深蓝学院
shenlanxueyuan.com

语音合成：第五章作业讲解

主讲人 从坚

GMM-Attention复现

深蓝学院 shenlanxueyuan.com

- 作业要求：
  - 实现Tacotron系统中的GMM attention模块，完成模型训练和测试
- 作业提示：

$$\widehat{\sigma_i}, \widehat{\Delta_i}, \widehat{\omega_i} = f(h_i)$$

$$\alpha_{i,j} = \sum_{k=1}^{K} \frac{w_{i,k}}{Z_{i,k}} \exp\left(-\frac{(j-\mu_{i,k})^2}{2(\sigma_{i,k})^2}\right)$$

$$\boldsymbol{\mu}_i = \boldsymbol{\mu}_{i-1} + \boldsymbol{\Delta}_i$$

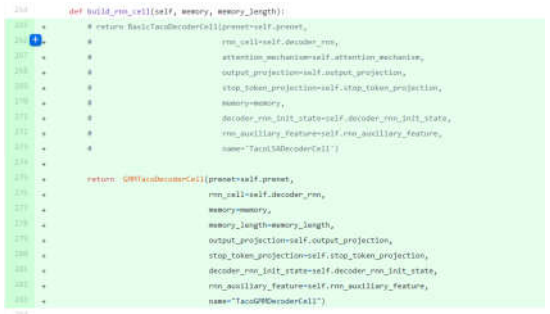
Table 1. Conversion of intermediate parameters computed in (7) to final mixture parameters for the three tested GMM-based attention mechanisms. $\mathrm{S}_{\max}(\cdot)$ is the softmax function, while $\mathrm{S}_{+}(\cdot)$ is the soft-plus function.

	$\boldsymbol{Z}_i$	$\boldsymbol{w}_i$	$\boldsymbol{\Delta}_i$	$\boldsymbol{\sigma}_i$
V0 [1]	$\mathbf{1}$	$\exp(\hat{\boldsymbol{w}}_i)$	$\exp(\hat{\boldsymbol{\Delta}}_i)$	$\sqrt{\exp(-\hat{\boldsymbol{\sigma}}_i)/2}$
V1	$\sqrt{2\pi\boldsymbol{\sigma}_i^2}$	$\mathrm{S}_{\max}(\hat{\boldsymbol{w}}_i)$	$\exp(\hat{\boldsymbol{\Delta}}_i)$	$\sqrt{\exp(\hat{\boldsymbol{\sigma}}_i)}$
V2	$\sqrt{2\pi\boldsymbol{\sigma}_i^2}$	$\mathrm{S}_{\max}(\hat{\boldsymbol{w}}_i)$	$\mathrm{S}_{+}(\hat{\boldsymbol{\Delta}}_i)$	$\mathrm{S}_{+}(\hat{\boldsymbol{\sigma}}_i)$

GMM-Attention复现

深蓝学院 shenlanxueyuan.com

- 作业提示：
  - 修改以下几个文件
  - 04_seq2seq_tts/tacotron/modules/decoder_cells.py
  - 04_seq2seq_tts/tacotron/modules/attention.py
  - 04_seq2seq_tts/tacotron/modules/decoder.py



```
+ class GMMTacoDecoderCellState(
+     collections.namedtuple("BasicTacoDecoderCellState",
+                            ("rnn_cell_state", "attention", "time", "mu",
+                             "alignment_history"))):
+     def replace(self, **kwargs):
+         return super(GMMTacoDecoderCellState, self)._replace(**kwargs)
+
+ class GMMTacoDecoderCell(BasicTacoDecoderCell):
+     """RNN cell for GMM attention based decoder"""
+
```

```
+ class GMMAttentionComputer():
+     """GMM attention"""
+
+     def __init__(self,
+                  memory,
+                  input_length,
+                  num_mixture=3,
+                  scope='gmm_attention'):
+         super(GMMAttentionComputer, self).__init__()
+         self.memory = memory
+         self.input_length = input_length
+         self.num_mixture = num_mixture
+         self.batch_size = get_tensor_shape(memory)[0]
+         self.memory_length = get_tensor_shape(memory)[1]
+         self.scope = scope
+
+     def __call__(self, cell_output, prev_mu):
+         with tf.variable_scope(self.scope):
```

GMM-Attention复现

深蓝学院 shenlanxueyuan.com

- 作业提示：

```
def __call__(self, cell_output, prev_mu):
    with tf.variable_scope(self.scope):
        # cell_output: [B, D_c]
        mlp_out = tf.layers.dense(cell_output, units=256, activation=tf.nn.relu)
        mlp_out = tf.layers.dense(mlp_out, units=3 * self.num_mixture)  # [B, 3 * num_mixture]
        sigma_hat, Delta_hat, omega_hat = tf.split(mlp_out, 3, axis=1)  # [B, num_mixture]
                                                    cell_output 过FC得到 sigma_hat, Delta_hat, omega_hat
        # v0 version
        omega = tf.exp(omega_hat)  # [B, num_mixture]
        Delta = tf.exp(Delta_hat)  # [B, num_mixture]
        sigma = tf.sqrt(tf.exp(-sigma_hat) / 2)  # [B, num_mixture]
        Z = 1                                                    计算omega, sigma, mu
        mu = prev_mu + Delta  # [B, 1] + [B, num_mixture] = [B, num_mixture]

        omega = tf.expand_dims(omega, axis=2)  # [B, num_mixture, 1]
        sigma = tf.expand_dims(sigma, axis=2)  # [B, num_mixture, 1]
        mu = tf.expand_dims(mu, axis=2)  # [B, num_mixture, 1]
                                                    维度变化以便下面计算alignment
        j = tf.range(self.memory_length)  # [T]      j: [1,2,3,4..T]
        j = tf.reshape(j, (1, 1, self.memory_length))  # [1, 1, T]
        j = tf.tile(j, (self.batch_size, self.num_mixture, 1))  # [B, num_mixture, T]
        j = tf.cast(j, tf.float32)  # [B, num_mixture, T]

        alignments = (omega / Z) * tf.exp(- (j - mu) ** 2 / (2 * sigma ** 2 + 1e-8))  # [B, num_mixture, T]
        alignments = tf.reduce_sum(alignments, axis=1)  # [B, T]
                                                    计算alignment
        mask = tf.cast(tf.sequence_mask(self.input_length,
                                        maxlen=self.memory_length),
                       tf.float32)  # [B, T]
        memory = self.memory * tf.expand_dims(mask, 2)  # [B, T, Dm] * [B, T, 1]
        # self.memory:
        attention = math_ops.matmul(tf.expand_dims(alignments, 1), memory)  # [B, 1, T] * [B, T, Dm] -> [B, 1, Dm]
        attention = tf.squeeze(attention, 1)                    attention

    return attention, alignments, tf.squeeze(mu, 2)  # [B, 1, Dm], [B, T], [B, 3]
```

参考链接

1. https://github.com/npujcong/TTS_Course/commit/186268b6cd272e259c5d22b3d9bf38d1906cc09c
2. https://github.com/npujcong/TTS_Course

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening