Keras 搭建生成对抗网络（GAN）

生成对抗网络（Generative Adversarial Networks，GAN）最早由 Ian Goodfellow 在 2014 年提出，它的核心思想是：同时训练两个相互协作、同时又相互竞争的深度神经网络（一个称为生成器 Generator，另一个称为判别器 Discriminator）来处理无监督学习的相关问题。生成器的目的是尽量去学习真实的数据分布, 而判别器的目的是尽量正确判别输入数据是来自真实数据还是来自生成器，这两个相互对抗并迭代优化的过程使得 D 和G 的性能不断提升, 当最终 D 的判别能力提升到一定程度, 并且无法正确判别数据来源时, 可以认为这个生成器 G 已经学到了真实数据的分布。

任意可微分的函数都可以用来表示 GAN 的生成器和判别器, 由此, 我们用可微分函数 D 和 G 来分别表示判别器和生成器, 它们的输入分别为真实数据 x 和随机变量 z. 如果判别器的输入来自真实数据, 标注为 1. 如果输入样本为 G(z), 标注为 0. 这里 D 的目标是实现对数据来源的二分类判别: 真 (来源于真实数据 x 的分布) 或者伪 (来源于生成器的伪数据 G(z)) 。

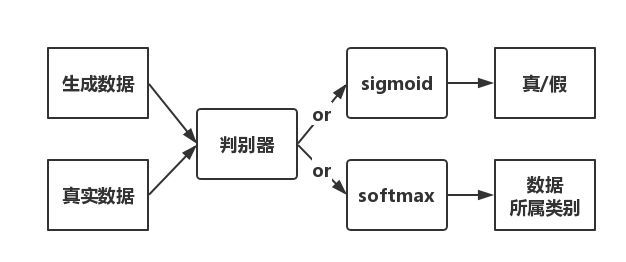
WCGAN实现源码链接：https://gist.github.com/myurasov/6ecf449b32eb263e7d9a7f6e9aed5dc2

## GAN模型

GAN 模型，需要搭建两个模型：一个是判别器模型，代表警察；另一个是对抗模型，代表制造假币的犯罪分子。

### 判别器

识别此数据是真实的，还是由 Generative model 生成的。



鉴别器将（真实/生成）数据作为输入，并有两个输出：

* 用Sigmoid非线性激活来得到输入的概率值，判断真假；
* 预测数据所属类别与softmax激活；

权重是从正态分布初始化的，stddev为0.02，所以初始剪切不会切断所有的权重

weight\_init **=** RandomNormal(mean**=**0., stddev**=**0.02)

参数可训练

set\_trainability(D, True)

判别器的输出为 Sigmoid 函数，因此采用了二元交叉熵为判别器的损失函数

d\_loss ='binary\_crossentropy'

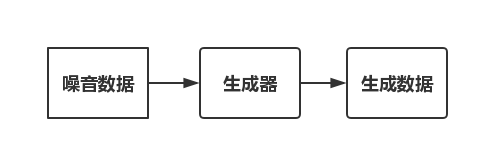
d\_loss =’categorical\_crossentropy’

优化算法设置了权值衰减和clipvalue等参数来稳定后期的训练过程

RMSprop(lr=0.00005, clipvalue=1.0, decay=6e-8)

### 生成器

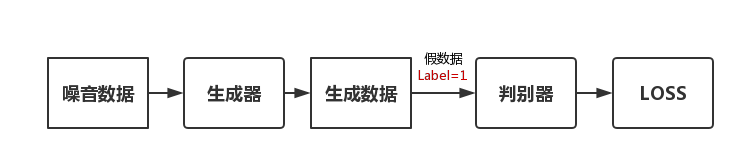
模仿真实数据生成接近的数据。



中间过程，无损失。

## 对抗模型GAN

如图所示，对抗模型的基本结构是判别器和生成器的叠加。生成器试图骗过判别器，同时从其反馈中提升自己。其中，除了学习速率的降低和相对权值衰减之外，训练参数与判别器模型中的训练参数完全相同。



set\_trainability(D, False)

RMSprop(lr=0.0004, clipvalue=1.0, decay=3e-8)

