

# 生成网络

鲁 鹏

北京邮电大学 计算机学院 智能科学与技术中心

本章教学课件参考了斯坦福大学CS231N以及台湾大学机器学习课件，  
感谢这两门课程的团队在课程建设方面所做的工作！

## 今日主题

- 无监督学习
- 产生式模型
  - PixelRNN and PixelCNN
  - Variational Autoencoders (VAE)
  - Generative Adversarial Networks (GAN)

# 有监督学习 vs 无监督学习

## 有监督学习

数据:  $(x, y)$  , 其中 $x$ 表示样本,  $y$ 表示标签

目标: 学习  $x \rightarrow y$ 的映射

例子: 分类, 回归, 目标检测, 语义分割等等



→ 猫

分类

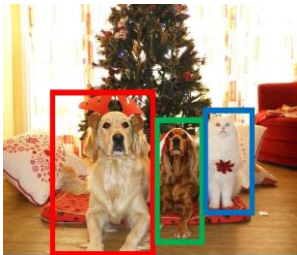
# 有监督学习 vs 无监督学习

## 有监督学习

数据:  $(x, y)$  , 其中 $x$ 表示样本,  $y$ 表示标签

目标: 学习  $x \rightarrow y$ 的映射

例子: 分类, 回归, 目标检测, 语义分割等等



狗, 狗, 猫

目标检测

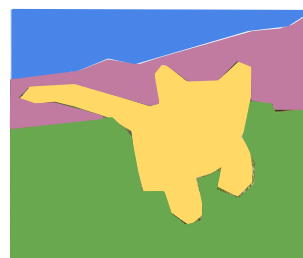
# 有监督学习 vs 无监督学习

## 有监督学习

**数据:**  $(x, y)$  , 其中 $x$ 表示样本,  $y$ 表示标签

**目标:** 学习  $x \rightarrow y$  的映射

**例子:** 分类, 回归, 目标检测, 语义分割等等



草地, 猫, 树,  
天空

语义分割

# 有监督学习 vs 无监督学习

## 无监督学习

**数据:**  $x$  , 其中 $x$ 为数据

**目标:** 找出隐含在数据里的模式或者结构

**例子:** 聚类, 降维, 特征学习, 密度估计等

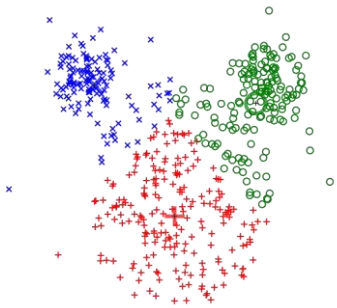
# 有监督学习 vs 无监督学习

## 无监督学习

数据:  $x$  , 其中 $x$ 为数据

目标: 找出隐含在数据里的模式或者结构

例子: 聚类, 降维, 特征学习, 密度估计等



K-means聚类

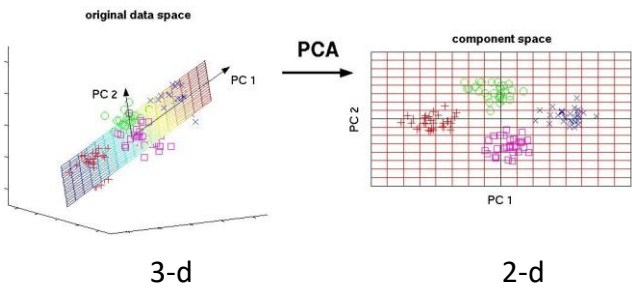
# 有监督学习 vs 无监督学习

## 无监督学习

数据:  $x$  , 其中 $x$ 为数据

目标: 找出隐含在数据里的模式或者结构

例子: 聚类, 降维, 特征学习, 密度估计等



主成分分析 (降维)

