レイヤーの重み初期化方法

初期化用引数で，Kerasレイヤーの重みをランダムに初期化する確率分布を指定できます．

初期化用引数のキーワードはレイヤーにより異なりますが，大抵は単純に
`kernel_initializer` 及び `bias_initializer` です:

```
model.add(Dense(64,
                kernel_initializer='random_uniform',
                bias_initializer='zeros'))
```

利用可能な初期化方法

以下の初期化方法は全て `keras.initializers` モジュールとして定義されています．

Initializer [source]

```
keras.initializers.Initializer()
```

これは初期化クラスの基底クラスです．

Zeros [source]

```
keras.initializers.Zeros()
```

全て重みを0で初期化します．

Ones [source]

```
keras.initializers.Ones()
```

全て重みを1で初期化します．

Constant [source]

```
keras.initializers.Constant(value=0)
```

全て重みを定数で初期化します．

引数

- **value**: 浮動小数点数またはテンソルです

RandomNormal [source]

```
keras.initializers.RandomNormal(mean=0.0, stddev=0.05, seed=None)
```

正規分布に従って重みを初期化します.

引数

- **mean**: 浮動小数点数またはスカラテンソルであって分布の平均です
- **stddev**: 浮動小数点数またはスカラテンソルであって分布の標準偏差です
- **seed**: 整数. 乱数生成に使われます

RandomUniform [source]

```
keras.initializers.RandomUniform(minval=-0.05, maxval=0.05, seed=None)
```

一様分布に従って重みを初期化します.

引数

- **minval**: 浮動小数点数またはスカラテンソル. 乱数を発生する範囲の下限です
- **maxval**: 浮動小数点数またはスカラテンソル. 乱数を発生する範囲の上限です
- **seed**: 整数. 乱数生成に使われます

TruncatedNormal [source]

```
keras.initializers.TruncatedNormal(mean=0.0, stddev=0.05, seed=None)
```

切断正規分布に従って重みを初期化します.

これは正規分布と似ていますが, 平均より標準偏差の分以上離れた値は切り捨てらます. これはニューラルネットワークの重みの初期化方法として推奨されます.

引数

- **mean**: 浮動小数点数またはスカラテンソルであって分布の平均です
- **stddev**: 浮動小数点数またはスカラテンソルであって分布の標準偏差です
- **seed**: 整数. 乱数生成に使われます

VarianceScaling [source]

```
keras.initializers.VarianceScaling(scale=1.0, mode='fan_in', distribution='normal', seed=Non
```

重みテンソルのサイズ（`shape`）に合わせてスケーリングした初期化を行います．

`distribution="normal"` としたとき， 平均を 0 とし標準偏差を `stddev = sqrt(scale / n)` とした切断正規分布が使われます． ここで `n` は

- `mode="fan_in"` のとき，入力ユニットの数
- `mode="fan_out"` のとき，出力ユニットの数
- `mode="fan_avg"` のとき，入力ユニットと出力ユニットの数の平均

が使われます．

`distribution="uniform"` としたとき， [-limit, limit] を範囲とする一様分布が用いられます． ここで `limit = sqrt(3 * scale / n)` です．

引数

- **scale**: スケーリング値（正の実数）
- **mode**: "fan_in"， "fan_out"， "fan_avg" のいずれか
- **distribution**: 用いる確率分布． "normal"， "uniform" のいずれか
- **seed**: 整数． 乱数生成に使われます

Orthogonal [source]

```
keras.initializers.Orthogonal(gain=1.0, seed=None)
```

重みテンソルが直交行列となるように初期化されます．

引数

- **gain**: 最後に直交行列に乗ずる係数です
- **seed**: 整数． 乱数生成に使われます

参考文献

Saxe et al., http://arxiv.org/abs/1312.6120

Identity [source]

```
keras.initializers.Identity(gain=1.0)
```

単位行列で初期化されます． これは重みテンソルが2次正方行列の場合のみ使えます．

引数

- **gain**: 最後に単位行列に乗ずる係数です

---

glorot_normal [source]

```
glorot_normal(seed=None)
```

Glorot の正規分布（Xavier の正規分布とも呼ばれます）による初期化を返します．

これは平均を 0，標準偏差を `stddev = sqrt(2 / (fan_in + fan_out))` とする切断正規分布と同じです． ここで `fan_in` は入力ユニット数，`fant_out` は出力ユニット数です．

引数

- **seed**: 整数．乱数生成に使われます

戻り値

初期化インスタンス

参考文献

Glorot & Bengio, AISTATS 2010 - http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf

---

glorot_uniform [source]

```
glorot_uniform(seed=None)
```

Glorot の一様分布（Xavier の一様分布とも呼ばれます）による初期化を返します．

これは limit を `sqrt(6 / (fan_in + fan_out))` としたとき [limit, -limit] を範囲とする一様分布と同じです． ここで `fan_in` は入力ユニット数，`fant_out` は出力ユニット数です．

引数

- **seed**: 整数．乱数生成に使われます

戻り値

初期化インスタンス

参考文献

Glorot & Bengio, AISTATS 2010 - http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf

---

he_normal [source]

```
he_normal(seed=None)
```

He の正規分布による初期化を返します． これは平均を 0 ，標準偏差を `stddev = sqrt(2 / fan_in)` とする切断正規分布です． ここで `fan_in` は入力ユニット数です．

引数

- **seed**: 整数．乱数生成に使われます

戻り値

初期化インスタンス

参考文献

He et al., http://arxiv.org/abs/1502.01852

---

lecun_normal [source]

```
lecun_normal(seed=None)
```

LeCunの正規分布による初期化．

平均を 0 ，標準偏差を `stddev = sqrt(1 / fan_in)` とする切断正規分布からサンプルします． ここで `fan_in` は入力ユニット数です．

引数

- **seed**: 整数．乱数生成に使われます

戻り値

初期化インスタンス

参考文献

- Self-Normalizing Neural Networks
- Efficient Backprop

---

he_uniform

[source]

```
he_uniform(seed=None)
```

He の一様分布による初期化を返します． これは limit を `sqrt(6 / fan_in)` としたとき [limit, -limit] を範囲とする一様分布を用います． ここで `fan_in` は入力ユニット数です．

引数

- **seed**: 整数．乱数生成に使われます

戻り値

初期化インスタンス

参考文献

He et al., http://arxiv.org/abs/1502.01852

---

lecun_uniform

[source]

```
lecun_uniform(seed=None)
```

LeCun の一様分布による初期化を返します． これは limit を `sqrt(3 / fan_in)` とするとき [-limit, limit] を範囲とする一様分布を用います． ここで `fan_in` は入力ユニット数です．

引数

- **seed**: 整数．乱数生成に使われます

戻り値

初期化インスタンス

参考文献

LeCun 98, Efficient Backprop - http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf

---

初期化は，文字列（上記の利用可能な初期化方法のいずれかとマッチしなければならない）かcallableとして渡される．

```
from keras import initializers

model.add(Dense(64, kernel_initializer=initializers.random_normal(stddev=0.01)))

# also works; will use the default parameters.
model.add(Dense(64, kernel_initializer='random_normal'))
```

カスタマイズ

callable なオブジェクトを渡す場合には，初期化しようとする変数の `shape` と `dtype` を引数に取るように設計してください．

```
from keras import backend as K

def my_init(shape, dtype=None):
    return K.random_normal(shape, dtype=dtype)

model.add(Dense(64, kernel_initializer=my_init))
```