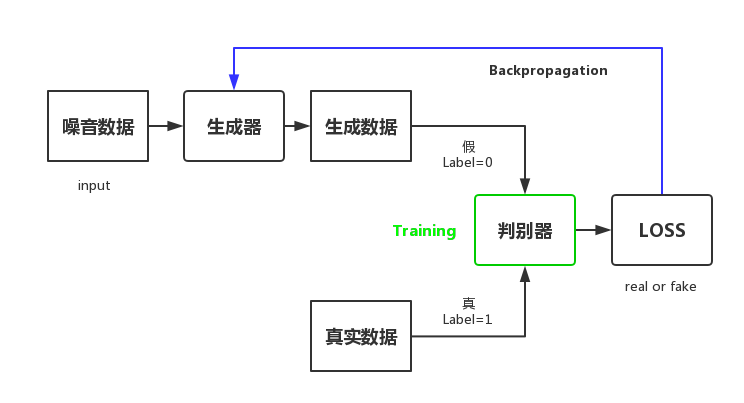
loss='binary\_crossentropy'

## 训练

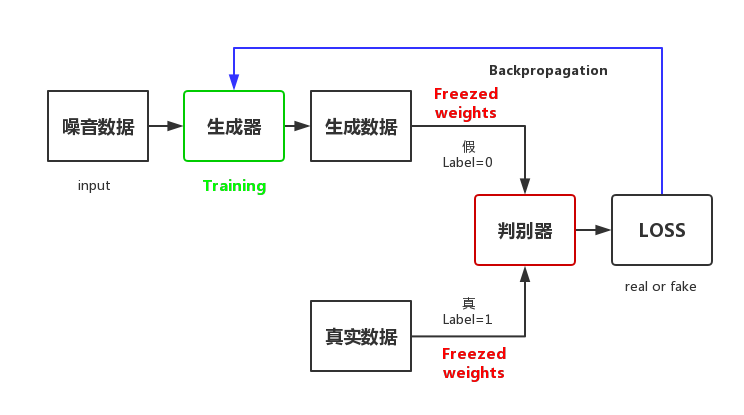
搭好模型之后，训练是最难实现的部分。这里我们首先用**真实图像和假图像**对判别器模型单独进行训练，以提高其判断的正确性。接着，训练对抗模型，每次训练生成器时要冻住判别器模型set\_trainability (Discriminator, False)。最后，**交替**对判别器模型和对抗模型展开训练。

|  |
| --- |
| 培训过程包括以下步骤：   1. 解冻***D***的重量，使他们可以学习。 2. 向***D***提供真实样本，并尝试通过在损失函数中乘以-1来最大化其输出，并使其值最小化。 3. 向***D***提供生成样本，并尽量使它的输出最小化； 4. 重复步骤3和4，*D\_ITERS次；* 5. 冻结***D***权重 6. 训练GAN - ***D（G）*** - 并试图最小化其输出。这优化了***G***的权重，使得冻结的和已经被训练的***D***检测到生成的假样本为真样本。 |

