3.2 Kerasの紹介

3.2 Kerasの紹介

- KerasはPythonのディープラーニングフレームワークであり、ほぼあらゆる種類のディープラーニングモデルを定義し、訓練できる。
- Kerasの特徴

  GPU,CPUで同じコードをシームレスに実行できる

  ディープラーニングモデルのプロトタイプを簡単に作成できるAPI

  畳み込みネットワーク、リカレントネットワーク、およびそれらの組み合わせをサポートしている

  複数入力/複数出力モデル、層の共有、モデルの共有など、任意のネットワークアーキテクチャをサポートしている。

3.2.1 Keras,TensorFlow,Theano,CNTK

- Kerasはモデルレベルのライブラリ

→低レベル(微分、テンソルの操作)でなく、バックエンドエンジンとしての最適化されてテンソルライブラリを利用

→数種類のバックエンドエンジンをKerasにシームレスに接続することが可能。 TensorFlow,Theano,CNTKなど

Kerasでのコードを変更しなくても利用できる

3.2.2 Kerasを使った開発

- Kerasの標準的なワークフロー

1.訓練データ(入力テンソルと目的テンソル)を定義する

2.入力値を目的地にマッピングする複数の層からなるネットワークを定義する

3.損失関数、オプティマイザ、監視する指標を選択することで、学習プロセスを設定する。

4.モデルのfitメソッドを呼び出すことで、訓練データを繰り返し学習する。

3.2.2 Kerasを使った開発

モデルを定義する方式

・Squentialクラスを使用

・FunctionAPI(関数型API)を使用

3.2.2 Kerasを使った開発

- Squentialクラスを使った定義(2層モデル)

```
>>> from keras import layers
>>> from keras import models
>>> model = models.Sequential()
>>> model.add(layers.Dense(32,input_shape=(784,)))
>>> model.add(layers.Dense(32))
```

3.2.2 Kerasを使った開発

- FunctionAPIを使用した定義

```
>>> from keras import layers
>>> from keras import models
>>> input_tensor = layers.Input(shape=(784,))
>>> x=layers.Dense(32,activation='relu')(input_tensor)
>>> output_tensor=layers.Dense(10,activation='softmax')(x)
>>>
model=models.Model(inputs=input_tensor,outputs=output_tensor)
```

3.2.2 Kerasを使った開発

- 損失関数の定義

```
>>> from keras import optimizers
>>>
model.compile(optimizer=optimizers.RMSprop(lr=0.001),loss=
'mse',metrics=['accuracy'])
```

3.2.2 Kerasを使った開発

- Fitメソッドを呼び出す(学習プロセス)

```
>>>
model.fit(input_tensor,target_tensor,batch_size=128,epochs=10)
```