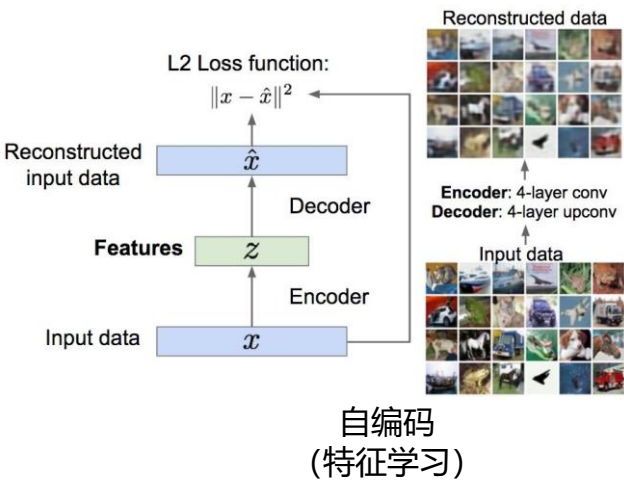
# 有监督学习 vs 无监督学习

## 无监督学习

**数据:**  $x$  , 其中 $x$ 为数据

**目标:** 找出隐含在数据里的模式或者结构

**例子:** 聚类, 降维, 特征学习, 密度估计等



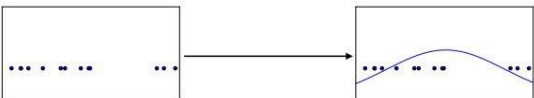
# 有监督学习 vs 无监督学习

## 无监督学习

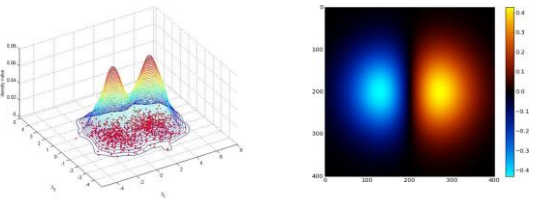
**数据:**  $x$  , 其中 $x$ 为数据

**目标:** 找出隐含在数据里的模式或者结构

**例子:** 聚类, 降维, 特征学习, 密度估计等



1-d density estimation



2-d 密度估计

# 有监督学习 vs 无监督学习

**有监督学习**

**数据:**  $(x, y)$  ,其中 $x$ 表示样本,  $y$ 表示

标签

**目标:** 学习  $x \rightarrow y$ 的映射

**例子:** 分类, 回归, 目标

检测, 语义分割等等

**无监督学习**

**数据:**  $x$  , 其中 $x$ 为数据

**目标:** 找出隐含在数据里的模

式或者结构

**例子:** 聚类, 降维, 特

征学习, 密度估计等

# 有监督学习 vs 无监督学习

**有监督学习**

**数据:**  $(x, y)$  ,其中 $x$ 表示样本,  $y$ 表示

标签

**目标:** 学习  $x \rightarrow y$ 的映射

**例子:** 分类, 回归, 目标

检测, 语义分割等等

**无监督学习**

数据获取成本低

**数据:**  $x$  , 其中 $x$ 为数据

**目标:** 找出隐含在数据里的模

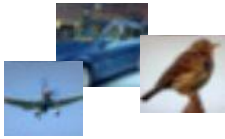
式或者结构

**例子:** 聚类, 降维, 特

征学习, 密度估计等

# 产生式模型

给定训练集，产生与训练集同分布的新样本。



训练数据服从  $p_{\text{data}}(x)$

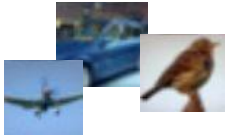


产生样本服从  $p_{\text{model}}(x)$

希望学到一个模型  $p_{\text{model}}(x)$ ，其与训练样本的分布  $p_{\text{data}}(x)$  相近

# 产生式模型

给定训练集，产生与训练集同分布的新样本。



训练数据服从  $p_{\text{data}}(x)$



产生样本服从  $p_{\text{model}}(x)$

希望学到一个模型  $p_{\text{model}}(x)$ ，其与训练样本的分布  $p_{\text{data}}(x)$  相近

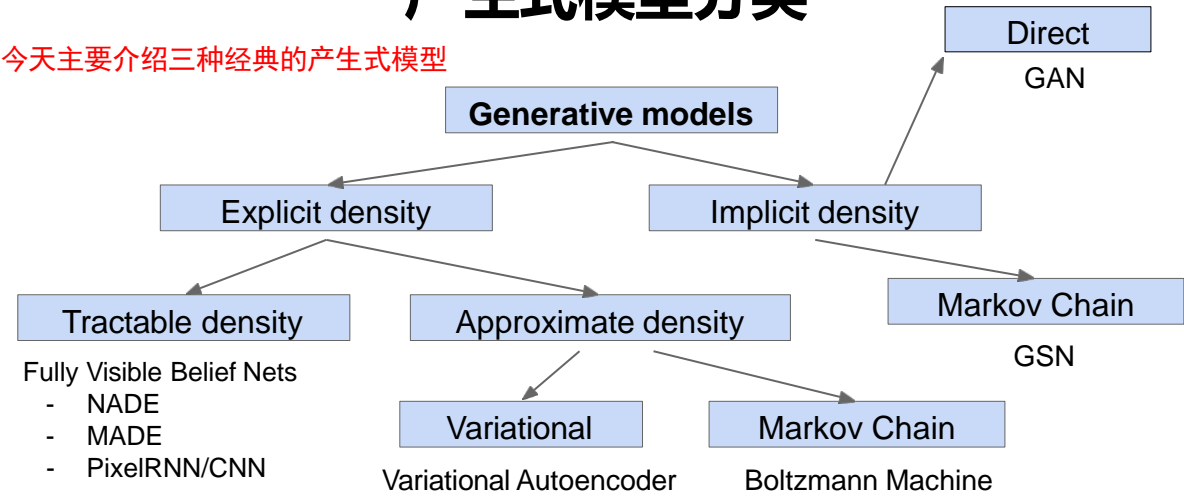
➤ 无监督学习里的一个核心问题——密度估计问题

几种典型思路：

- 显式的密度估计：显式的定义并求解分布  $p_{\text{model}}(x)$
- 隐式的密度估计：学习一个模型  $p_{\text{model}}(x)$ ，而无需显式的定义它

# 产生式模型分类

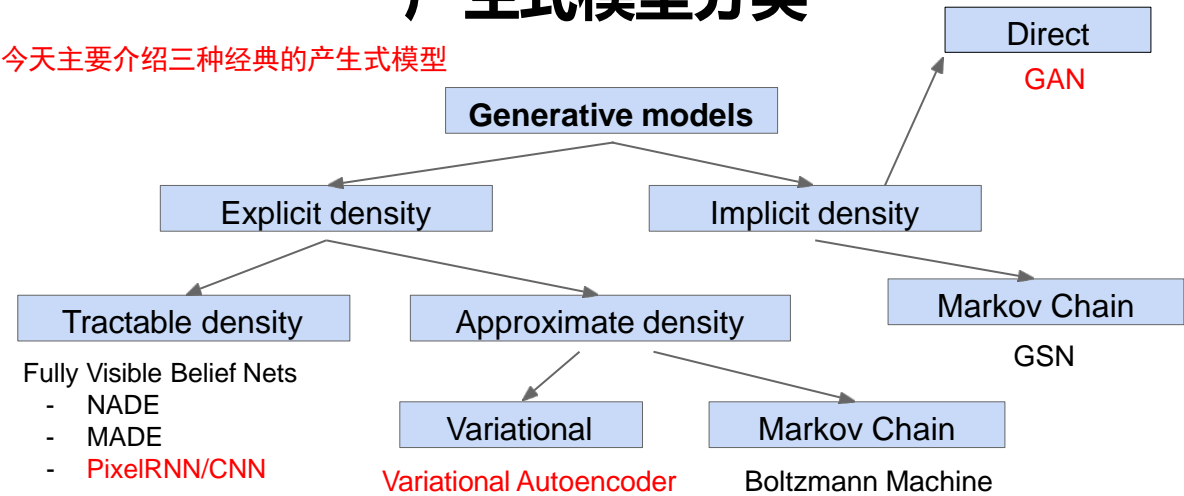
今天主要介绍三种经典的产生式模型



Tutorial on Generative Adversarial Networks, 2017.

# 产生式模型分类

今天主要介绍三种经典的产生式模型



Tutorial on Generative Adversarial Networks, 2017.