如下图展示了**判别器模型训练**的基本流程。



如下图展示了**对抗模型训练**的基本流程。

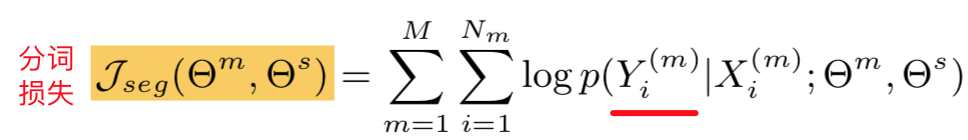


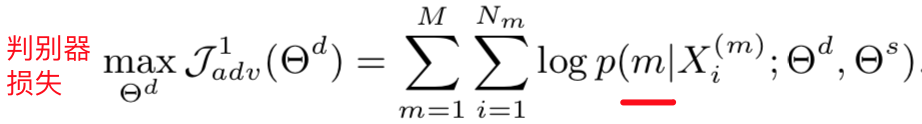
|  |
| --- |
| def sample\_noise(G, noise\_dim=10, n\_samples=10000):  X = np.random.uniform(0, 1, size=[n\_samples, noise\_dim])  y = np.zeros((n\_samples, 2))  y[:, 1] = 1  return X, y  def train(GAN, G, D, epochs=500, n\_samples=10000, noise\_dim=10, batch\_size=32, verbose=False, v\_freq=50):  d\_loss = []  g\_loss = []  e\_range = range(epochs)  if verbose:  e\_range = tqdm(e\_range)  for epoch in e\_range:  X, y = sample\_data\_and\_gen(G, n\_samples=n\_samples, noise\_dim=noise\_dim)  set\_trainability(D, True)  d\_loss.append(D.train\_on\_batch(X, y))    X, y = sample\_noise(G, n\_samples=n\_samples, noise\_dim=noise\_dim)  set\_trainability(D, False)  g\_loss.append(GAN.train\_on\_batch(X, y))  if verbose and (epoch + 1) % v\_freq == 0:  print("Epoch #{}: Generative Loss: {}, Discriminative Loss: {}".format(epoch + 1, g\_loss[-1], d\_loss[-1]))  return d\_loss, g\_loss  d\_loss, g\_loss = train(GAN, G, D, verbose=True) |

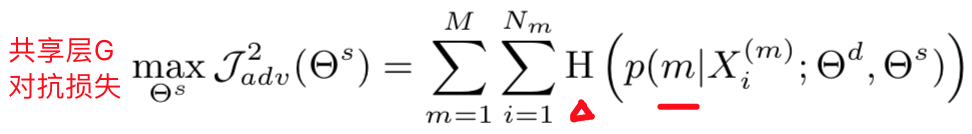
## 训练的trick

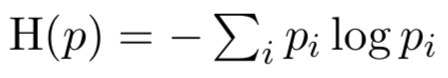
1. 尝试在判别器和生成器模型上引入 dropout
2. 判别器的损失函数迅速收敛为零，导致发生器无法训练。解决：不要对判别器进行预训练。而是调整学习率，使判别器的学习率大于对抗模型的学习率。
3. 针对不同的任务就可以设计不同类型的损失函数。
4. 采用交替优化的方法: 先固定生成器 G, 优化判别器 D, 使得 D 的判别准确率最大化; 然后固定判别器 D, 优化生成器 G, 使得 D 的判别准确率最小化. 当且仅当 pdata = pg 时达到全局最优解. 训练 GAN 时, 同一轮参数更新中, 一般对 D 的参数更新 k 次再对 G 的参数更新 1 次.

