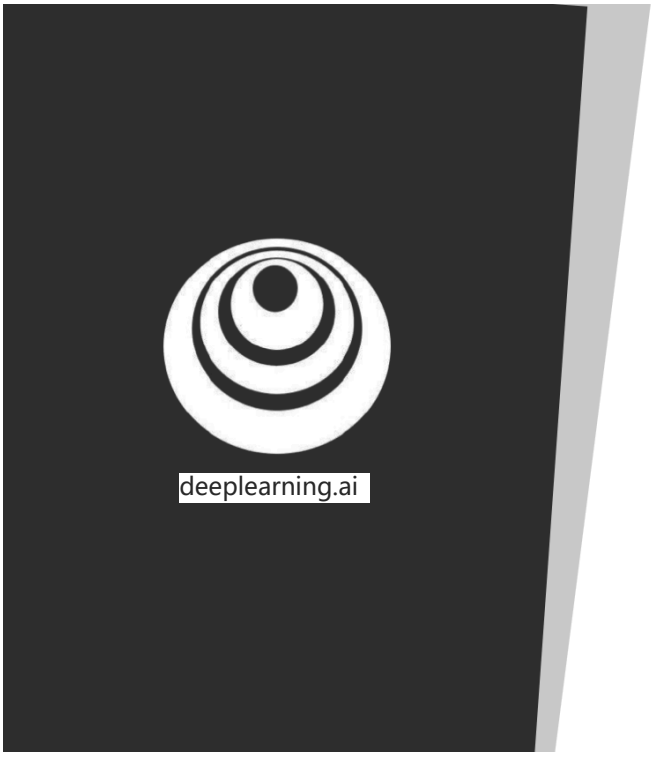
版权声明



生成模型

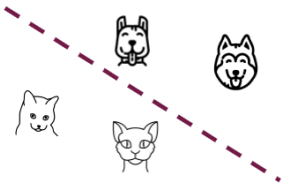
大纲

- 什么是生成模型
- 生成模型的类型

deeplearning.ai

生成模型与判别模型

判别模型



特征 类

$$X \rightarrow Y$$

$$P(Y|X)$$

生成模型

噪声 类 特征

$$\xi, Y \rightarrow X$$

$$P(X|Y)$$

deeplearning.ai

生成模型与判别模型

生成模型

噪声 类 特征

$$\xi, Y \rightarrow X$$

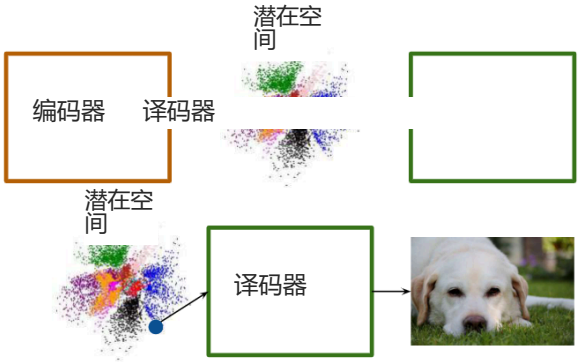
$$P(X|Y)$$

deeplearning.ai

生成模型

变分自动编码器

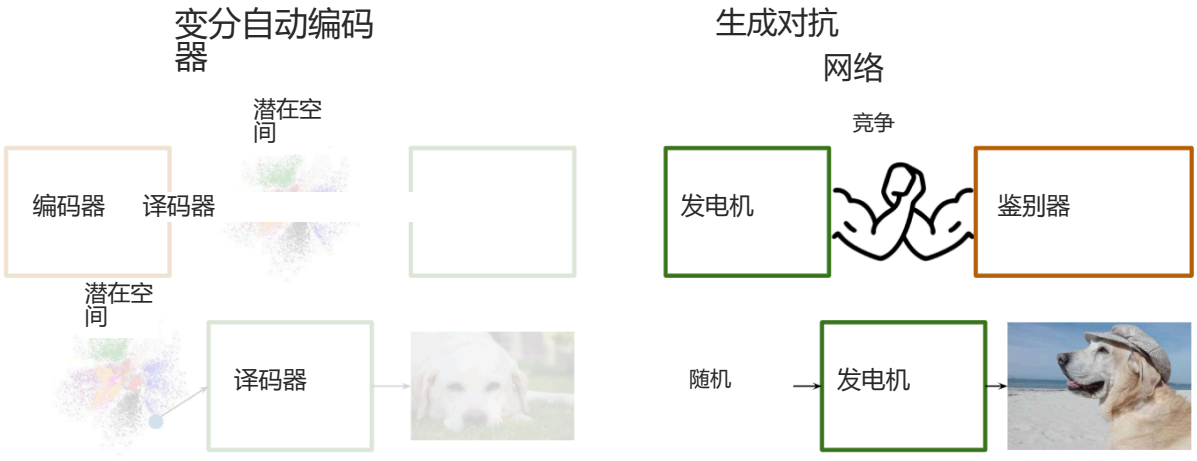
生成对抗网络



可从： https://arxiv.org/abs/1804.00891

deeplearning.ai

生成模型



可从： https://arxiv.org/abs/1804.00891

deeplearning.ai

总结

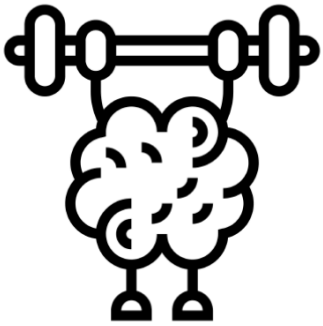
- 生成模型学习生成示例
- 判别模型区分类
- 接下来，GAN!

deeplearning.ai

现实生活中的 GAN

大纲

- GAN 的酷应用
- 使用它们的大公司



GANs 随时间的变化

2014 黑色和白色

↓

2018 彩色照片写实

deeplearning.ai

GANs 随时间的变化

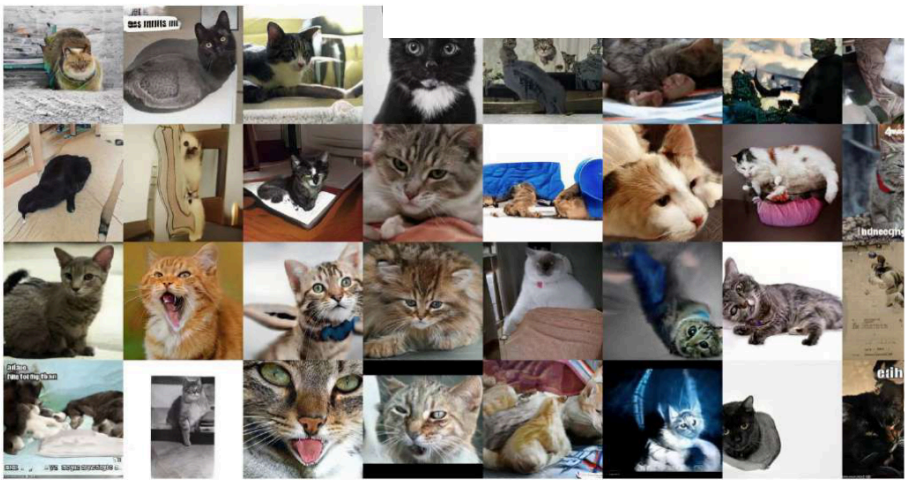
人脸生成样式 GAN2

这些人会不存在!