# 今日主题

- 无监督学习
- 产生式模型
  - PixelRNN and PixelCNN
  - Variational Autoencoders (VAE)
  - Generative Adversarial Networks (GAN)

2020/6/2

北京邮电大学计算机学院 鲁鹏

16

## PixelRNN 与 PixelCNN

显式的密度模型

利用链式准则将图像  $x$  的生成概率转变为每个像素生成概率的乘积：

$$p(x) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1})$$

$\uparrow$   
 图像  $x$  的似然

$\uparrow$   
 给定已经生成的像素的前  
 提下生成第  $i$  个像素的概率

最大化训练数据的似然！

2020/6/2

北京邮电大学计算机学院 鲁鹏

17

# PixelRNN 与 PixelCNN

显式的密度模型

利用链式准则将图像  $x$  的生成概率转变为每个像素生成概率的乘积：

$$p(x) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1})$$

↑

图像  $x$  的似然

↑

给定已经生成的像素的前提下生成第  $i$  个像素的概率

需要定义像素的产生顺序

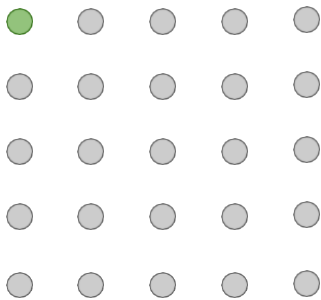
最大化训练数据的似然！

这个分布会很复杂，但是可以使用神经网络来建模！

## PixelRNN

从图像左上角开始产生像素

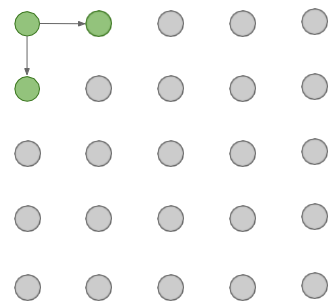
将像素生成看成一个序列生成的问题，利用 RNN（LSTM）的序列描述能力来生成新的像素。



# PixelRNN

从图像左上角开始产生像素

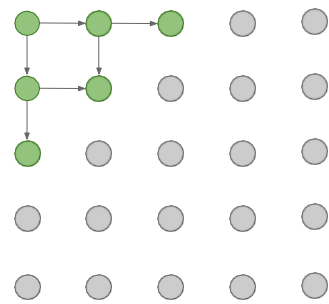
将像素生成看成一个序列生成的问题，利用 RNN（LSTM）的序列描述能力来生成新的像素。



# PixelRNN

从图像左上角开始产生像素

将像素生成看成一个序列生成的问题，利用 RNN（LSTM）的序列描述能力来生成新的像素。

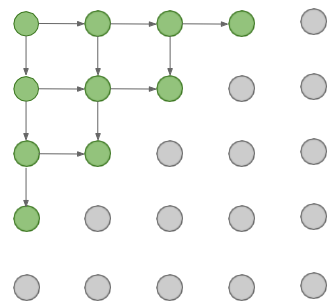


# PixelRNN

从图像左上角开始产生像素

将像素生成看成一个序列生成的问题，利用 RNN（LSTM）的序列描述能力来生成新的像素。

缺陷: 序列生成整张图片太慢了!

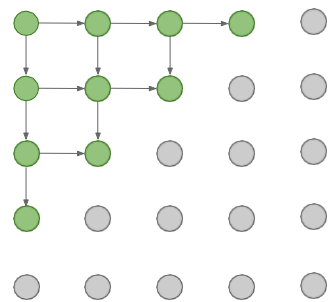


# PixelRNN [van der Oord et al. 2016]

从图像左上角开始产生像素

将像素生成看成一个序列生成的问题，利用 RNN（LSTM）的序列描述能力来生成新的像素。

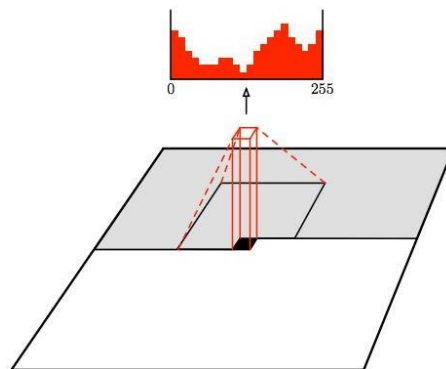
缺陷: 序列生成整张图片太慢了!



## PixelCNN [van der Oord et al. 2016]

依然是从图像左上角开始产生像素

基于已生成的像素，利用CNN来生成新的像素



2020/6/2

北京邮电大学计算机学院 鲁鹏

24

## PixelCNN [van der Oord et al. 2016]

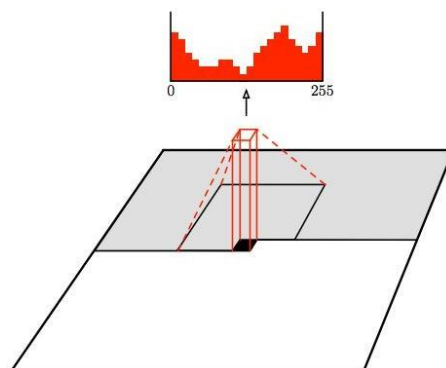
依然是从图像左上角开始产生像素

基于已生成的像素，利用CNN来生成新的像素

训练：最大化训练数据的似然

$$p(x) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1})$$

每个像素点使用Softmax loss



2020/6/2

北京邮电大学计算机学院 鲁鹏

25

# PixelCNN [van der Oord et al. 2016]

依然是从图像左上角开始产生像素

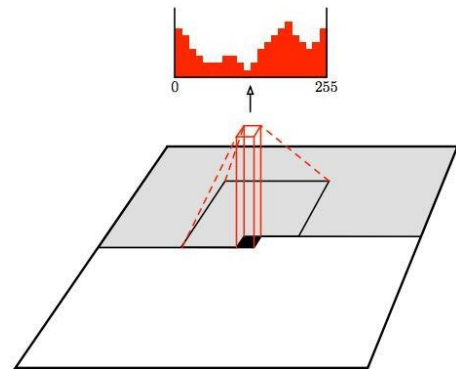
基于已生成的像素，利用CNN来生成新的像素

相对于PixelRNN，PixelCNN训练更快  
(由于上下文信息已知，在训练时可以并行卷积)

图像的产生过程还是逐像素的序列生成

=> 依然很慢

每个像素点使用Softmax loss

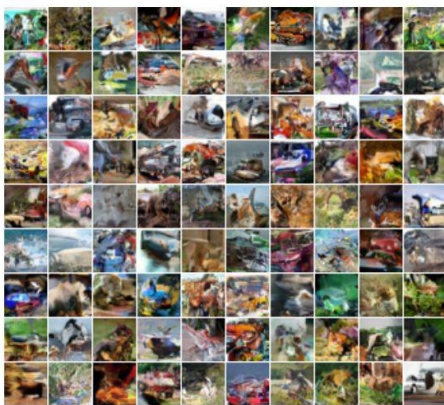


2020/6/2

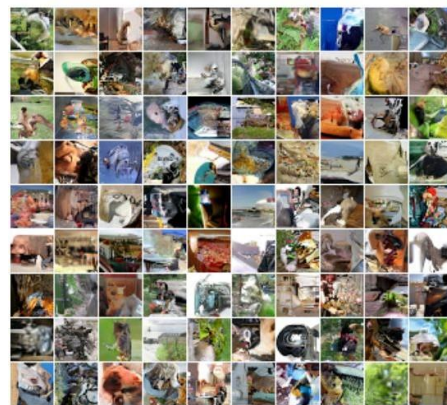
北京邮电大学计算机学院 鲁鹏

26

## 模型生成结果展示



32x32 CIFAR-10



32x32 ImageNet

Figures copyright Aaron van der Oord et al., 2016. Reproduced with permission.