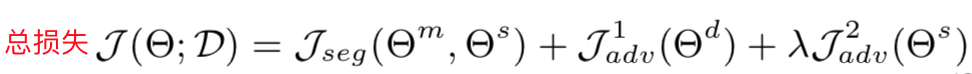
ACL2017对抗中分词（最大化下列损失）

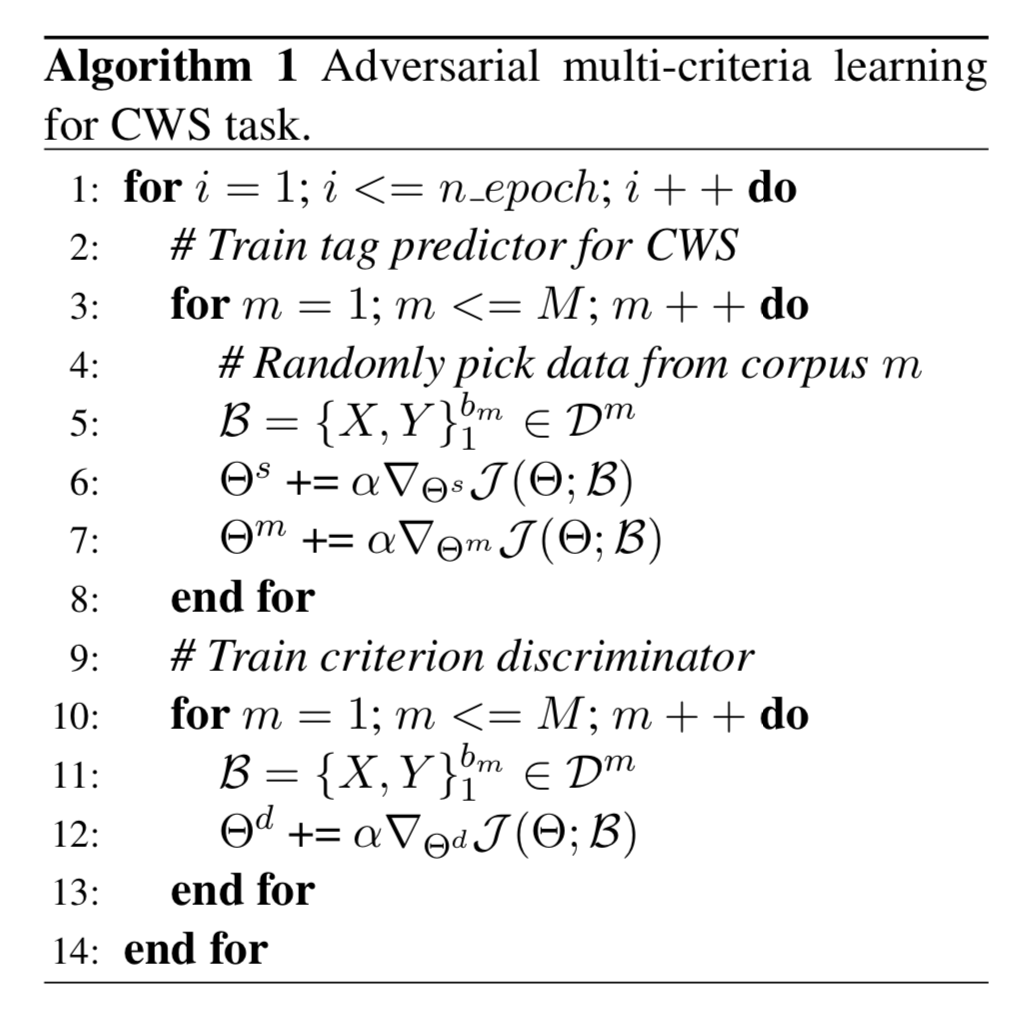
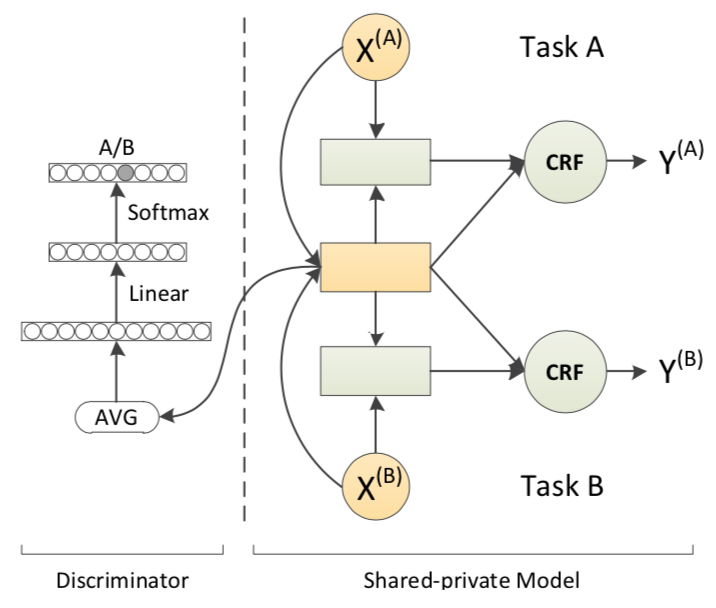




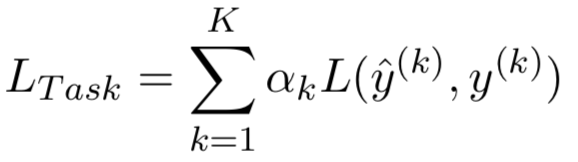


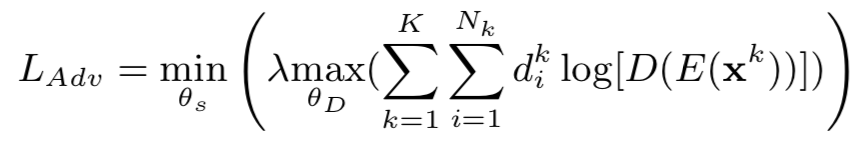




ACL2017对抗多任务的文本分类





EMNLP2017对抗词性POS标注

