基于蒙特卡罗 Dropout 近似贝叶斯模块实现

该模块主要是实现 MC Dropout 的具体操作，这是一种在评估阶段实现不确定性估计的方法，通过多次前向传递并使用 Dropout 来模拟不同的模型，与近似贝叶斯的计算提供了坚实的实践基础，MC Dropout 原理如图 4 所示：

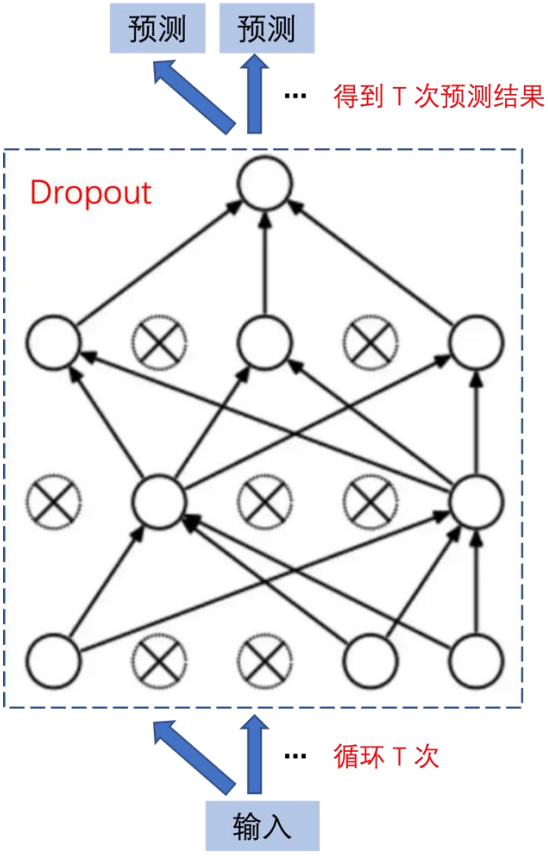

图 4 MC Dropout 原理图

- **模块整体框架**

```
def mc_dropout_evaluate(model, gpus, classes, x, T=30, batch_size=64, training=True):

    y_T = np.zeros((T, len(x['input_ids']), classes))
    acc = None

    logger.info("Yielding predictions looping over ...")
    strategy = tf.distribute.MirroredStrategy()
    data = tf.data.Dataset.from_tensor_slices(x).batch(batch_size*gpus)
    dist_data = strategy.experimental_distribute_dataset(data)
    for i in range(T):
        print(i)
```

```
12.
13.         y_pred = []
14.         with strategy.scope():
15.             def eval_step(inputs):
16.                 return model(inputs, training=training).numpy()  # [:,0]
17.
18.             def distributed_eval_step(dataset_inputs):
19.                 return strategy.run(eval_step, args=(dataset_inputs,))
20.
21.             for batch in dist_data:
22.                 pred = distributed_eval_step(batch)
23.                 for gpu in range(gpus):
24.                     y_pred.extend(pred)
25.
26.         y_T[i] = tf.nn.softmax(np.array(y_pred))
27.
28.     y_mean = np.mean(y_T, axis=0)
29.     assert y_mean.shape == (len(x['input_ids']), classes)
30.
31.     y_pred = np.array([np.argmax(np.bincount(row))
32.                        for row in np.transpose(np.argmax(y_T, axis=-1))])
33.     logger.info("y_pred")
34.     logger.info(y_pred)
35.     assert y_pred.shape == (len(x['input_ids']),)
36.
37.     y_var = np.var(y_T, axis=0)
38.     assert y_var.shape == (len(x['input_ids']), classes)
39.
40.     return y_mean, y_var, y_pred, y_T
```

- **参数解释**
  - **model**: 要进行评估的模型。
  - **gpus**: 使用的 GPU 数量，用于分布式计算。
  - **classes**: 输出类别数目。
  - **x**: 输入数据，包含模型所需的输入字典，例如 input_ids、attention_mask 及 token_type_ids 等。
  - **T**: MC Dropout 采样的次数，即前向传递的次数。
  - **batch_size**: 批大小，用于每次前向传递时的样本数量。
  - **training**: 一个布尔值，指示在评估过程中是否启用 Dropout。

