Ошибки

Использование функций потерь

Функция потерь (или объективная функция, или функция оценки результатов оптимизации) является одним из двух параметров, необходимых для компиляции модели:

```
model.compile(loss='mean_squared_error', optimizer='sgd')
from keras import losses
model.compile(loss=losses.mean_squared_error,
optimizer='sgd')
```

Можно либо передать имя существующей функции потерь, либо передать символическую функцию TensorFlow/Theano, которая возвращает скаляр для каждой точки данных и принимает следующие два аргумента:

y_true: истинные метки. Тензор TensorFlow/Theano.

y_pred: Прогнозы. Тензор TensorFlow/Theano той же формы, что и y_true.

Фактически оптимизированная цель — это среднее значение выходного массива по всем точкам данных.

Доступные функции потери

mean_squared_error

```
keras.losses.mean_squared_error(y_true, y_pred)
```

mean_absolute_error

```
keras.losses.mean_absolute_error(y_true, y_pred)
```

---

mean_absolute_percentage_error

```
keras.losses.mean_absolute_percentage_error(y_true, y_pred)
```

---

mean_squared_logarithmic_error

```
keras.losses.mean_squared_logarithmic_error(y_true, y_pred)
```

---

squared_hinge

```
keras.losses.squared_hinge(y_true, y_pred)
```

---

hinge

```
keras.losses.hinge(y_true, y_pred)
```

---

categorical_hinge

```
keras.losses.categorical_hinge(y_true, y_pred)
```

---

logcosh

```
keras.losses.logcosh(y_true, y_pred)
```

Логарифм гиперболического косинуса ошибки прогнозирования.

log(cosh(x)) приблизительно равен (x ** 2) / 2 для малого x и abs(x) — log(2) для большого x. Это означает, что 'logcosh' работает в основном как средняя квадратичная ошибка, но не будет так сильно зависеть от случайного сильно неправильного предсказания.

Аргументы

- **y_true:** тензор истинных целей.
- **y_pred:** тензор прогнозируемых целей.

Возвращает

Тензор с одной записью о скалярной потере на каждый сэмпл.

---

huber_loss

```
keras.losses.huber_loss(y_true, y_pred, delta=1.0)
```

---

categorical_crossentropy

```
keras.losses.categorical_crossentropy(y_true, y_pred,
from_logits=False, label_smoothing=0)
```

---

sparse_categorical_crossentropy

```

```

```
keras.losses.sparse_categorical_crossentropy(y_true, y_pred,
from_logits=False, axis=-1)
```

---

binary_crossentropy

```
keras.losses.binary_crossentropy(y_true, y_pred,
from_logits=False, label_smoothing=0)
```

---

kullback_leibler_divergence

```
keras.losses.kullback_leibler_divergence(y_true, y_pred)
```

---

poisson

```
keras.losses.poisson(y_true, y_pred)
```

---

cosine_proximity

```
keras.losses.cosine_proximity(y_true, y_pred, axis=-1)
```

---

is_categorical_crossentropy

```
keras.losses.is_categorical_crossentropy(loss)
```

Примечание: при использовании потери categorical_crossentropy ваши данные должны быть в категориальном формате (например, если у вас 10 классов, то целью для каждой выборки должен быть 10-мерный вектор, который является полностью нулевым, за исключением 1 в индексе, соответствующем классу выборки). Для того, чтобы преобразовать целые данные в категорические, можно использовать утилиту Keras to_categorical:

```
from keras.utils import to_categorical
categorical_labels = to_categorical(int_labels,
num_classes=None)
```

При использовании переменной sparse_categorical_crossentropy loss, ваши данные должны быть целыми. Если у вас есть категориальные данные, следует использовать categoryical_crossentropy.

categoryical_crossentropy — это еще один термин для обозначения потери лога по нескольким классам.