Karras， Tero， et al. “分析和提高 stylegan 的图像质量。” 计算机视觉和模式识别 IEEE/CVF 会议论文集。2020.

deeplearning.ai

GANs 随时间的变化 样式GAN2



模拟训练数据的分布

Karras， Tero， et al. “分析和提高 stylegan 的图像质量。” 计算机视觉和模式识别 IEEE/CVF 会议论文集。2020. https://9gag.com/gag/aWYZKWx

deeplearning.ai

用于图像翻译的 GAN

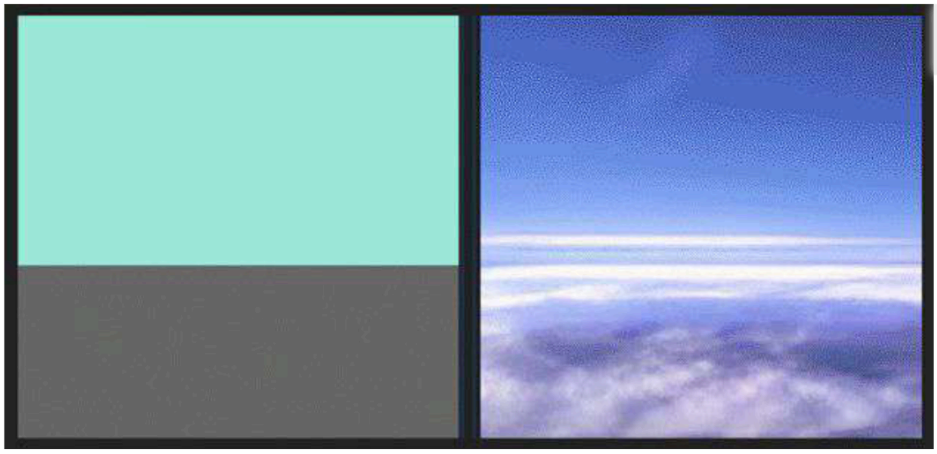
从一个域到另一个域

循环甘

Park， Taesung， et al. “具有空间自适应归一化的语义图像合成。” IEEE 计算机视觉和模式识别会议论文集。2019.

deeplearning.ai

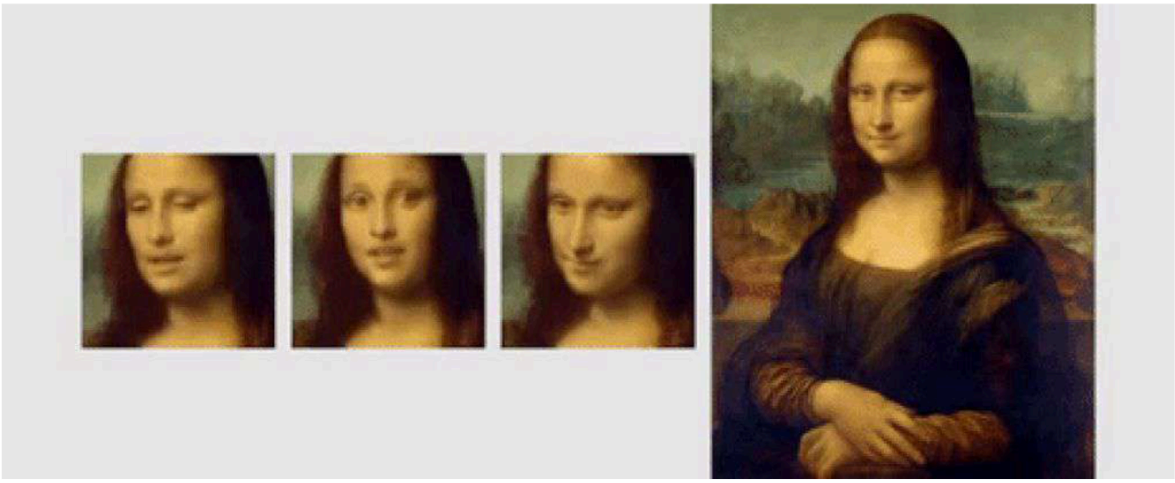
用于图像翻译的 GAN



Park， Taesung， et al. “具有空间自适应归一化的语义图像合成。” IEEE 计算机视觉和模式识别会议论文集。2019.

deeplearning.ai

GAN 就是魔法!



Zakharov， Egor， et al. “现实神经说话头模型的少镜头对抗性学习。” IEEE 计算机视觉国际会议论文集。2019.

deeplearning.ai

用于 3D 对象的 GAN

3D-GAN

生成
设计

Wu， Jiajun， et al. “通过 3D 生成-对抗建模学习对象形状的概率潜在空间。” 神经信息处理系统的进步。2016.

使用 GAN 的公司

Adobe 下一代 Photoshop

Google 文本生成

IBM 数据增强

图像过滤器

Disney 超分辨率

deeplearning.ai

总结

- GAN 的性能正在迅速提高
- 在这个领域工作的巨大机会!
- 大公司正在使用它们

deeplearning.ai

deeplearning.ai

GAN 背后的直觉

大纲

- 生成器和判别器的目标
- 他们之间的竞争

deeplearning.ai

生成对抗网络

Generator 学会造假
那个看起来真的

判别器学会区分
真假

deeplearning.ai

生成对抗网络

Generator 学会造假
那个看起来真的

假

真正

判别器学会区分
真假

deeplearning.ai

生成对抗网络

Generator 学会造假
那个看起来真的

判别器学会区分
真假

deeplearning.ai

生成对抗网络

Generator 学会造假
那个看起来真的

不知道怎么做
它应该看起来

deeplearning.ai

生成对抗网络

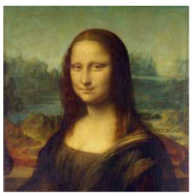
判别器学会区分
真假

deeplearning.ai

游戏开始了！

5% 真实

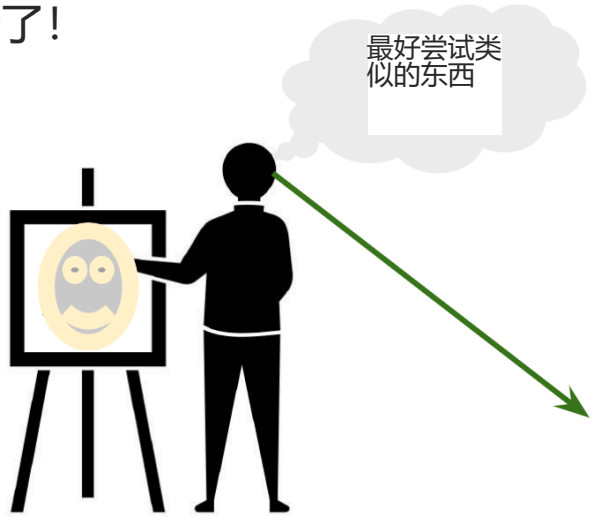
40% 真实



80% 真实

deeplearning.ai

游戏开始了！



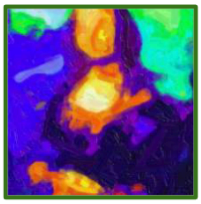
5% 真实

40% 真实

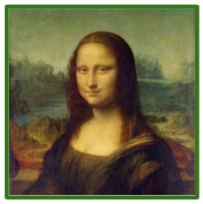
deeplearning.ai

游戏开始了！

30% 真实



60% 真实

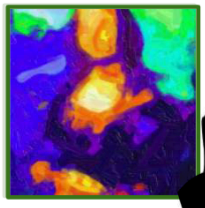


95% 真实

deeplearning.ai

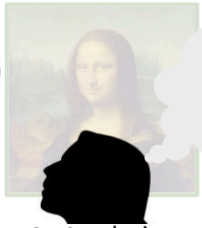
游戏开始了！

30% 真实



60% 真实

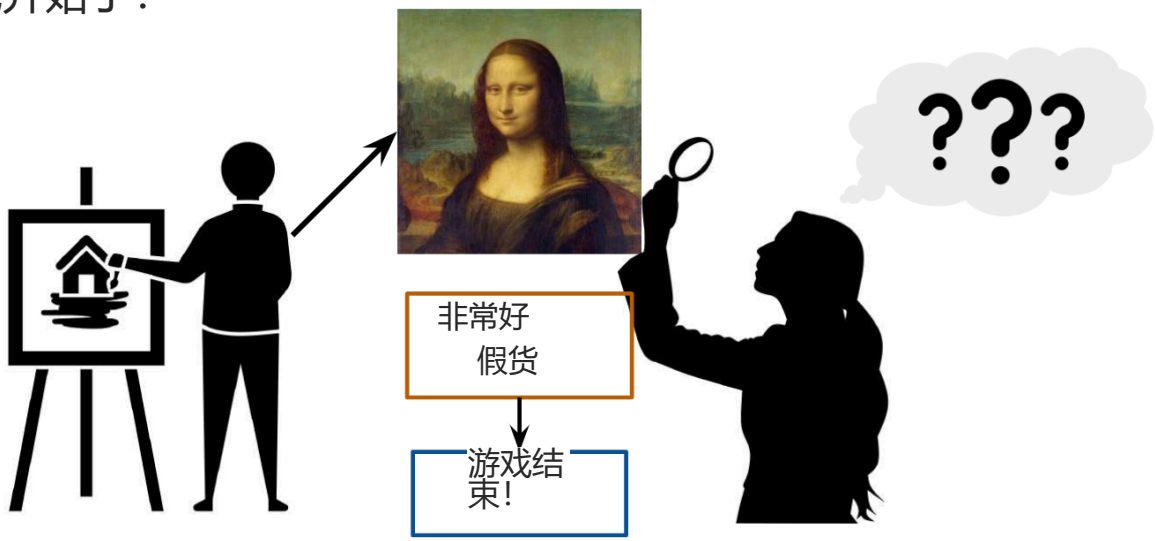
错！



95% 真实

deeplearning.ai

游戏开始了！



deeplearning.ai

总结

- 生成器的目标是欺骗判别器
- 判别器的目标是区分真假
- 他们从竞争中相互学习
- 到头来，假货看起来是真的

deeplearning.ai

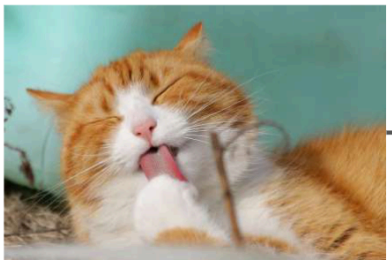
鉴别器

大纲

- 分类器回顾
- 分类器在概率方面的作用
- 鉴别器

分类

区分不同的类



分类

海龟

鸟

猫

狗

鱼

deeplearning.ai

分类

区分不同的类

“它喵喵叫，嬉戏纱线”

