

Gated Attention 条件熵代理易懂手册

自动生成

November 6, 2025

Abstract

这份文档面向第一次接触`gated-att/`的同学，目标是把Gated Attention 条件熵代理讲成“看完就懂、上手就能用”的版本。我们会用通俗比喻串联整条链路：为什么要加这个模块、每一步在代码里的位置、它和Slot+Cross 方案的区别、调参时应该先动哪个旋钮。你可以把它当作一篇讲稿/说明书，直接拿去给团队做分享。

Contents

1	背景与动机	2
1.1	条件熵最小化的目标	2
1.2	Gated Attention 的角色	2
2	组件详解	2
2.1	GatedAttentionPooling: 按类自适应聚合	2
2.2	GatedAttentionCEM: 加权统计与阈值化	2
2.3	训练循环中的集成	3
3	与Slot+Cross 方案的对比	4
4	数值稳定性与调参要点	4
4.1	关键超参数	4
4.2	稳定性技巧	4
5	实践建议	4
6	结论	5

1 一张图先感知：我们在解决什么？

- 目的：让同一类别的特征更集中（条件熵更低），减少模型泄露细节的风险。
- 做法：不做复杂聚类，只学一个“注意力分布”，挑出类内真正关键的样本，按权重算出“类内波动大不大”。
- 位置：所有逻辑都在gated-att/model_training_parallel_pruning.py 的GatedAttentionPooling和GatedAttentionCEM 中。

2 整体故事：像是给班级点名

1. 先给每个班（类别）归好队：把同一标签的特征拿出来。
2. 点名+ 权重：Gated Attention 会给每个学生一个“重要程度”。被判定为关键的样本权重高，普通样本权重低。
3. 算班里的平均表现和波动：根据权重算加权均值、加权方差，看看班里到底稳不稳。
4. 超过阈值就扣分：如果某个维度波动太大超过阈值，就对总损失贡献一个正则惩罚。
5. 加权汇总到总损失：班里人多的影响更大，人少的影响更小。

3 组件1：GatedAttentionPooling（谁是关键样本？）

3.1 直觉解释

这一步像是给班里的每个学生打一个“关注度分数”。模型通过两个分支来决定：

- \tanh 分支 (W_V) 告诉我们：“这个学生的特征方向怎么看？”
- σ 分支 (W_U) 像一扇门：“我要不要放大/缩小这名学生的影响？”

两个分支相乘后，再映射成一个标量，最后经过softmax，得到一组权重 a ，保证所有权重加起来是1。

3.2 和代码对照

- 定义位置：第17–44 行。
- 输入尺寸：[样本数 M , 特征维 D]。
- 输出：形状 $[M, 1]$ 的softmax 权重。

4 组件2: GatedAttentionCEM (用权重来判定是否“稳”)

4.1 步骤拆解

1. **LayerNorm**: 防止某些维度异常大造成偏差。
2. 调用池化: 拿到上一节的权重 a 。
3. 算加权均值 μ_c : 像做“班级平均成绩”, 只是每个人成绩乘以他的权重。
4. 算加权方差 σ_c^2 : 看加权后的样本偏离均值多少。
5. 防止数值爆炸: 用`eps` 把方差下界住, 再取`log`。
6. 阈值判定: 如果 $\log \sigma_c^2$ 超过阈值, 就记入惩罚; 否则就是0。阈值跟`var_threshold`、`reg_strength` 有关。
7. 平均到类级指标 L_c : 对所有维度取平均, 得到这个班级的“惩罚分”。
8. 计算类内MSE: 只是为了日志监控, 和梯度无关。
9. 按样本占比加权: 班里人越多, 在总损失里占比越大。

4.2 关键变量一览

- `var_threshold`: 阈值基准, 越大越宽容。
- `reg_strength`: 来自主程序的正则强度, 和阈值一起决定严不严格。
- `attention_loss_scale`: 训练时会乘在`rob_loss` 上, 控制梯度力度。

5 训练循环里怎么接入?

5.1 触发条件

- 三个开关: 不是随机中心、 $\lambda > 0$ 、当前`epoch` 超过`warmup`。
- 隐藏层宽度: 自动取 $\min(512, \max(64, D/4))$, 不用手调。
- 参数注册: 第一次建模块时会把参数加到优化器里。

5.2 梯度流程

1. `rob_loss` 先反向一次, 把梯度存下来。
2. 清梯度, 对主任务反向。
3. 把正则梯度按学习率缩放后加回编码器参数。
4. 注意力模块的梯度直接累加。

5.3 容错机制

- `rob_loss` 或 `intra_mse` 出现NaN/Inf, 会被置0。
- 如果类内样本数 ≤ 1 , 直接跳过, 保证梯度路径连续。

6 Slot+Cross vs. Gated Attention: 该用谁?

6.1 快速对比表

- **Slot+Cross:**
优点——能建模多个子簇, 对复杂数据更有力;
缺点——结构复杂, 算力和调参成本更高。
- **Gated Attention:**
优点——结构简单、只有一个注意力池化, 部署轻量;
缺点——天然偏向“单簇”, 对类内非常multimodal 的数据可能不够精细。

6.2 建议

- 如果设备有限或想优先验证思路, 可以先用Gated Attention。
- 如果发现类内仍旧很散, 考虑切换或加入Slot+Cross。

7 调参/排障套路

7.1 常调参数

- `attention_loss_scale`: 直接影响正则力度; 训练稳定后可慢慢调大。
- `var_threshold`: 阈值大, 惩罚少; 阈值小, 惩罚多。
- `eps`: 如果log 里还是出现NaN, 可以适当增大。

7.2 日志里要看什么?

- **`rob_loss` 曲线**: 应该随着训练逐渐稳定, 不建议直接拉到很大。
- **`intra_mse`**: 如果一直不降, 说明注意力权重没抓对关键样本。
- **注意力权重分布**: 可以打印权重直方图, 看看是否过于平均或过于集中。

8 常见问题速查

- Q: 权重会不会全都平均?
A: 初始可能接近平均, 但随着训练, W_V 、 W_U 会把关键样本权重大幅拉升。
- Q: 阈值怎么选?
A: 默认值通常够用, 如果你希望正则更激进, 可以把`var_threshold` 调小一些。
- Q: 怎么知道正则有没有帮到隐私?
A: 除了看`rob_loss`, 还可以在攻击测试里比较是否更难重构。同时关注主任务准确率, 避免过正则。

9 下一步可以做什么?

- 结合同目录下的流程图 (PDF/PNG), 向同事展示每一步的输入输出。
- 记录每个epoch 的`rob_loss`、`intra_mse`、准确率, 做一张曲线图观察趋势。
- 如果需要进一步解释, 可以把注意力权重打印成热力图, 让团队直观看到“谁最重要”。