深度之眼
deepshare.net

常用损失函数与自定义损失函数

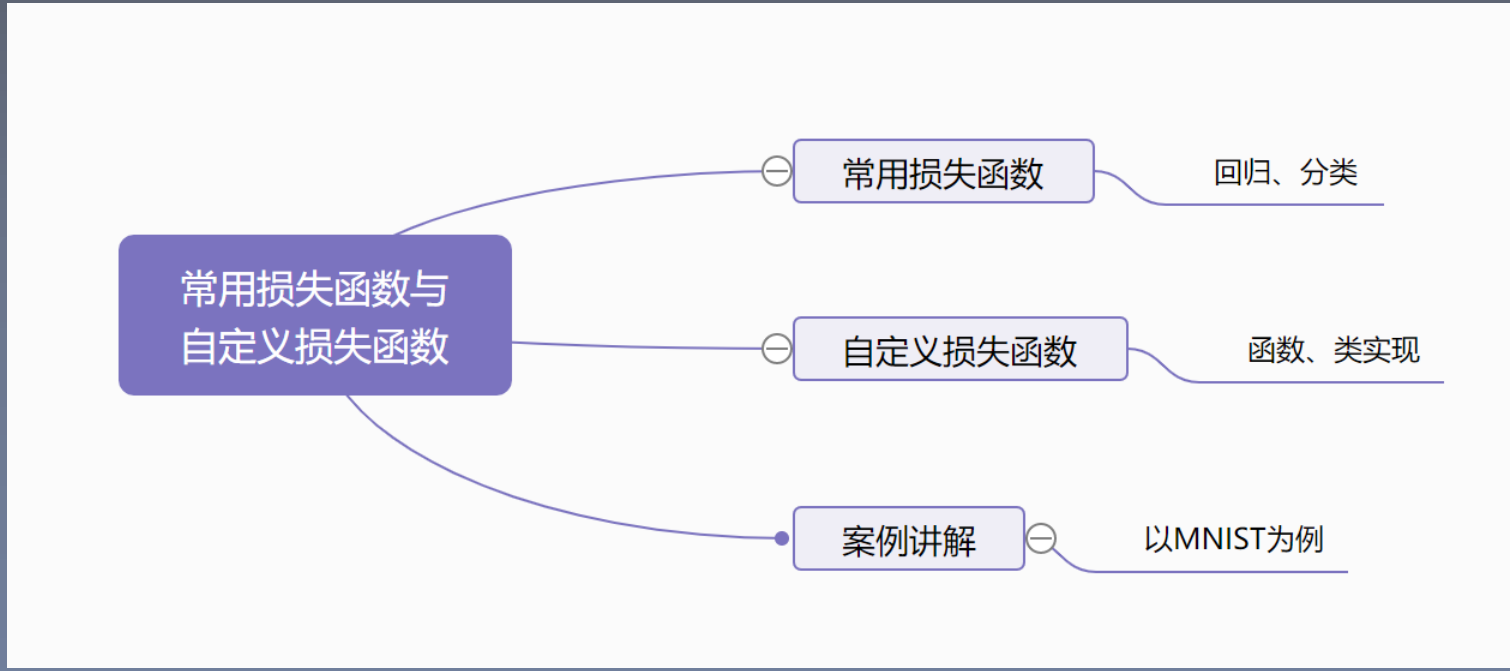
导师：GAUSS

深度之眼
deepshare.net

目录

知识树

Knowledge tree



深度之眼
deepshare.net

常用损失函数

常用损失函数

tf.keras.losses

参考网站：https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses

常用损失函数

- mean_squared_error（平方差误差损失，用于回归，简写为 mse, 类实现形式为 MeanSquaredError 和 MSE）
- binary_crossentropy(二元交叉熵，用于二分类，类实现形式为 BinaryCrossentropy)
- categorical_crossentropy(类别交叉熵，用于多分类，要求label为onehot编码，类实现形式为 CategoricalCrossentropy)
- sparse_categorical_crossentropy(稀疏类别交叉熵，用于多分类，要求label为序号编码形式，类实现形式为 SparseCategoricalCrossentropy)

这里列举部分！！！

更多参考：https://www.tensorflow.org/api_docs/python/tf/keras/losses

常用损失函数

BinaryCrossentropy 和 binary_crossentropy有什么区别?

