Usage of optimizers

An optimizer is one of the two arguments required for compiling a Keras model:

```
model = Sequential()
model.add(Dense(64, init='uniform', input_dim=10))
model.add(Activation('tanh'))
model.add(Activation('softmax'))

sgd = SGD(lr=0.1, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='mean_squared_error', optimizer=sgd)
```

You can either instantiate an optimizer before passing it to `model.compile()`, as in the above example, or you can call it by its name. In the latter case, the default parameters for the optimizer will be used.

```
# pass optimizer by name: default parameters will be used
model.compile(loss='mean_squared_error', optimizer='sgd')
```

SGD [source]

```
keras.optimizers.SGD(lr=0.01, momentum=0.0, decay=0.0, nesterov=False)
```

Stochastic gradient descent, with support for momentum, learning rate decay, and Nesterov momentum.

Arguments

- **lr**: float >= 0. Learning rate.
- **momentum**: float >= 0. Parameter updates momentum.
- **decay**: float >= 0. Learning rate decay over each update.
- **nesterov**: boolean. Whether to apply Nesterov momentum.

RMSprop [source]

```
keras.optimizers.RMSprop(lr=0.001, rho=0.9, epsilon=1e-08)
```

RMSProp optimizer.

It is recommended to leave the parameters of this optimizer at their default values (except the learning rate, which can be freely tuned).

This optimizer is usually a good choice for recurrent neural networks.

Arguments

- **lr**: float >= 0. Learning rate.
- **rho**: float >= 0.
- **epsilon**: float >= 0. Fuzz factor.

Adagrad [source]

```
keras.optimizers.Adagrad(lr=0.01, epsilon=1e-08)
```

Adagrad optimizer.

It is recommended to leave the parameters of this optimizer at their default values.

Arguments

- **lr**: float >= 0. Learning rate.
- **epsilon**: float >= 0.

Adadelta [source]

```
keras.optimizers.Adadelta(lr=1.0, rho=0.95, epsilon=1e-08)
```

Adadelta optimizer.

It is recommended to leave the parameters of this optimizer at their default values.

Arguments

- **lr**: float >= 0. Learning rate. It is recommended to leave it at the default value.
- **rho**: float >= 0.
- **epsilon**: float >= 0. Fuzz factor.

References

- Adadelta - an adaptive learning rate method

Adam [source]

```
keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08)
```

Adam optimizer.

Default parameters follow those provided in the original paper.

Arguments

- **lr**: float >= 0. Learning rate.
- **beta_1/beta_2**: floats, 0 < beta < 1. Generally close to 1.
- **epsilon**: float >= 0. Fuzz factor.

References

- Adam - A Method for Stochastic Optimization

Adamax [source]

```
keras.optimizers.Adamax(lr=0.002, beta_1=0.9, beta_2=0.999, epsilon=1e-08)
```

Adamax optimizer from Adam paper's Section 7. It is a variant of Adam based on the infinity norm.

Default parameters follow those provided in the paper.

Arguments

- **lr**: float >= 0. Learning rate.
- **beta_1/beta_2**: floats, 0 < beta < 1. Generally close to 1.
- **epsilon**: float >= 0. Fuzz factor.

References

- Adam - A Method for Stochastic Optimization

Nadam [source]

```
keras.optimizers.Nadam(lr=0.002, beta_1=0.9, beta_2=0.999, epsilon=1e-08, schedule_decay=0.004)
```

Nesterov Adam optimizer: Much like Adam is essentially RMSprop with momentum, Nadam is Adam RMSprop with Nesterov momentum.

Default parameters follow those provided in the paper. It is recommended to leave the parameters of this optimizer at their default values.

Arguments

- **lr**: float >= 0. Learning rate.
- **beta_1/beta_2**: floats, 0 < beta < 1. Generally close to 1.
- **epsilon**: float >= 0. Fuzz factor.

References

[1] Nadam report - http://cs229.stanford.edu/proj2015/054_report.pdf [2] On the importance of initialization and momentum in deep learning - - **http**://www.cs.toronto.edu/~fritz/absps/momentum.pdf