1. 初始化存储预测结果的数组

```
y_T = np.zeros((T, len(x['input_ids']), classes))
acc = None
```

这里初始化了一个大小为 (T, len(x['input_ids']), classes) 的数组 y_T，用于存储每次前向传递的预测结果。acc 初始化为 None，为后续处理做准备。

2. 打印进程日志

```
logger.info("Yielding predictions looping over ...")
```

打印日志信息，指示开始进行多次前向传递，为 debug 提供便利。

3. 设置分布式策略

```
strategy = tf.distribute.MirroredStrategy()
data = tf.data.Dataset.from_tensor_slices(x).batch(batch_size * gpus)
dist_data = strategy.experimental_distribute_dataset(data)
```

在这部分代码中：tf.distribute.MirroredStrategy 用于分布式训练，管理多个 GPU 并行进行计算。之后将输入数据 x 转换为 tf.data.Dataset 并按 batch_size * gpus 进行批处理，以适应多 GPU 的并行计算。而 strategy.experimental_distribute_dataset(data) 则将数据分布在多个设备上，以此加快系统的运行速度，包括模型的训练与推理速度等。

4. 蒙特卡罗 (MC) Dropout 推理循环

```
for i in range(T):
    print(i)
    y_pred = []

    with strategy.scope():
        def eval_step(inputs):
            return model(inputs, training=training).numpy()  # Perform forward pass

        def distributed_eval_step(dataset_inputs):
            return strategy.run(eval_step, args=(dataset_inputs,))

        for batch in dist_data:
            pred = distributed_eval_step(batch)
            for gpu in range(gpus):
                y_pred.extend(pred)

    y_T[i] = tf.nn.softmax(np.array(y_pred))
```

这段代码是此模块的核心所在，它指导了 MC Drpout 操作的完整流程，即：

- **循环 T 次**：每次循环进行一次前向传递，使用 Dropout 以不同方式模拟模型的不确定性。

- **评估步骤定义：**
- **eval_step**: 定义前向传递步骤，在模型的 training 模式下进行。
- **distributed_eval_step**: 将评估步骤分发到多个设备上。
- **批处理评估：**
- 遍历分布式数据集 dist_data。
- 对每个批次运行 distributed_eval_step，并将结果存储到 y_pred 中。
- **归一化输出：**
- 对生成的预测结果应用 softmax 激活函数，并存储在 y_T 的第 i 次采样位置。

5. 计算均值预测

```
y_mean = np.mean(y_T, axis=0)
```

此处我们计算 MC Dropout 结果的均值，得到最终的均值预测结果，均值向量维度大小为 (len(x['input_ids']), classes)。

6. 确定最终预测类别

```
y_pred = np.array([np.argmax(np.bincount(row))
                   for row in np.transpose(np.argmax(y_T, axis=-1))])
```

此处我们进行逐样本投票：对每个样本在 T 次采样中的预测结果取多数投票结果，得到最终的类别预测。数组 y_pred 包含每个输入样本的最终类别预测，其向量维度大小为 (len(x['input_ids']),)。

7. 计算预测结果的方差

```
y_var = np.var(y_T, axis=0)
```

计算每个样本在 T 次预测中的方差，得到结果的不确定性度量，并为后续 loss 权重的计算提供必需值，其向量维度大小为 (len(x['input_ids']), classes)。

8. 返回结果

```
return y_mean, y_var, y_pred, y_T
```

在计算完所有必需的值之后，我们将结果返回用于后续采样与权重计算等步骤。这四个值的含义分别为：

- **y_mean**: 多次采样的均值预测。
- **y_var**: 预测方差，反映预测不确定性。
- **y_pred**: 最终类别预测结果。
- **y_T**: 所有采样的原始预测结果。

这个模块在恶意隐写载体检测任务中，发挥着提供预测不确定性的无可替代的作用。MC Dropout 提供了一种量化的方法来获取模型对每个输入的预测置信度，这是后续贝叶斯神经网络理论实现的基础，为其实现提供了实践层面的可行性。