損失関数の利用方法

損失関数（損失関数や最適スコア関数）はモデルをコンパイルする際に必要なパラメータの1つです:

```
model.compile(loss='mean_squared_error', optimizer='sgd')
```

```
from keras import losses

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

既存の損失関数の名前を引数に与えるか，各データ点に対してスカラを返し，以下の2つの引数を取るTensorFlow/Theanoのシンボリック関数を与えることができます:

- **y_true**: 正解ラベル．TensorFlow/Theano テンソル
- **y_pred**: 予測値．y_trueと同じshapeのTensorFlow/Theano テンソル

実際に最適化される目的関数値は全データ点における出力の平均です．

このような関数の実装例に関しては，losses sourceを参照してください．

利用可能な損失関数

mean_squared_error

```
mean_squared_error(y_true, y_pred)
```

mean_absolute_error

```
mean_absolute_error(y_true, y_pred)
```

mean_absolute_percentage_error

```
mean_absolute_percentage_error(y_true, y_pred)
```

mean_squared_logarithmic_error

```
mean_squared_logarithmic_error(y_true, y_pred)
```

squared_hinge

```
squared_hinge(y_true, y_pred)
```

---

hinge

```
hinge(y_true, y_pred)
```

---

categorical_hinge

```
categorical_hinge(y_true, y_pred)
```

---

logcosh

```
logcosh(y_true, y_pred)
```

予測誤差のハイパボリックコサインの対数.

`log(cosh(x))`は`x`が小さければ`(x ** 2) / 2`とほぼ等しくなり，`x`が大きければ`abs(x) - log(2)`とほぼ等しくなります．つまり'logcosh'は平均二乗誤差とほぼ同じように働きます．しかし，時折ある乱雑な誤った予測にそれほど強く影響されません.

---

categorical_crossentropy

```
categorical_crossentropy(y_true, y_pred)
```

---

sparse_categorical_crossentropy

```
sparse_categorical_crossentropy(y_true, y_pred)
```

---

binary_crossentropy

```
binary_crossentropy(y_true, y_pred)
```

---

kullback_leibler_divergence

```
kullback_leibler_divergence(y_true, y_pred)
```

---

poisson

```
poisson(y_true, y_pred)
```

---

cosine_proximity

```
cosine_proximity(y_true, y_pred)
```

NOTE: `categorical_crossentropy` を使う場合，目的値はカテゴリカルにしなければいけません．（例．もし10クラスなら，サンプルに対する目的値は，サンプルのクラスに対応する次元の値が1，それ以外が0の10次元のベクトルです）． 整数の目的値からカテゴリカルな目的値に変換するためには，Keras utilityの `to_categorical` を使えます．

```
from keras.utils.np_utils import to_categorical

categorical_labels = to_categorical(int_labels, num_classes=None)
```