Zeros

Ones

Свободные члены LSTM

(Long short-term memory Долгая краткосрочная память)

Constant

0.01 - Свободные члены, если активационная функция ReLU

`Если все веса одинаковые, то и поправки будут одинаковыми`

Поэтому

RandomNormal

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

Initializer that generates tensors with a normal distribution.

Arguments

- **mean**: a python scalar or a scalar tensor. Mean of the random values to generate.
- **stddev**: a python scalar or a scalar tensor. Standard deviation of the random values to generate.
- **seed**: A Python integer. Used to seed the random generator.

RandomUniform

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

Initializer that generates tensors with a uniform distribution.

Arguments

- **minval**: A python scalar or a scalar tensor. Lower bound of the range of random values to generate.

- **maxval**: A python scalar or a scalar tensor. Upper bound of the range of random values to generate. Defaults to 1 for float types.
- **seed**: A Python integer. Used to seed the random generator.

TruncatedNormal

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

Initializer that generates a truncated normal distribution.

These values are similar to values from a `RandomNormal` except that values more than two standard deviations from the mean are discarded and re-drawn. This is the recommended initializer for neural network weights and filters.

Arguments

- **mean**: a python scalar or a scalar tensor. Mean of the random values to generate.
- **stddev**: a python scalar or a scalar tensor. Standard deviation of the random values to generate.
- **seed**: A Python integer. Used to seed the random generator.

Удобно сохранять зерно датчика случайных чисел, чтобы не сохранять всю сеть.

Когда перебирается несколько сетей.

VarianceScaling

```
keras.initializers.VarianceScaling(scale=1.0, mode='fan_in', distribution='normal', seed=None)
```

Initializer capable of adapting its scale to the shape of weights.

With `distribution="normal"`, samples are drawn from a truncated normal distribution centered on zero, with `stddev = sqrt(scale / n)` where n is:

- number of input units in the weight tensor, if mode = "fan_in"
- number of output units, if mode = "fan_out"
- average of the numbers of input and output units, if mode = "fan_avg"

With `distribution="uniform"`, samples are drawn from a uniform distribution within [-limit, limit], with `limit = sqrt(3 * scale / n)`.

Arguments

- **scale**: Scaling factor (positive float).
- **mode**: One of "fan_in", "fan_out", "fan_avg".
- **distribution**: Random distribution to use. One of "normal", "uniform".
- **seed**: A Python integer. Used to seed the random generator.

Raises

- **ValueError**: In case of an invalid value for the "scale", mode" or "distribution" arguments.

[source]

Orthogonal

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

Initializer that generates a random orthogonal matrix.

Arguments

- **gain**: Multiplicative factor to apply to the orthogonal matrix.
- **seed**: A Python integer. Used to seed the random generator.

References

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

Identity

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

Initializer that generates the identity matrix.

Only use for 2D matrices. If the long side of the matrix is a multiple of the short side, multiple identity matrices are concatenated along the long side.

Arguments

- **gain**: Multiplicative factor to apply to the identity matrix.

---

lecun_uniform

```
keras.initializers.lecun_uniform(seed=None)
```

LeCun uniform initializer.

It draws samples from a uniform distribution within [-limit, limit] where `limit` is `sqrt(3 / fan_in)` where `fan_in` is the number of input units in the weight tensor.

Arguments

- **seed**: A Python integer. Used to seed the random generator.

Returns

An initializer.

References

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

glorot_normal

`keras.initializers.glorot_normal(seed= None)`

Glorot normal initializer, also called Xavier normal initializer.

It draws samples from a truncated normal distribution centered on 0 with `stddev = sqrt(2 / (fan_in + fan_out))` where `fan_in` is the number of input units in the weight tensor and `fan_out` is the number of output units in the weight tensor.

Arguments

- **seed**: A Python integer. Used to seed the random generator.

Returns

An initializer.

References

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

glorot_uniform

`keras.initializers.glorot_uniform(seed= None)`

Glorot uniform initializer, also called Xavier uniform initializer.

It draws samples from a uniform distribution within [-limit, limit] where `limit` is `sqrt(6 / (fan_in + fan_out))` where `fan_in` is the number of input units in the weight tensor and `fan_out` is the number of output units in the weight tensor.

Arguments

- **seed**: A Python integer. Used to seed the random generator.

Returns

An initializer.

References

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

he_normal

`keras.initializers.he_normal(seed= None)`

He normal initializer.

It draws samples from a truncated normal distribution centered on 0 with `stddev = sqrt(2 / fan_in)` where `fan_in` is the number of input units in the weight tensor.

Arguments

- **seed**: A Python integer. Used to seed the random generator.

Returns

An initializer.

References

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

lecun_normal

`keras.initializers.lecun_normal(seed=None)`

LeCun normal initializer.

It draws samples from a truncated normal distribution centered on 0 with `stddev = sqrt(1 / fan_in)` where `fan_in` is the number of input units in the weight tensor.

Arguments

- **seed**: A Python integer. Used to seed the random generator.

Returns

An initializer.

References

- Self-Normalizing Neural Networks
- Efficient Backprop

he_uniform

`keras.initializers.he_uniform(seed=None)`

He uniform variance scaling initializer.

It draws samples from a uniform distribution within [-limit, limit] where `limit` is `sqrt(6 / fan_in)` where `fan_in` is the number of input units in the weight tensor.

Arguments

- **seed**: A Python integer. Used to seed the random generator.

Returns

An initializer.

References

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

An initializer may be passed as a string (must match one of the available initializers above), or as a callable:

```
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'))
```

Using custom initializers