前者是类的实现形式，后者是函数的实现形式。

```python
def binary_crossentropy(target, output, from_logits=False):
    if not from_logits:
        if (isinstance(output, (ops.EagerTensor, variables_module.Variable)) or
                output.op.type != 'Sigmoid'):
            epsilon_ = _constant_to_tensor(epsilon(), output.dtype.base_dtype)
            output = clip_ops.clip_by_value(output, epsilon_, 1. - epsilon_)

            # Compute cross entropy from probabilities.
            bce = target * math_ops.log(output + epsilon())
            bce += (1 - target) * math_ops.log(1 - output + epsilon())
            return -bce
        else:
            # When sigmoid activation function is used for output operation,
            # use logits from the sigmoid function directly to compute loss
            # to prevent collapsing zero when training.
            assert len(output.op.inputs) == 1
            output = output.op.inputs[0]
    return nn.sigmoid_cross_entropy_with_logits(labels=target, logits=output)
```

```python
@keras_export('keras.losses.BinaryCrossentropy')
class BinaryCrossentropy(LossFunctionWrapper):
    def __init__(self,
                 from_logits=False,
                 label_smoothing=0,
                 reduction=losses_utils.ReductionV2.AUTO,
                 name='binary_crossentropy'):
        super(BinaryCrossentropy, self).__init__(
            binary_crossentropy,
            name=name,
            reduction=reduction,
            from_logits=from_logits,
            label_smoothing=label_smoothing)
        self.from_logits = from_logits
```

交叉熵损失函数

多分类交叉熵损失函数：

$$L = \frac{1}{N}\sum_{i} L_i = \frac{1}{N}\sum_{i} -\sum_{c=1}^{M} y_{ic}\log(p_{ic})$$

其中：

- M — 类别的数量；
- y_{ic} — 指示变量（0或1），如果该类别和样本 i 的类别相同就是1，否则是0；
- p_{ic} — 对于观测样本 i 属于类别 c 的预测概率。

现在我们利用这个表达式计算下面例子中的损失函数值：

tf版本:

```
cce = tf.keras.losses.CategoricalCrossentropy()
loss = cce(
  [[1., 0., 0.], [0., 1., 0.], [0., 0., 1.]],
  [[.9, .05, .05], [.05, .89, .06], [.05, .01, .94]])
print('Loss: ', loss.numpy())  # Loss: 0.0945
```

Numpy版本:

```
a =np.array([[1., 0., 0.], [0., 1., 0.], [0., 0., 1.]],)
b =np.array([[.9, .05, .05], [.05, .89, .06],
[.05, .01, .94]])
np.average(-np.sum(a*np.log(b),axis=1))
```

深度之眼
deepshare.net

自定义损失函数

自定义损失函数

tf.keras.losses.Loss

参考：https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Loss

两种方法自定义函数：

函数的实现形式

类的实现形式

自定义损失函数

举个例子

类的实现形式：

函数的实现形式：

```
class MeanSquaredError(tf.keras.losses.Loss):
    def call(self, y_true, y_pred):
        return tf.reduce_mean(tf.square(y_pred - y_true))


def MeanSquaredError(y_true, y_pred):
  return tf.reduce_mean(tf.square(y_pred - y_true))
```

Focal loss损失函数

论文地址：**Focal Loss for Dense Object Detection**

相关讨论：https://www.zhihu.com/question/63581984

原始论文的Focal loss损失函数针对于二分类，这里改为**多分类损失函数**：

$$FL(p_t) = \sum_{c=1}^{m} -(1-p_t)^{\gamma} * y_c * \log(p_t)$$

其中，m表示类别数，y_c是真实标签，p_t是预测标签，γ是调节因子。

Focal loss损失函数

Focal loss 损失函数的实现

```python
#多分类的focal loss 损失函数
class SparseFocalLoss(tf.keras.losses.Loss):

    def __init__(self,gamma=2.0,alpha=0.25,class_num=10):
        self.gamma = gamma
        self.alpha = alpha
        self.class_num = class_num
        super(SparseFocalLoss, self).__init__()

    def call(self,y_true,y_pred):
        y_pred = tf.nn.softmax(y_pred,axis=-1)
        epsilon = tf.keras.backend.epsilon()
        y_pred = tf.clip_by_value(y_pred, epsilon, 1.0)

        y_true = tf.one_hot(y_true,depth=self.class_num)
        y_true = tf.cast(y_true,tf.float32)

        loss = -  y_true * tf.math.pow(1 - y_pred, self.gamma) * tf.math.log(y_pred)

        loss = tf.math.reduce_sum(loss,axis=1)
        return loss
```

