Usage of activations

Activations can either be used through an `Activation` layer, or through the `activation` argument supported by all forward layers:

```
from keras.layers import Activation, Dense

model.add(Dense(64))
model.add(Activation('tanh'))
```

This is equivalent to:

```
model.add(Dense(64, activation='tanh'))
```

You can also pass an element-wise TensorFlow/Theano/CNTK function as an activation:

```
from keras import backend as K

model.add(Dense(64, activation=K.tanh))
```

Available activations

softmax

```
softmax(x, axis=-1)
```

Softmax activation function.

Arguments

- **x**: Input tensor.
- **axis**: Integer, axis along which the softmax normalization is applied.

Returns

Tensor, output of softmax transformation.

Raises

- **ValueError**: In case `dim(x) == 1`.

elu

```
elu(x, alpha=1.0)
```

Exponential linear unit.

Arguments

- **x**: Input tensor.
- **alpha**: A scalar, slope of negative section.

Returns

The exponential linear activation: `x` if `x > 0` and `alpha * (exp(x)-1)` if `x < 0`.

References

- [Fast and Accurate Deep Network Learning by Exponential

Linear Units (ELUs)](https://arxiv.org/abs/1511.07289)

---

selu

```
selu(x)
```

Scaled Exponential Linear Unit (SELU).

SELU is equal to: `scale * elu(x, alpha)`, where alpha and scale are pre-defined constants. The values of `alpha` and `scale` are chosen so that the mean and variance of the inputs are preserved between two consecutive layers as long as the weights are initialized correctly (see `lecun_normal` initialization) and the number of inputs is "large enough" (see references for more information).

Arguments

- **x**: A tensor or variable to compute the activation function for.

Returns

The scaled exponential unit activation: `scale * elu(x, alpha)`.

Note

- To be used together with the initialization "lecun_normal".
- To be used together with the dropout variant "AlphaDropout".

References

- Self-Normalizing Neural Networks

---

softplus

```
softplus(x)
```

Softplus activation function.

Arguments

- **x**: Input tensor.

Returns

The softplus activation: `log(exp(x) + 1)`.

---

softsign

```
softsign(x)
```

Softsign activation function.

Arguments

- **x**: Input tensor.

Returns

The softplus activation: `x / (abs(x) + 1)`.

---

relu

```
relu(x, alpha=0.0, max_value=None)
```

Rectified Linear Unit.

Arguments

- **x**: Input tensor.
- **alpha**: Slope of the negative part. Defaults to zero.
- **max_value**: Maximum value for the output.

Returns

The (leaky) rectified linear unit activation: `x` if `x > 0`, `alpha * x` if `x < 0`. If `max_value` is defined, the result is truncated to this value.

---

tanh

```
tanh(x)
```

Hyperbolic tangent activation function.

---

sigmoid

```
sigmoid(x)
```

Sigmoid activation function.

---

hard_sigmoid

```
hard_sigmoid(x)
```

Hard sigmoid activation function.

Faster to compute than sigmoid activation.

Arguments

- **x**: Input tensor.

Returns

Hard sigmoid activation:

- `0` if `x < -2.5`

- `1` if `x > 2.5`
- `0.2 * x + 0.5` if `-2.5 <= x <= 2.5`.

---

linear

```
linear(x)
```

Linear (i.e. identity) activation function.

On "Advanced Activations"

Activations that are more complex than a simple TensorFlow/Theano/CNTK function (eg. learnable activations, which maintain a state) are available as Advanced Activation layers, and can be found in the module `keras.layers.advanced_activations`. These include `PReLU` and `LeakyReLU`.