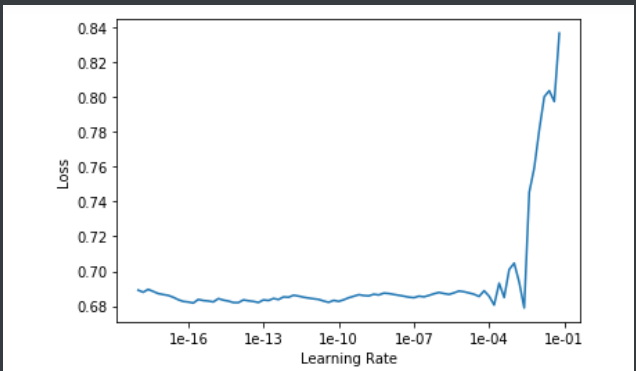
使用fastai寻找lr从而优化模型结果

在论文中说CLR相对于自适应optimizer节省了一部分计算量，这个计算量来自于自适应optimizer需要根据之前的值决定当前lr，而CLR不需要，因而CLR可以视为自适应optimizer的竞争者。

但是在实际使用过程中，还是存在一些问题。

1. 说明文档和当前fastai的版本不对应，导致文档中某些类中的某些函数无法使用。
2. 模型过于集成化，适合于新手练手或者寻找初始学习率，和用pytorch搭建并训练模型的过程暂时不可同日而语。（见 https://github.com/JiangRIVERS/Some_tricks_when_using_Pytorch/blob/master/使用Pytorch训练模型时踩的坑.md 中的第五条）

下面是使用fastai进行IDH训练的结果。fastai应该是采用了某些节省显存的操作，之前运行模型使12G的GPU爆显存，通过checkpoint操作使GPU显存降到4GB，而使用fastai时，不使用checkpoint操作（fastai集成化导致checkpoint操作本身不可使用）GPU显存占用降到2GB。



根据lr_loss曲线选择学习率lr=1e-3，之前自适应optimizer选取lr=1e-5

但是结果一般

epoch	train_loss	valid_loss	time
0	0.762231	1.099184	01:35
1	0.716263	0.689350	01:36
2	0.712488	0.830092	01:34
3	0.661349	0.680300	01:36
4	0.624829	0.659896	01:37
5	0.620334	0.649616	01:37
6	0.608802	0.690210	01:37
7	0.567499	0.632152	01:36
8	0.532524	0.689106	01:37
9	0.591171	0.753937	01:37
10	0.490172	0.709585	01:36
11	0.492996	1.274054	01:35
12	0.458058	0.840993	01:34
13	0.440150	0.930384	01:34
14	0.472666	0.879321	01:34
15	0.510134	0.965749	01:34
16	0.410688	1.498914	01:34
17	0.353249	0.815888	01:34
18	0.408619	1.118572	01:33
19	0.406074	0.991530	01:34
20	0.385039	1.442025	01:34
21	0.232443	1.156307	01:34
22	0.355276	1.568212	01:34
23	0.318549	0.968253	01:34
24	0.387852	1.492949	01:33
25	0.237177	1.109474	01:34
26	0.214187	1.392316	01:34

所以归根到底还是模型和数据的原因，fastai可能会帮助选取一个较为合适的lr，而且集成化的模型操作适合新手去了解并跑简单的模型，但是并不能真正意义上做到较大的提升且有些操作在fastai上较为不便。