2020/6/2

北京邮电大学计算机学院 鲁鹏

27

# PixelRNN 与 PixelCNN

优点:

- 似然函数可以精确计算
- 利用似然函数的值可以有效地评估模型性能

缺点:

- 序列产生 => 慢

2020/6/2

北京邮电大学计算机学院 鲁鹏

28

## 今日主题

- 无监督学习
- 产生式模型
  - PixelRNN and PixelCNN
  - Variational Autoencoders (VAE)
  - Generative Adversarial Networks (GAN)

2020/6/2

北京邮电大学计算机学院 鲁鹏

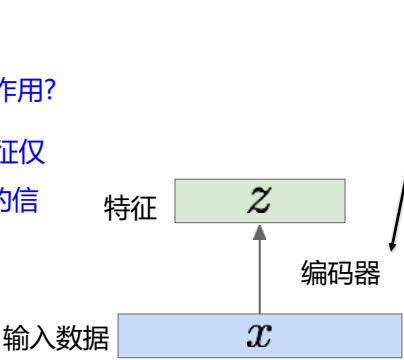
29

# 自编码器

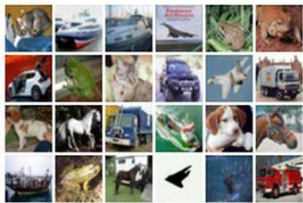
无监督的特征学习，其目标是利用无标签数据找到一个有效地低维的特征提取器。

z 的维度一般小于 x 的维度  
(特征降维)

Q: 特征降维有什么作用?  
A: 希望降维后的特征仅保留数据中有意义的信息



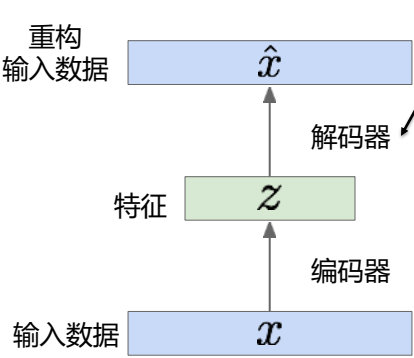
早先方法: Linear + nonlinearity (sigmoid)  
卷积神经网络流行之前: Deep, fully-connected  
卷积神经网络流行后: ReLU CNN



# 自编码器

如何学习?

自编码利用重构损失来训练低维的特征表示



早先方法: Linear + nonlinearity (sigmoid)  
卷积神经网络流行之前: Deep, fully-connected  
卷积神经网络流行后: ReLU CNN

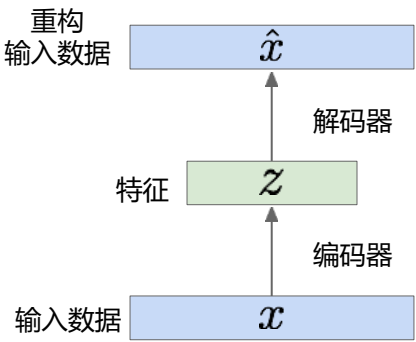




# 自编码器

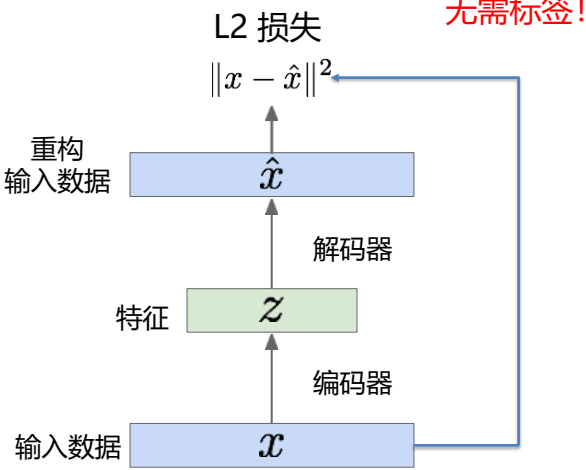
## 如何学习?

自编码利用重构损失来训练低维的特征表示



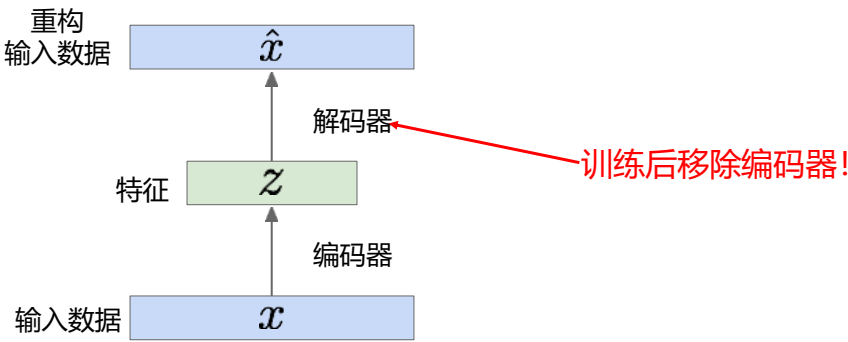
# 自编码器

## 如何学习?



# 自编码器

如何学习?