分类

海龟

鸟

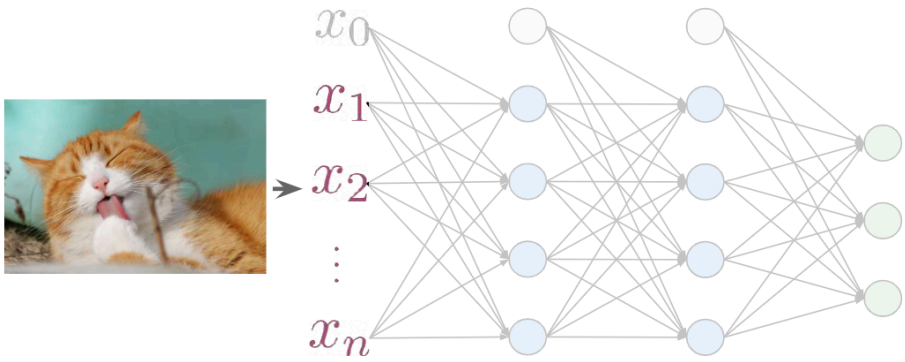
猫

狗

鱼

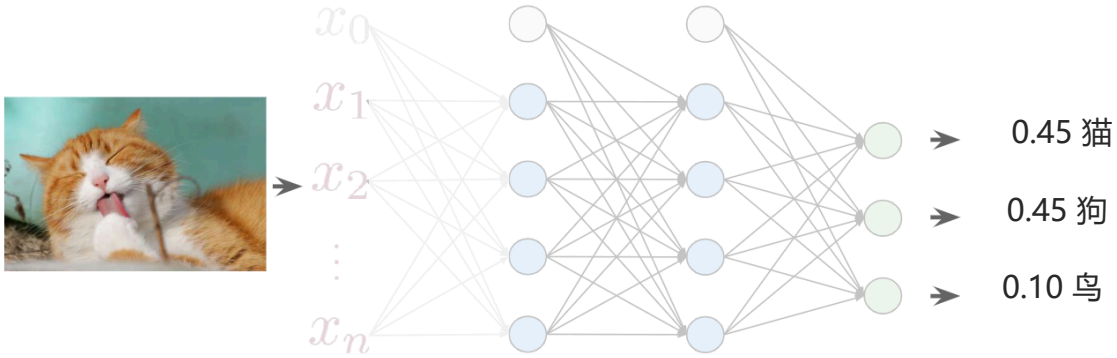
deeplearning.ai

神经网络



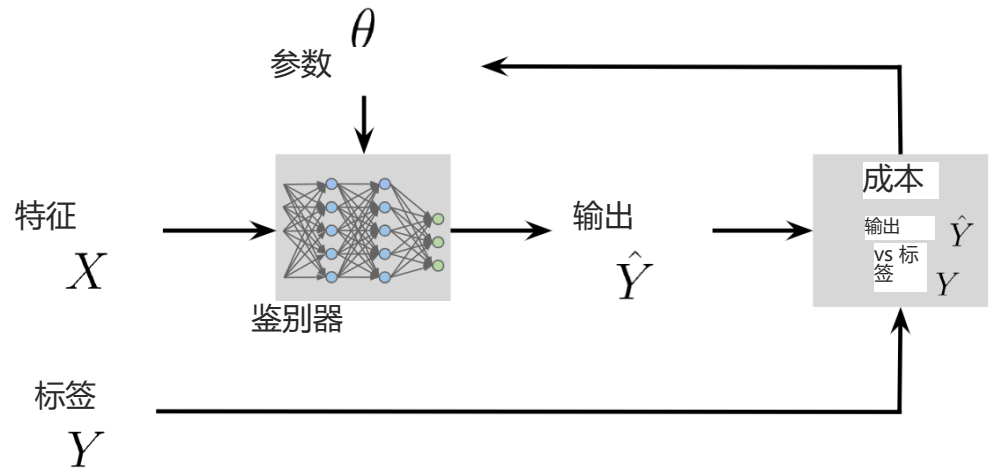
deeplearning.ai

神经网络



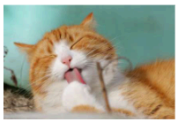
deeplearning.ai

分类器（训练）



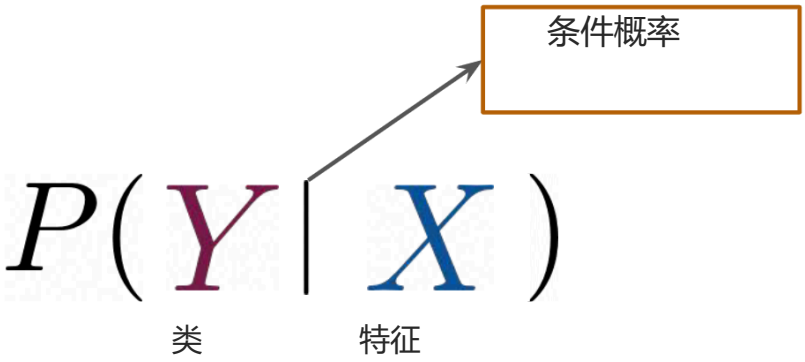
deeplearning.ai

分类

$$P(\text{海龟}, \text{鸟}, \text{猫}, \text{狗}, \text{鱼} \mid \text{  })$$

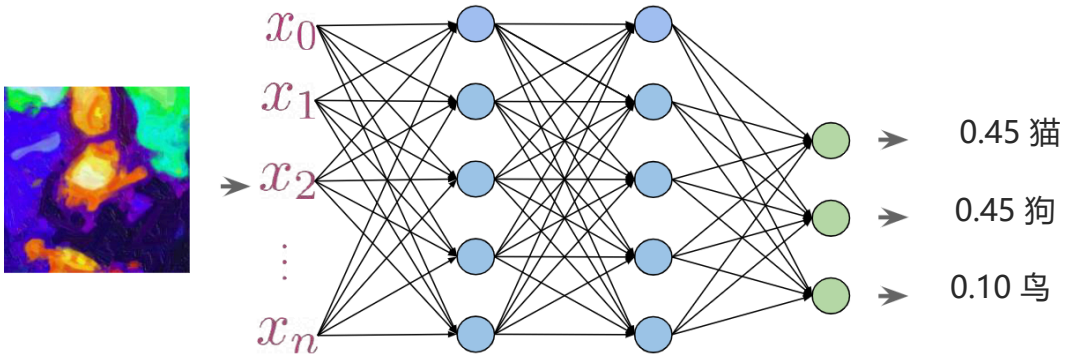
deeplearning.ai

分类



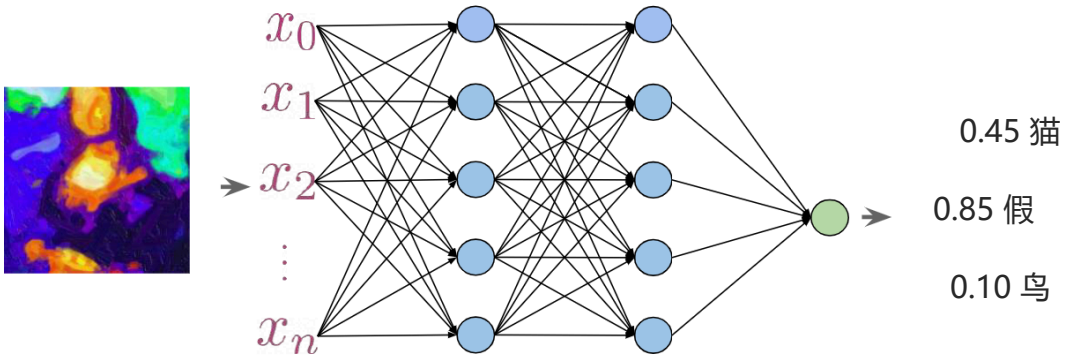
deeplearning.ai

鉴别器



deeplearning.ai

鉴别器



deeplearning.ai

鉴别器

$$P(\text{假} \mid X)$$

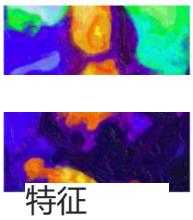
类　　特征

deeplearning.ai

鉴别器

$$P(\text{假} \mid \text{特征}) = 0.85$$

假类



deeplearning.ai

总结

- 鉴别器是一个分类器
- 它学习给定特征 X 的 Y 类（真实或虚假）的概率
- 概率是生成器的反馈

deeplearning.ai

发电机

大纲

- 生成器的作用
- 如何提高性能
- 概率方面的生成器

发电机

海龟

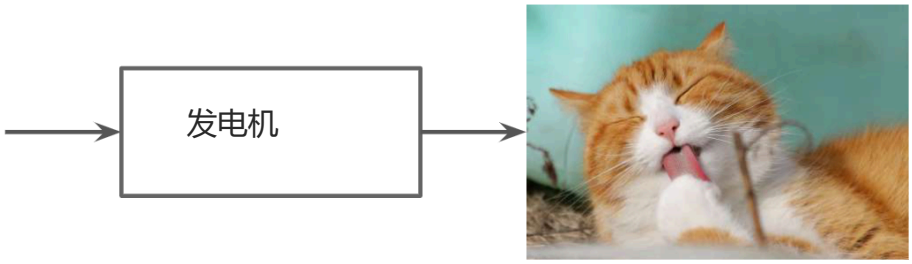
鸟

猫

狗

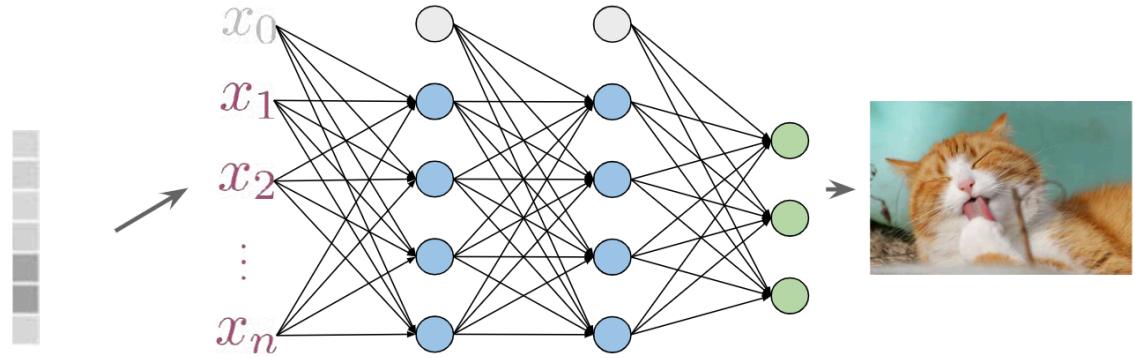
鱼

生成类的示例



deeplearning.ai

