

畳み込み層という特徴抽出の層と、プーリング層という位置のずれを吸収する層からなる。

その他の層は基本的には通常のNNと同じ。

機械学習入門まとめ

ここまでお疲れ様でした！

相当難しかったと思います。。。

最初にお伝えした通り、みなさんにぜひできてほしいことはsklearnを使ってモデルが作れるようになる、これだけです。

model.fit(X_train, y_train)

これがすんなりかければそれでオッケーです笑

その他標準化だの、正則化だの、交差検証だの、pytorchだの難しいことをやったかと思いますが、それはみなさんが追々必要になった時に思い出すきっかけとして与えたにすぎません。(一部僕より詳しい人もおりますが、、、笑)

まずは、自分でも機械学習のモデルが作れるんだ！というところに目を向けてあげてくださいね！

機械学習入門まとめ

夏休み何やろう？の人

1. 微分積分いい気分
2. りにあああるじぇぶら(線形代数)
3. 世界の美学(確率論・統計学)

1をやると最適化アルゴリズムが理解できるようになります。

2をやるとデータ操作が上手くなります。

3をやるとモデルの解釈ができるようになります。

後期

教師なし学習(クラスタリングなど)に入っていきます。

教師なし学習の方が割とわかりやすいかな？と思うので乞うご期待！

それが終わったらデータ分析実践の授業を行います。

そこではPandas君をゴリゴリとつかっていきますので、後期もよろしくお願いしますmmm

では、良い夏休みを！