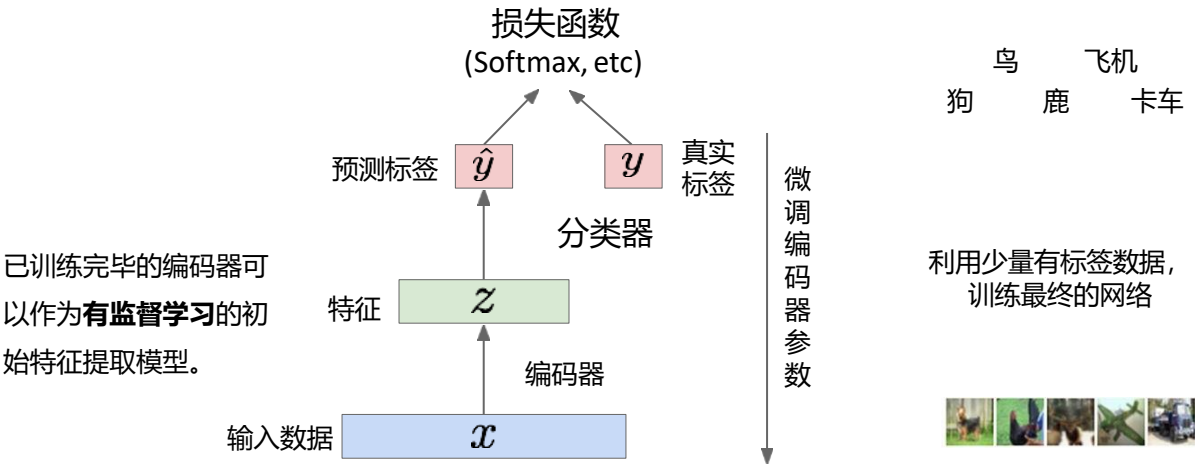
2020/6/2

北京邮电大学计算机学院 鲁鹏

34

# 自编码器

如何学习?