Focal loss损失函数

Focal loss 损失函数的实现

```
def focal_loss(gamma=2.0,alpha=0.25):
    def focal_loss_fixed(y_true, y_pred):
        y_pred = tf.nn.softmax(y_pred,axis=-1)
        epsilon = tf.keras.backend.epsilon()
        y_pred = tf.clip_by_value(y_pred, epsilon, 1.0)

        y_true = tf.cast(y_true,tf.float32)

        loss = -  y_true * tf.math.pow(1 - y_pred, gamma) * tf.math.log(y_pred)

        loss = tf.math.reduce_sum(loss,axis=1)
        return  loss
    return focal_loss_fixed
```

深度之眼
deepshare.net

案例讲解

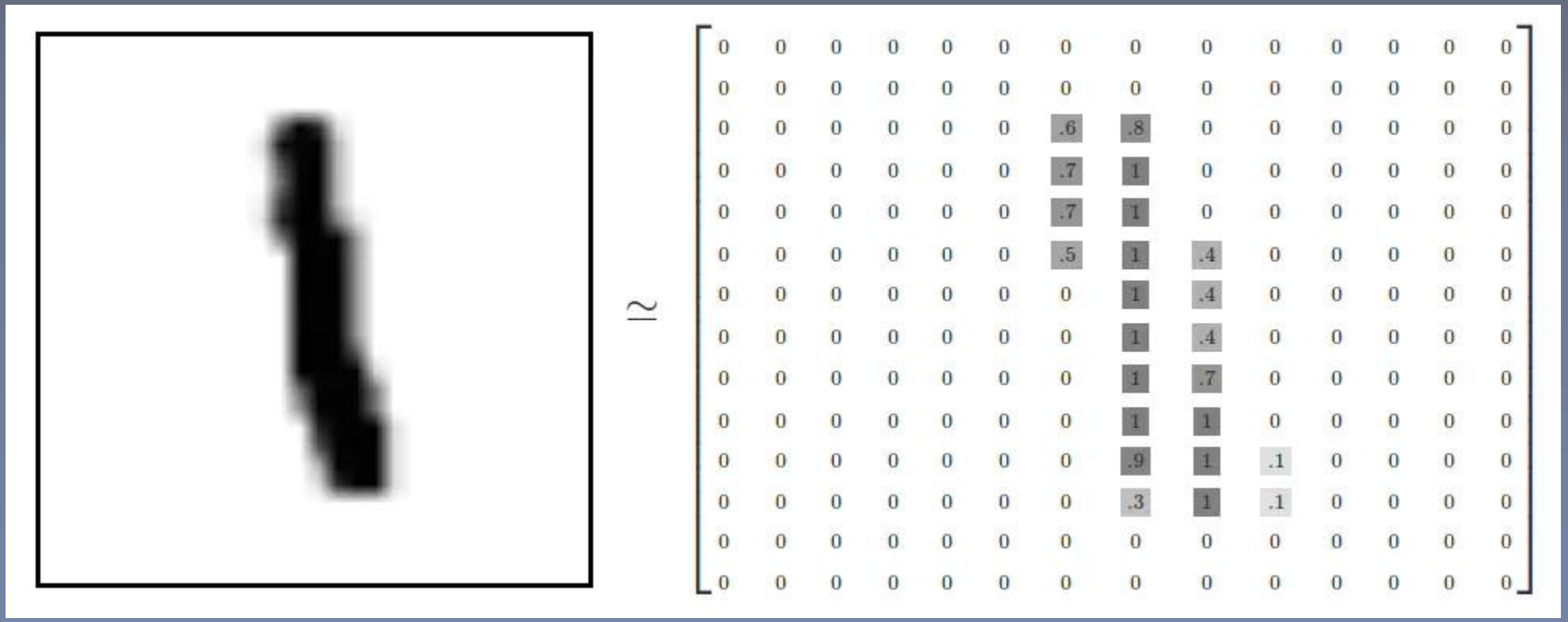
MNIST数据集

MNIST是一个入门级的计算机视觉数据集，它包含各种手写数字图片，如右图：

它也包含每一张图片对应的标签，告诉我们这个是数字几。比如，右图的第一行这10张图片的标签分别是0，4，1，9，2，1，3，1，4，3。

MMIST数据集

每一张图都是由（28，28，1）的矩阵组成：



案例建模

实战1： Focal Loss实现自定义损失函数（以mnist数据集构建图像分类算法）

在notebook中详解

本节小结

Summary

常用损失函数与自定义损失函数	常用损失函数	常用的损失函数（场景）
	自定义损失函数	类的实现形式
		函数的实现形式
	案例讲解	

结语

我 说

GAUSS老师个人公众号，主要分享NLP、推荐、比赛实战相关知识！

深度之眼
deepshare.net

联系我们：

电话：18001992849

邮箱：service@deepshare.net

Q Q： 2677693114

公众号

客服微信