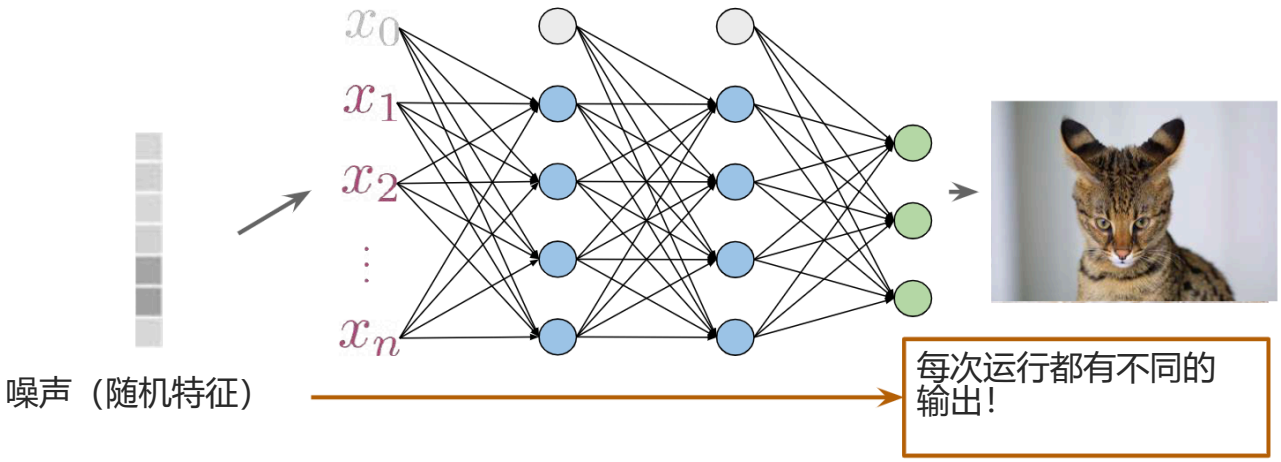
神经网络



噪声（随机特征）

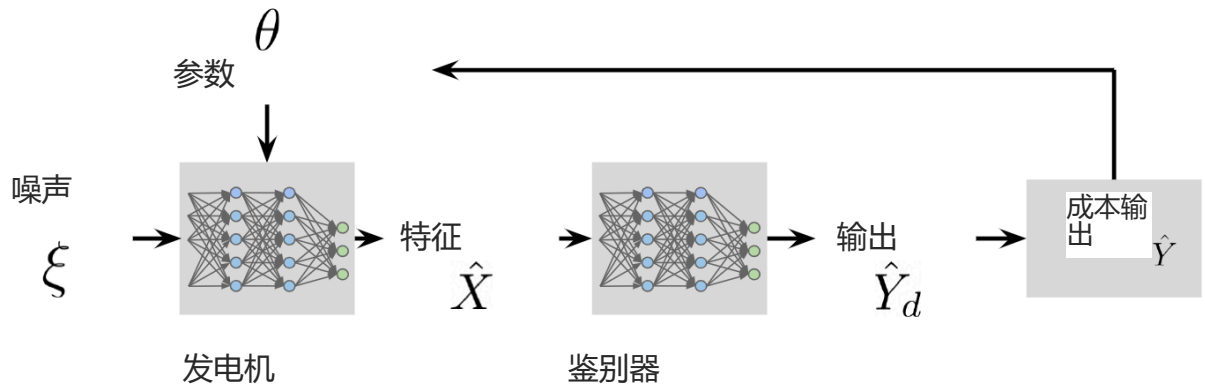
deeplearning.ai

神经网络



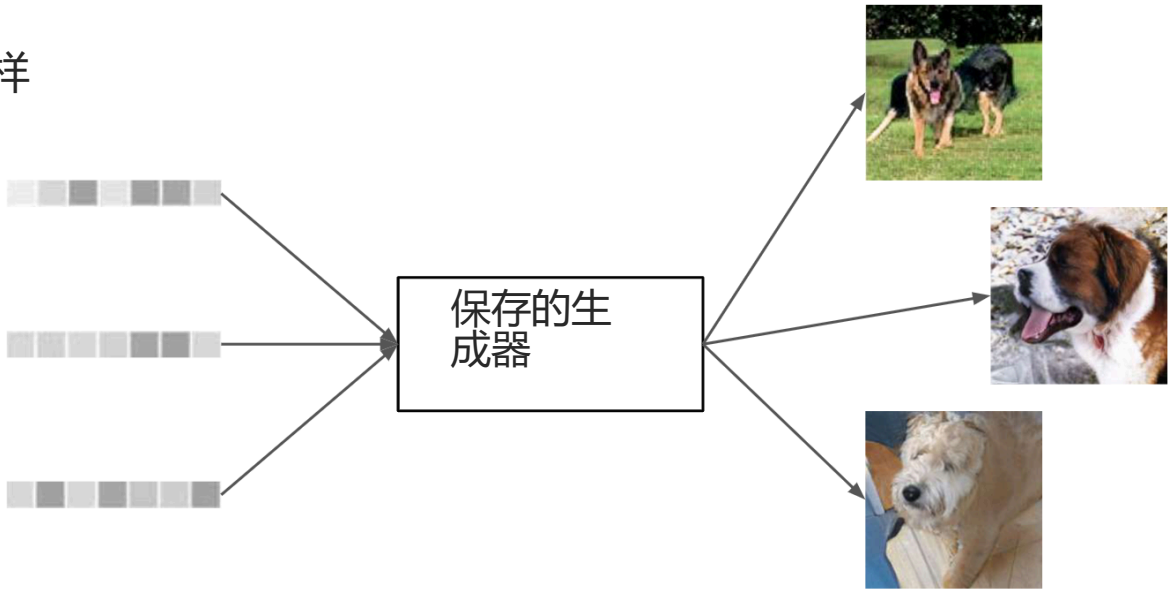
deeplearning.ai

生成器：学习



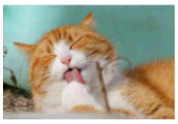
deeplearning.ai

采样



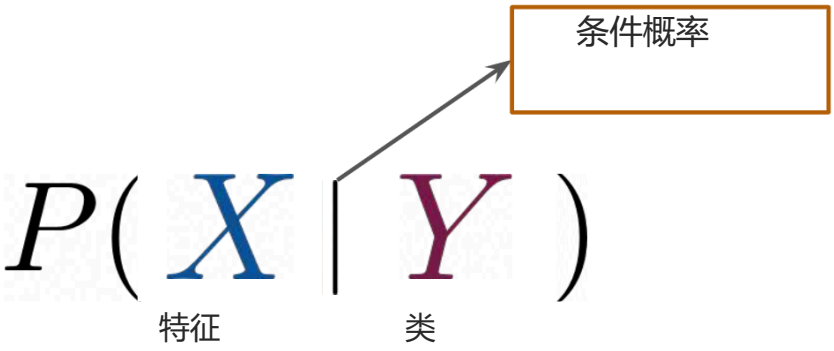
deeplearning.ai

发电机

$$P(\ \text{  }\ \mid \text{海龟 鸟 猫 狗 鱼})$$

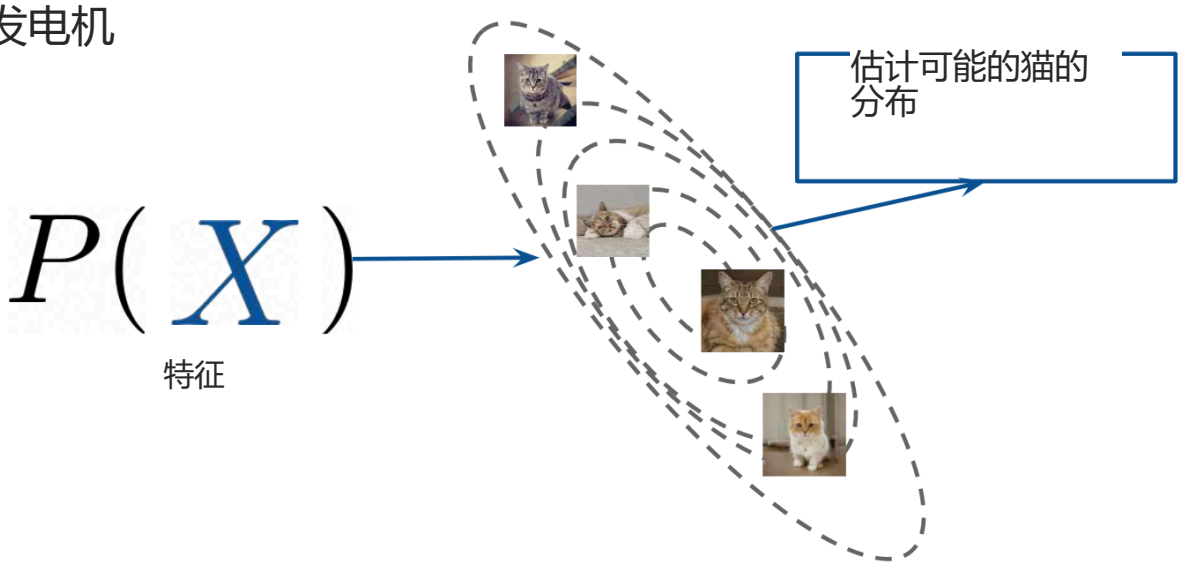
deeplearning.ai

发电机



deeplearning.ai

发电机



图片来自：http://thesecatsdonotexist.com/

deeplearning.ai

总结

- 生成器产生虚假数据
- 它学习特征 X 的概率
- 生成器将输入噪声（随机特征）

deeplearning.ai

deeplearning.ai

BCE 成本函数

大纲

- 二进制交叉熵 （BCE） 损失方程各部分
- 图形外观

deeplearning.ai

BCE 成本函数

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h(x^{(i)},\theta) + (1-y^{(i)})\log(1-h(x^{(i)},\theta))\right]$$

预测 标签 特征 参数

整批平均损失

deeplearning.ai

BCE 成本函数

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h(x^{(i)},\theta) + (1-y^{(i)})\log(1-h(x^{(i)},\theta))\right]$$

deeplearning.ai

BCE 成本函数