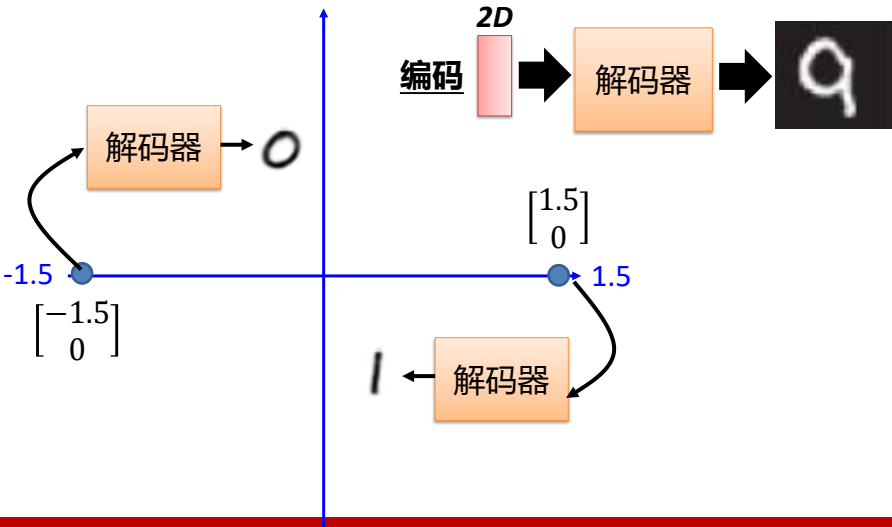


2020/6/2

北京邮电大学计算机学院 鲁鹏

35

# 自编码器

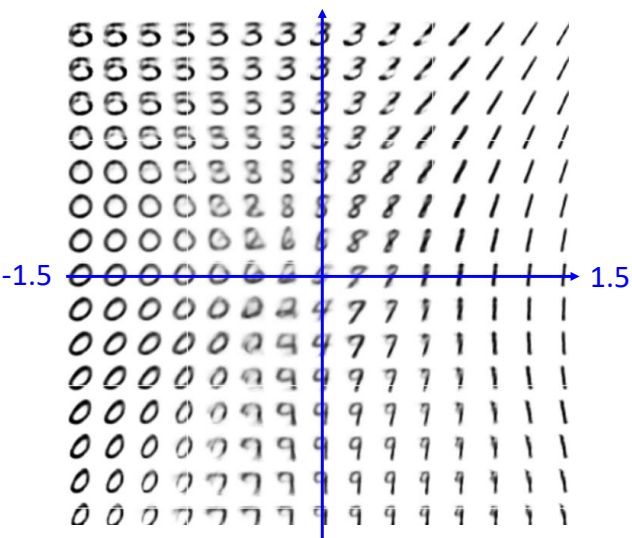


2020/6/2

北京邮电大学计算机学院 鲁鹏

36

# 自编码器



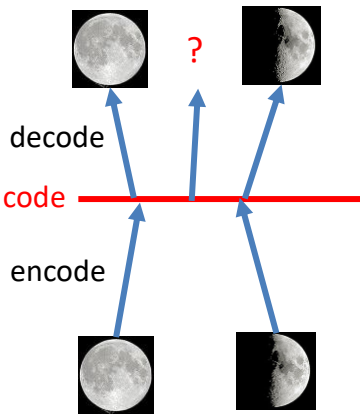
2020/6/2

北京邮电大学计算机学院 鲁鹏

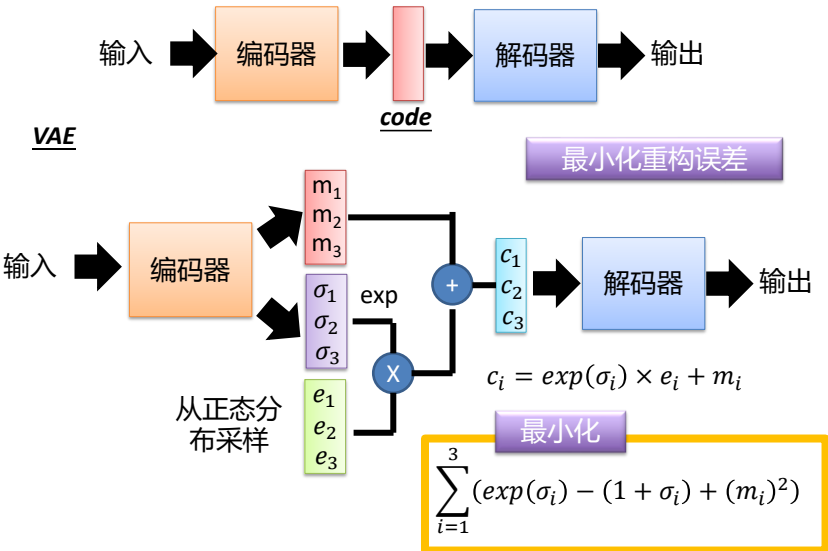
37

# Why VAE?

## 直接理由

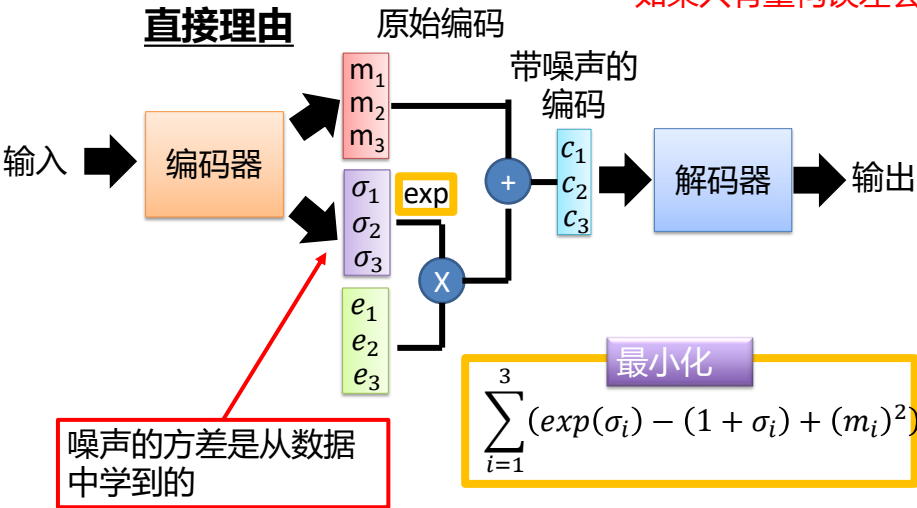


# 变分自编码器



# 变分自编码器

如果只有重构误差会如何？



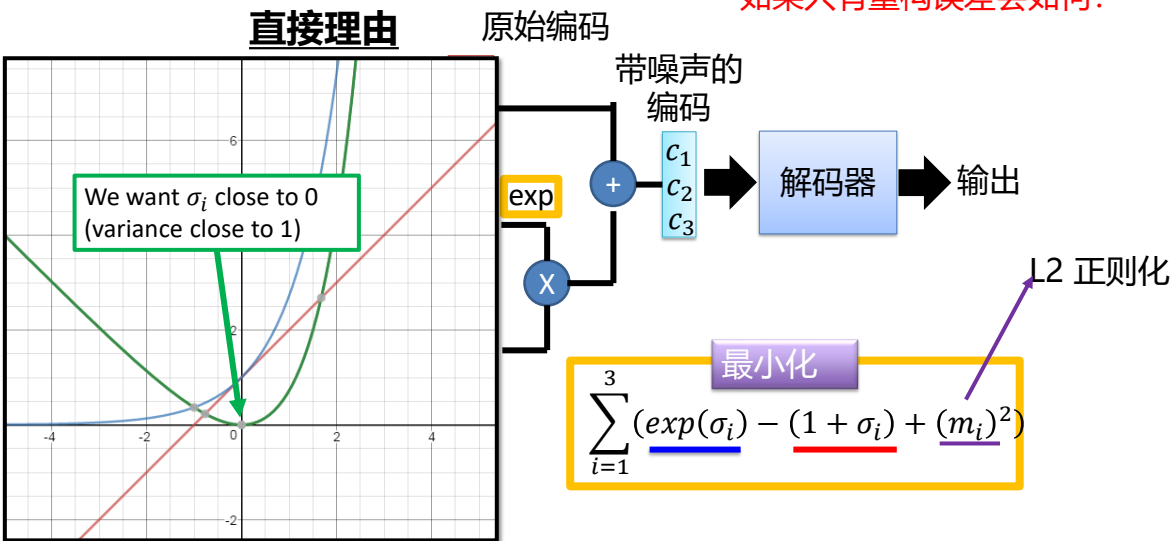
2020/6/2

北京邮电大学计算机学院 鲁鹏

40

# 变分自编码器

如果只有重构误差会如何？



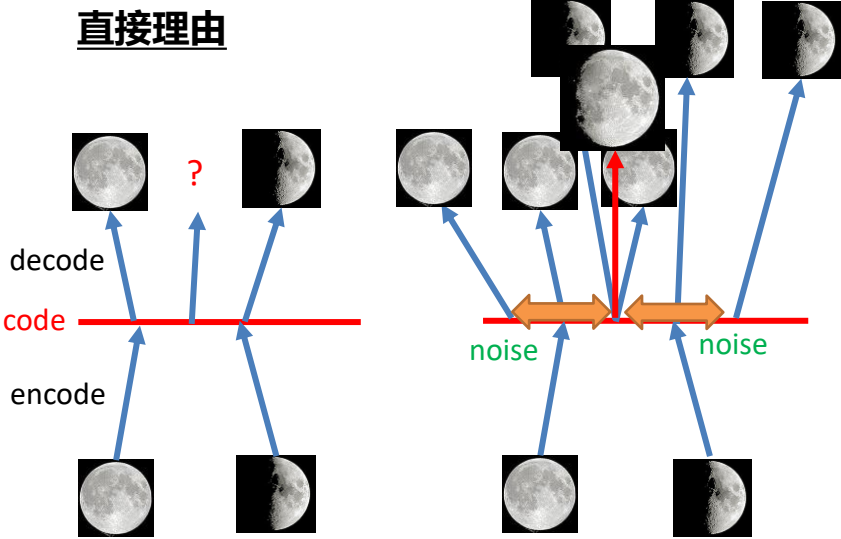
2020/6/2

北京邮电大学计算机学院 鲁鹏

41

# Why VAE?

## 直接理由



# 变分自编码器(推导篇)

如何采样?

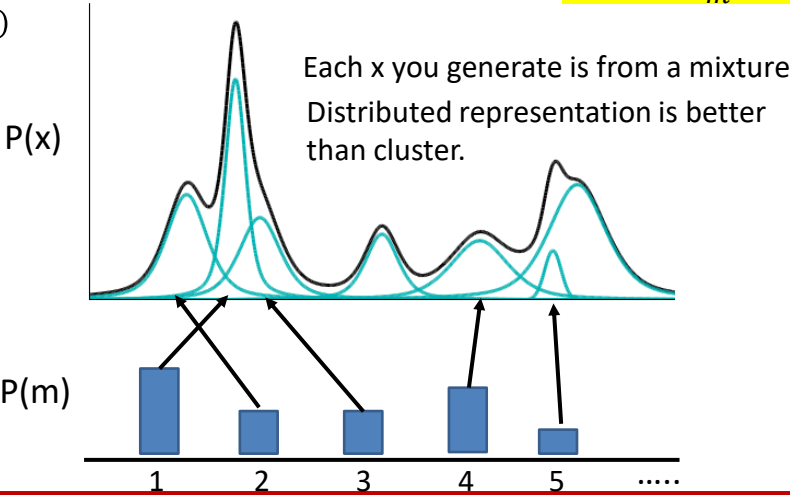
$m \sim P(m)$  (多项式分布)

