Оптимизаторы

Использование оптимизаторов

Оптимизатор — один из двух аргументов, необходимых для компиляции модели Keras:

```
from keras import optimizers
model = Sequential()
model.add(Dense(64, kernel_initializer='uniform',
input_shape=(10,)))
model.add(Activation('softmax'))
sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9,
nesterov=True)
model.compile(loss='mean_squared_error', optimizer=sgd)
```

Вы можете либо инстанцировать оптимизатор перед передачей его в model.compile(), как в приведенном выше примере, либо вызвать его по имени. В последнем случае будут использоваться параметры оптимизатора по умолчанию.

```
# при передаче оптимизатора по имени: будут использоваться
параметры по умолчанию
model.compile(loss='mean_squared_error', optimizer='sgd')
```

Общие для всех оптимизаторов Keras параметры.

Параметры clipnorm и clipvalue могут быть использованы со всеми оптимизаторами для управления градиентной обрезкой:

```
from keras import optimizers
# Все параметры градиентов будут обрезаны до максимальной
нормы от 1.
sgd = optimizers.SGD(lr=0.01, clipnorm=1.)
```

```
from keras import optimizers
# Все параметры градиентов будут обрезаны до максимального
значения 0.5 и минимального значения -0.5.
sgd = optimizers.SGD(lr=0.01, clipvalue=0.5)
```

[source]

SGD

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

Стохастический оптимизатор градиентного спуска.

Включает в себя поддержку импульса, затухания скорости обучения и импульса Нестерова.

Аргументы

- **learning_rate:** float >= 0. Скорость обучения.
- **momentum:** float >= 0. Параметр, ускоряющий SGD в соответствующем направлении и гасящий колебания.
- **nesterov:** boolean. Применять ли импульс Нестерова?

[source]

RMSprop

```
keras.optimizers.RMSprop(learning_rate=0.001, rho=0.9)
```

RMSProp оптимизатор.

Рекомендуется оставить параметры этого оптимизатора на значениях по умолчанию (за исключением скорости обучения, которая может быть свободно настроена).

Аргументы

- **learning_rate**: float >= 0. Скорость обучения.
- **rho**: float >= 0.

[source]

Adagrad

```
keras.optimizers.Adagrad(learning_rate=0.01)
```

Адаград оптимизатор.

Адаград — это оптимизатор, в котором скорость обучения зависит от конкретных параметров, которые адаптированы к тому, как часто параметр обновляется во время обучения. Чем больше обновлений получает параметр, тем меньше скорость обучения.

Рекомендуется оставлять параметры этого оптимизатора на значениях по умолчанию.

Аргументы

- **learning_rate**: float >= 0. Уровень начального обучения.

[source]

Adadelta

```
keras.optimizers.Adadelta(learning_rate=1.0, rho=0.95)
```

Ададельта-оптимизатор.

Adadelta — более надежное расширение Adagrad, которое адаптирует скорость обучения на основе скользящего окна обновления градиентов, вместо того, чтобы накапливать все градиенты прошлых лет. Таким образом, Adadelta

продолжает обучение даже тогда, когда сделано много обновлений. По сравнению с Adagrad, в оригинальной версии Adadelta нет необходимости устанавливать начальную скорость обучения. В этой версии, как и в большинстве других оптимизаторов Keras, можно установить начальную скорость обучения и коэффициент распада.

Рекомендуется оставить параметры этого оптимизатора в их значениях по умолчанию.

Аргументы

- **learning_rate:** float >= 0. Начальная скорость обучения, по умолчанию 1. Рекомендуется оставить значение по умолчанию.
- **rho: float >= 0.** Коэффициент распада ададельты, соответствующий доле градиента, который необходимо сохранять на каждом шаге.

[source]

Adam

```
keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9,
beta_2=0.999, amsgrad=False)
```

Оптимизатор Adam.

Параметры по умолчанию соответствуют параметрам, указанным в оригинальной работе.

Аргументы

- **learning_rate:** float >= 0. Скорость обучения.
- **beta_1:** float, 0 < beta < 1. Обычно близко к 1.
- **beta_2:** float, 0 < beta < 1. Обычно близко к 1.
- **amsgrad:** boolean. Применять ли AMSGrad вариант этого алгоритма из статьи «О конвергенции Adam и не только».

[source]

Adamax

```
keras.optimizers.Adamax(learning_rate=0.002, beta_1=0.9,
beta_2=0.999)
```

Оптимизатор Adamax из раздела 7 документации Adam.

Это вариант Adam, основанный на норме бесконечности. Параметры по умолчанию соответствуют параметрам, приведенным в статье.

Аргументы

- **Скорость обучения:** float >= 0. Скорость обучения.
- **beta_1:** float, 0 < beta < 1. Обычно близко к 1.
- **beta_2:** float, 0 < beta < 1. Обычно близко к 1.

[source]

Nadam

```
keras.optimizers.Nadam(learning_rate=0.002, beta_1=0.9,
beta_2=0.999)
```

Nesterov Adam оптимизатор.

Так же, как Adam, по сути являестя RMSprop с импульсом, так и Nadam это RMSprop с импульсом Нестерова.

Параметры по умолчанию соответствуют параметрам, приведенным в статье. Рекомендуется оставить параметры этого оптимизатора на их значениях по умолчанию.

Аргументы

- **learning_rate:** float >= 0. Скорость обучения.
- **beta_1:** float, 0 < beta < 1. Обычно близко к 1.
- **beta_2:** float, 0 < beta < 1. Обычно близко к 1.