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h(x^{(i)},\theta) + (1-y^{(i)})\log(1-h(x^{(i)},\theta))\right]$$

$y^{(i)}$	$h(x^{(i)},\theta)$	$y^{(i)}\log h(x^{(i)},\theta)$
0	任何	0
1	0.99	~0
1	~0	-inf

当标签为 1 时相关

deeplearning.ai

BCE 成本函数

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h(x^{(i)},\theta) + (1-y^{(i)})\log(1-h(x^{(i)},\theta))\right]$$

$y^{(i)}$	$h(x^{(i)},\theta)$	$(1-y^{(i)})\log(1-h(x^{(i)},\theta))$
1	任何	0
0	0.01	~0
0	~1	-inf

当标签为 0 时相关

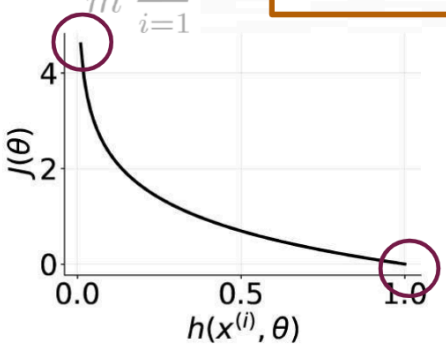
BCE 成本函数

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h(x^{(i)},\theta) + (1-y^{(i)})\log(1-h(x^{(i)},\theta))\right]$$

确保成本始终大于或等于 0

BCE 成本函数

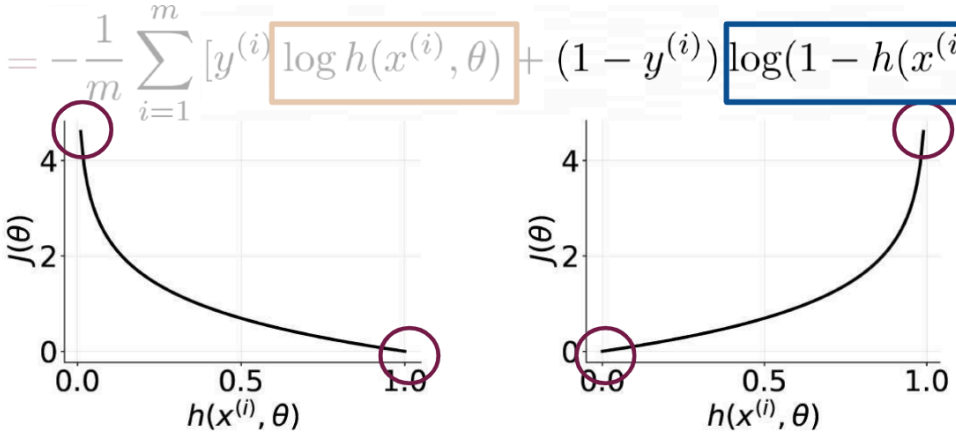
$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log h(x^{(i)},\theta) + (1-y^{(i)})\log(1-h(x^{(i)},\theta))]$$



deeplearning.ai

BCE 成本函数

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log h(x^{(i)},\theta) + (1-y^{(i)})\log(1-h(x^{(i)},\theta))]$$