$m$  是整数

$x|m \sim N(\mu^m, \Sigma^m)$

# 高斯混合模型

$$P(x) = \sum_m P(m)P(x|m)$$



# 变分自编码器(推导篇)

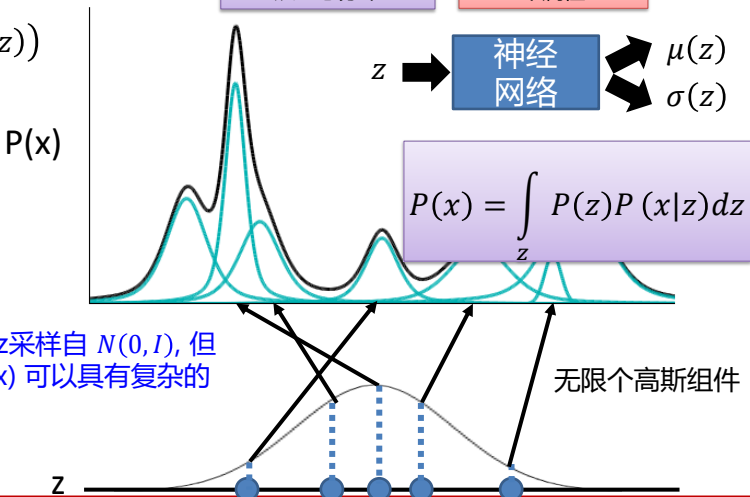
VAE

$z \sim N(0, I)$

$x|z \sim N(\mu(z), \sigma(z))$

$z$  是一个向量, 其服从正态分布

$z$  的每一维表示一个属性



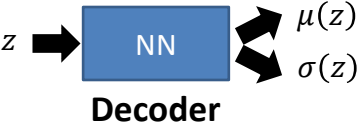
极大似然估计

$$P(x) = \int_z P(z)P(x|z)dz$$
$$L = \sum_x \log P(x)$$

P(z) 正态分布  
 $x|z \sim N(\mu(z), \sigma(z))$   
 $\mu(z), \sigma(z)$  为待估计的参数

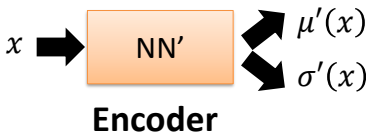
最大化观测 x 的似然

优化解码器参数使  
得似然函数 L 最大



我们需要另一个分布q(z|x)

$$z|x \sim N(\mu'(x), \sigma'(x))$$



极大似然估计

$$P(x) = \int_z P(z)P(x|z)dz$$
$$L = \sum_x \log P(x)$$

P(z) 正态分布  
 $x|z \sim N(\mu(z), \sigma(z))$   
 $\mu(z), \sigma(z)$  为待估计的参数

最大化观测 x 的似然

$$\log P(x) = \int_z q(z|x) \log P(x) dz$$
$$= \int_z q(z|x) \log \left( \frac{P(z, x)}{P(z|x)} \right) dz = \int_z q(z|x) \log \left( \frac{P(z, x) q(z|x)}{q(z|x) P(z|x)} \right) dz$$
$$= \int_z q(z|x) \log \left( \frac{P(z, x)}{q(z|x)} \right) dz + \int_z q(z|x) \log \left( \frac{q(z|x)}{P(z|x)} \right) dz$$
$$\geq \int_z q(z|x) \log \left( \frac{P(x|z)P(z)}{q(z|x)} \right) dz$$

q(z|x) 可以是任何分布

$KL(q(z|x)||P(z|x)) \geq 0$


下界 $L_b$



极大似然估计


$P(z)$  正态分布  
 $x|z \sim N(\mu(z), \sigma(z))$   
 $\mu(z), \sigma(z)$  为待估计的参数

$$P(x) = \int_z P(z)P(x|z)dz$$
  
$$L = \sum_x \log P(x)$$
 Maximizing the likelihood of the observed x

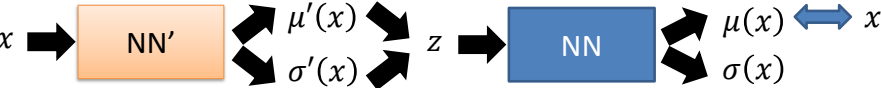
$$L_b = \int_z q(z|x) \log \left( \frac{P(z, x)}{q(z|x)} \right) dz = \int_z q(z|x) \log \left( \frac{P(x|z)P(z)}{q(z|x)} \right) dz$$
  
$$= \underbrace{\int_z q(z|x) \log \left( \frac{P(z)}{q(z|x)} \right) dz}_{-KL(q(z|x)||P(z))} + \int_z q(z|x) \log P(x|z) dz$$
  
$$z|x \sim N(\mu'(x), \sigma'(x))$$
  


Connection with Network

最小化  $KL(q(z|x)||P(z))$



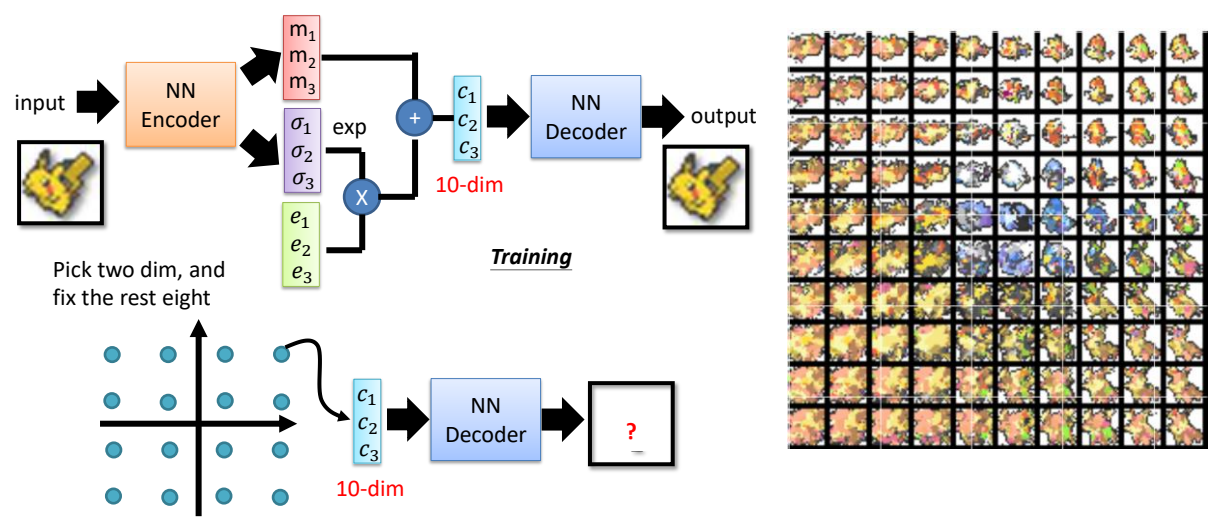
最大化  $\int_z q(z|x) \log P(x|z) dz = E_{q(z|x)}[\log P(x|z)]$



