Custom Loss
Graded Quiz • 30 min

Congratulations! You passed!

Grade received 100%   **Latest Submission Grade** 100%   **To pass** 80% or higher

Go to next item

1. One of the ways of declaring a loss function is to import its object. Is the following code correct for using a loss object?

1 / 1 point

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

[✓ False]
True

Correct
Correct! You import from tensorflow.keras.losses.

2. It is possible to add parameters to the object call when using the loss object.

1 / 1 point

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

[✓ True]
False

Correct
Correct! Adding parameters provides flexibility for other steps such as hyperparameter tuning.

3. You learned that you can do hyperparameter tuning within custom-built loss functions by creating a wrapper function around the loss function with hyperparameters defined as its parameter. What is the purpose of creating a wrapper function around the original loss function?

1 / 1 point

```
def my_huber_loss_with_threshold(threshold):
  def my_huber_loss(y_true, y_pred):
    error = y_true - y_pred
    is_small_error = tf.abs(error) <= threshold
    small_error_loss = tf.square(error) / 2
    big_error_loss = threshold * (tf.abs(error) - (0.5 * threshold))

    return tf.where(is_small_error, small_error_loss, big_error_loss)

  return my_huber_loss
```

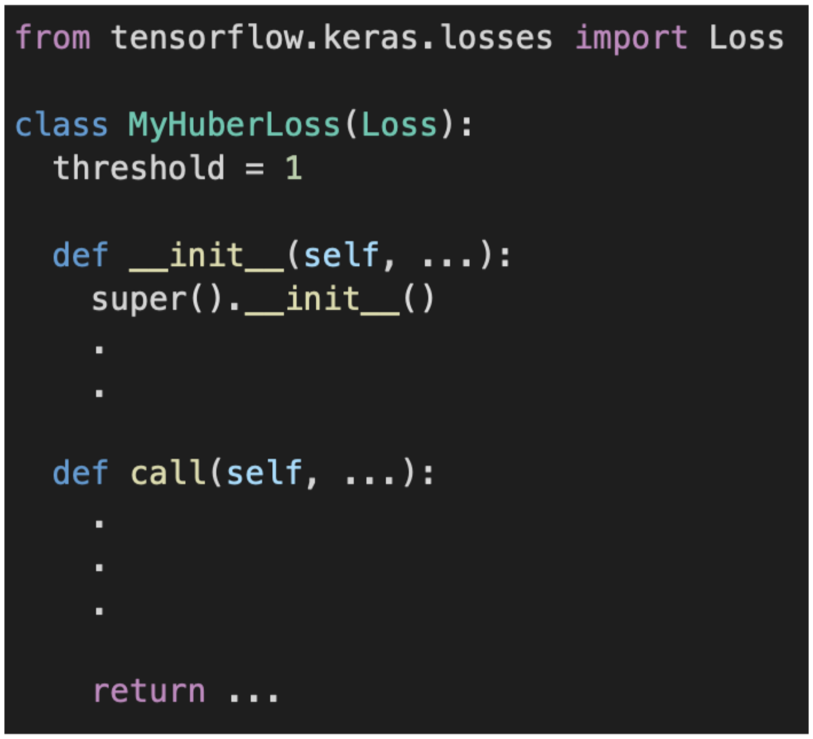
- ○ No particular reason, it just looks neater this way.
- ○ The loss (model.compile(..., loss =)) expects a function that is only a wrapper function to the loss function itself.
- [✓ The loss (model.compile(..., loss =)) expects a function with two parameters, y_true and y_pred, so it is not possible to pass a 3rd parameter (threshold) to the loss function itself. This can be achieved by creating a wrapper function around the original loss function.]
- ○ That's one way of doing it. We can also do the same by passing y_true, y_pred and threshold as parameters to the loss function itself.

Correct

Correct!

4. One other way of implementing a custom loss function is by creating a class with two function definitions, init and call.

1 / 1 point



```
from tensorflow.keras.losses import Loss

class MyHuberLoss(Loss):
  threshold = 1

  def __init__(self, ...):
    super().__init__()
    .
    .

  def call(self, ...):
    .
    .
    .

    return ...
```

Which of the following is correct?

Which of the following is correct?

- We pass the hyperparameter (threshold) , y_true and y_pred to the init function, and the call function returns the init function.
- [✓ We pass the hyperparameter (threshold) to the init function, y_true and y_pred to the call function.]
- We pass y_true and y_pred to the init function, the hyperparameter (threshold) to the call function.
- We pass the hyperparameter (threshold) , y_true and y_pred to the call function, and the init function returns the call function.

Correct

Correct! Threshold is passed into the inherent init function to initialize it as a class object and pass it back to the base class, and y_true and y_pred are passed into the call function when the class object, threshold, is instantiated.

.

Which of the following is correct?

- ○ We pass the hyperparameter (threshold) , y_true and y_pred to the init function, and the call function returns the init function.
- [✓ We pass the hyperparameter (threshold) to the init function, y_true and y_pred to the call function.]
- ○ We pass y_true and y_pred to the init function, the hyperparameter (threshold) to the call function.
- ○ We pass the hyperparameter (threshold) , y_true and y_pred to the call function, and the init function returns the call function.

✓ **Correct**

Correct! Threshold is passed into the inherent init function to initialize it as a class object and pass it back to the base class, and y_true and y_pred are passed into the call function when the class object, threshold, is instantiated.