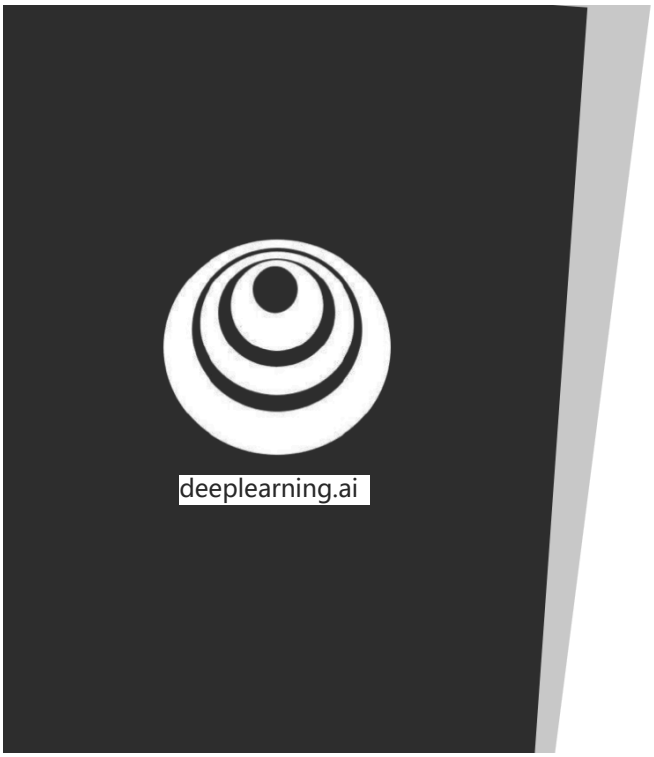


deeplearning.ai

总结

- BCE 成本函数有两个部分（每个类一个相关）
- 当标签和预测相似时，接近零
- 当标签和预测不同时接近无穷大

deeplearning.ai

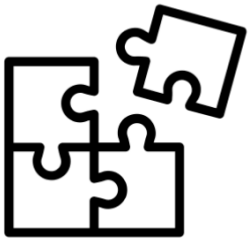


把所有东西都放进去

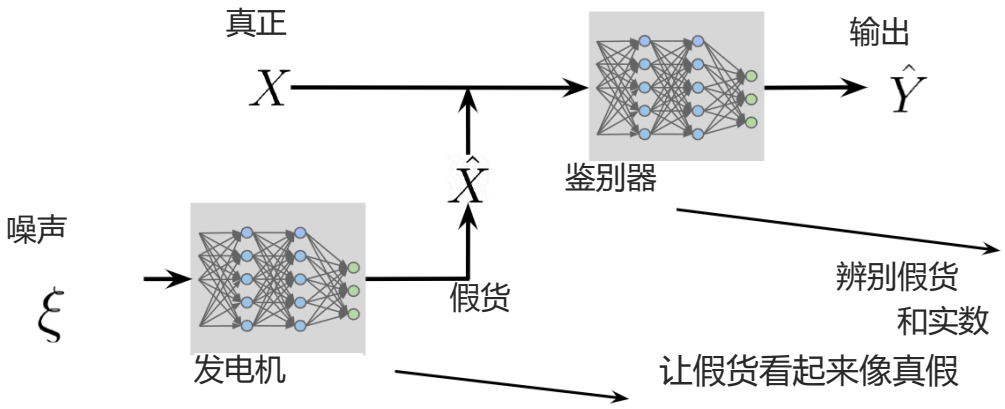
一起

大纲

- 整个架构的外观
- 如何训练 GAN

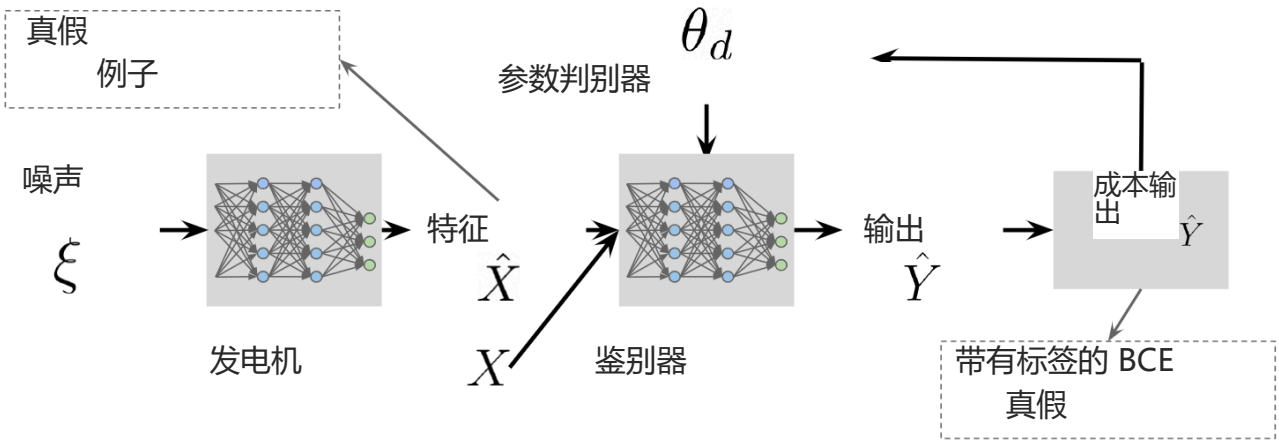


GAN 模型



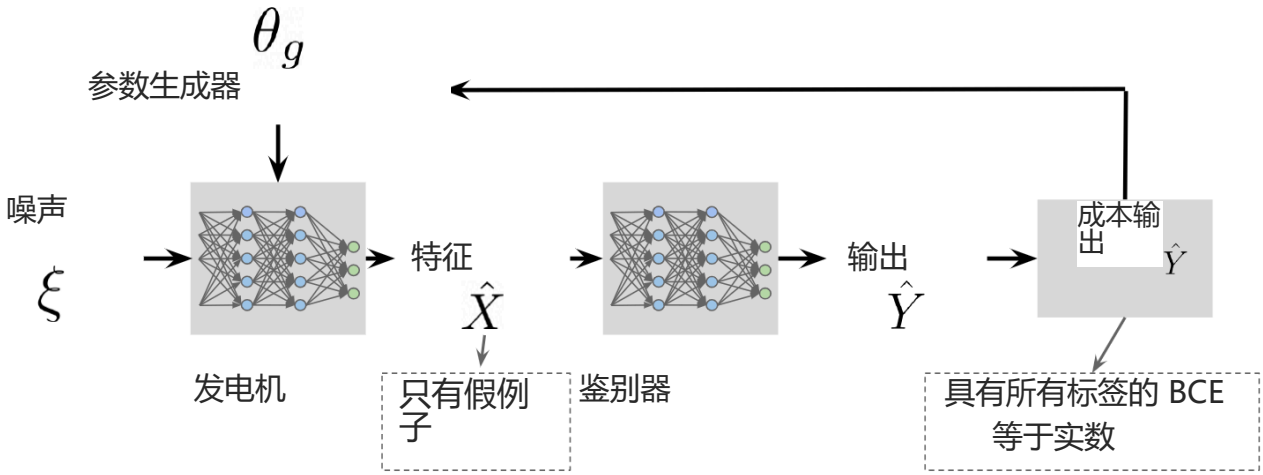
deeplearning.ai

训练 GAN：判别器



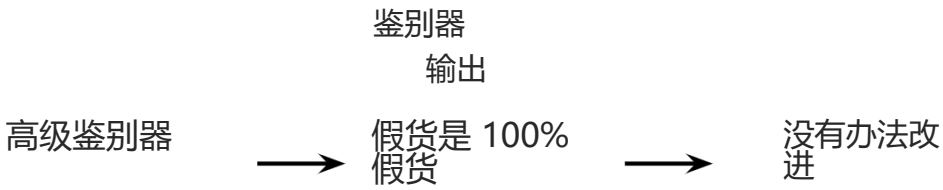
deeplearning.ai

训练 GAN：生成器



deeplearning.ai

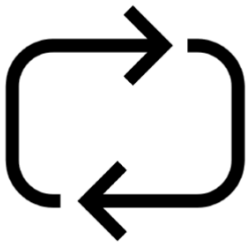
训练 GAN



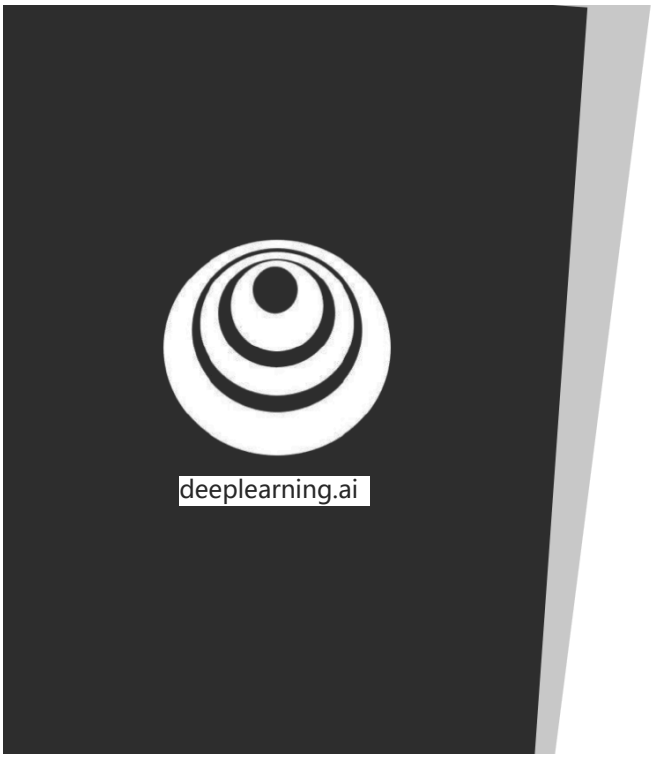
deeplearning.ai

总结

- GAN 以交替的方式训练
- 这两个模型应始终处于相似的“技能”级别



deeplearning.ai

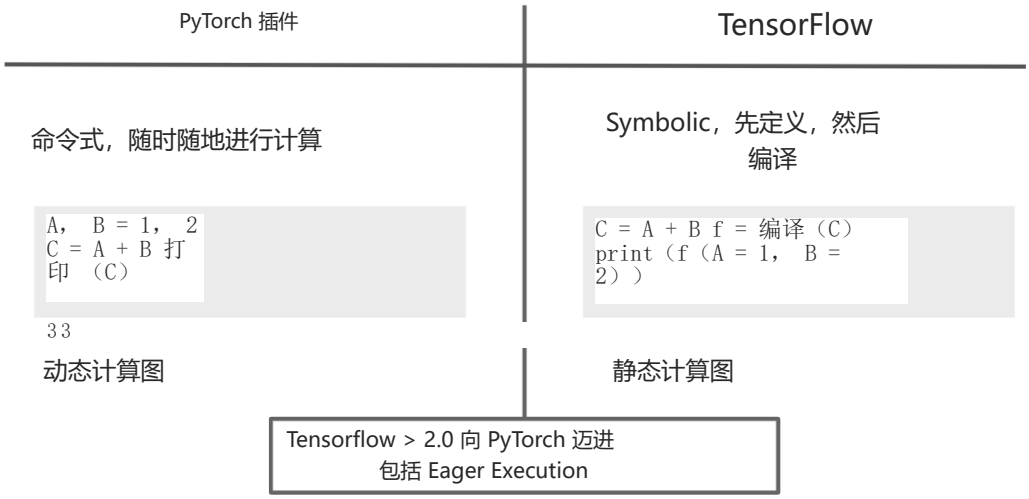


PyTorch 简介（可选）

大纲

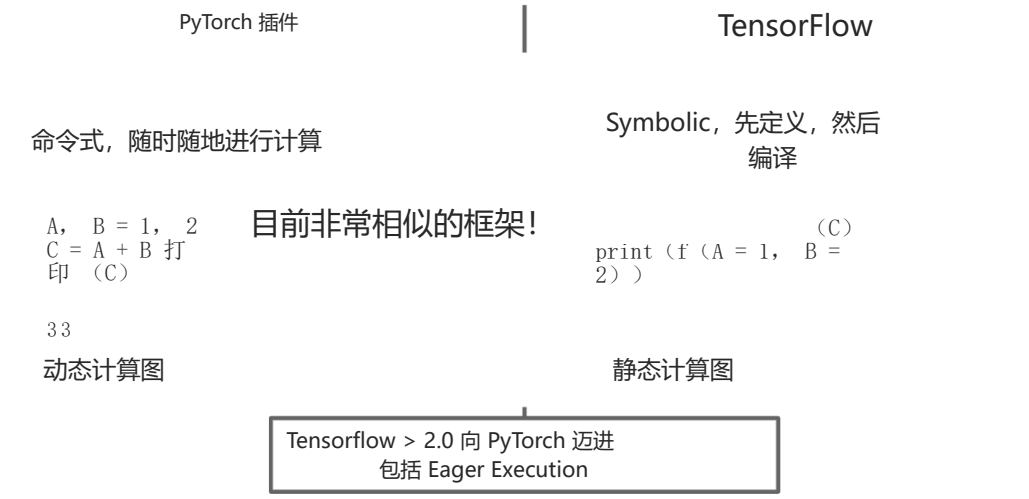
- 与 TensorFlow 的比较
- 定义模型
- 训练

PyTorch 与 TensorFlow



deeplearning.ai

PyTorch 与 TensorFlow



deeplearning.ai

在 PyTorch 中定义模型

```
从 Torch Import NN 导入
Torch
```

→ DL 的自定义图层

```
class LogisticRegression（nn.模块）：
      def __init__（self， in）：
      super（）.__init__（）
            self.log_reg = nn.顺序 （
                  nn.线性（in，      1），
            nn.Sigmoid（）  ）

      def forward（self， x）：返回
      self.log_reg（x）
```

将模型定义为类

带参数的初始化方法

架构的定义

输入 x 的模型的正向计算

deeplearning.ai

PyTorch 中的训练模型

```
模型 = LogisticRegression（16）
```

模型的初始化

```
标准 = nn。BCELoss（）
```

成本函数

```
优化器 = torch.optim.SGD（model.parameters（），  lr=0.01）
```

优化

```
对于 range （n_epochs） 中的 t：
      y_pred = 模型 （x） 损失 = 标准
      （y_pred， y）

      optimizer.zero_grad（）
      loss.backward（）
      optimizer.step（）
```

epoch 数的训练循环

前向传播

优化步骤

deeplearning.ai

总结

- PyTorch 在运行时进行计算
- Pytorch 中的动态计算图
- 只是另一个框架，类似于 Tensorflow!