自编码结构

# 变分自编码器

## Pokémon Creation



2020/6/2

北京邮电大学计算机学院 鲁鹏

50

# 变分自编码器



32x32 CIFAR-10



Labeled Faces in the Wild

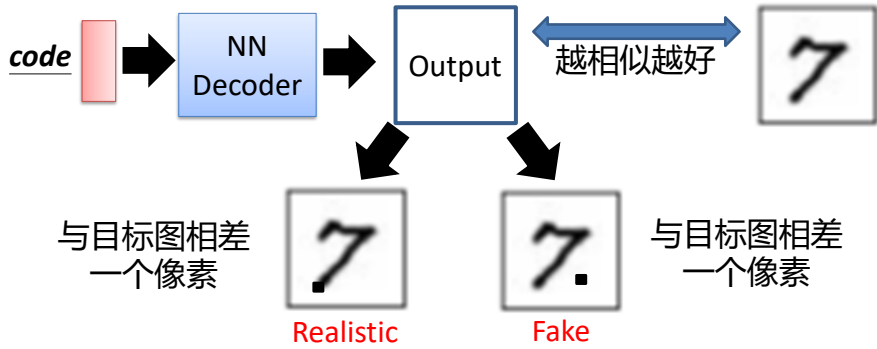
2020/6/2

北京邮电大学计算机学院 鲁鹏

51

## 变分自编码器存在的问题

- VAE 并没有真的在尝试模仿真实图片



VAE 貌似仅记住了存在的图像，而不是产生新的图像

2020/6/2

北京邮电大学计算机学院 鲁鹏

52

## 变分自编码 (VAE) 总结

基于典型自编码器拓展成的概率框架=> 可以产生新的样本

定义一个难以计算的密度函数=> 通过推导来优化一个下边界

**优点:**

- 产生式模型里的一种主要方法
- 可以计算  $q(z|x)$ ，这个特征表示可以用在其他的许多任务中。

**缺点:**

- 最大化似然函数的下边界能够有效地工作，但是模型本身没有PixelRNN/PixelCNN那样好评估
- 与最新的技术 (GANs) 相比，产生的样本较模糊，质量较低

2020/6/2

北京邮电大学计算机学院 鲁鹏

53

# 今日主题

- 无监督学习
- 产生式模型
  - PixelRNN and PixelCNN
  - Variational Autoencoders (VAE)
  - Generative Adversarial Networks (GAN)

2020/6/2

北京邮电大学计算机学院 鲁鹏

54

## 2019: BigGAN



Brock et al., 2019

2020/6/2

北京邮电大学计算机学院 鲁鹏

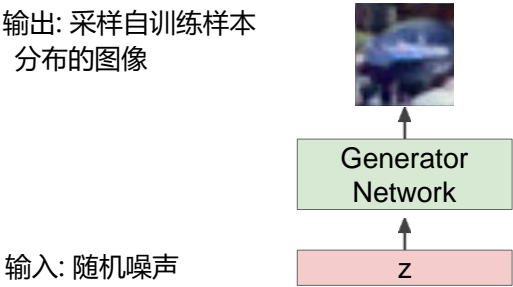
55

# 生成对抗网络 (GAN)

**问题:** 希望从训练样本分布中采样新数据, 但这个分布不仅维度高而且还很复杂, 难以直接实现。  
**解决方案:** 对一个简单的分布采样, 比如均匀分布; 然后, 学习一种映射将其变换到训练样本分布

Q: 用什么方法能够实现这个复杂的映射?

A: 神经网络!



Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

# 训练GAN: 两个玩家的游戏

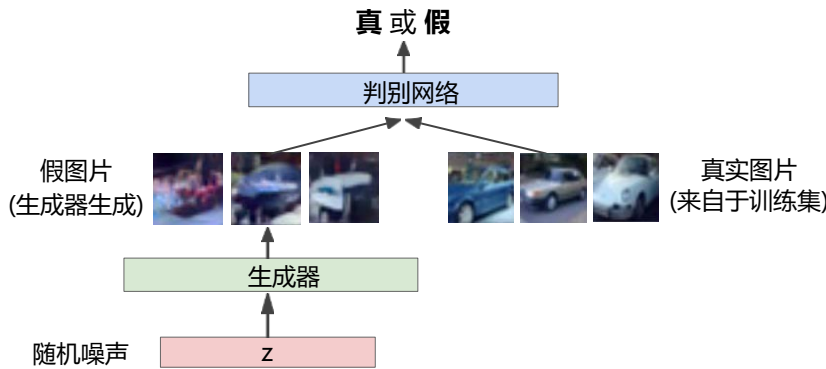
**生成网络:** 期望能够产生尽量真实的图片, 进而骗过判别器  
**判别网络:** 期望能够准确的区分真假图片

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

# 生成对抗网络 (GAN)

**生成网络:** 期望能够产生尽量真实的图片，进而骗过判别器

**判别网络:** 期望能够准确的区分真假图片



Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

# 训练GAN: 两个玩家的游戏

**生成网络:** 期望能够产生尽量真实的图片，进而骗过判别器

**判别网络:** 期望能够准确的区分真假图片

采用 **minimax** 的方式联合训练

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

# 训练GAN: 两个玩家的游戏

**生成网络:** 期望能够产生尽量真实的图片，进而骗过判别器

**判别网络:** 期望能够准确的区分真假图片

采用 **minimax 的方式联合训练**

判别器输出真实图片的似然，其值在(0,1) 之间

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\substack{\text{判别器对真实} \\ \text{样本 } x \text{ 的打分}}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\substack{\text{判别器对生成} \\ \text{样本 } G(z) \text{ 的打分}}}) \right]$$

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

# 训练GAN: 两个玩家的游戏

**生成网络:** 期望能够产生尽量真实的图片，进而骗过判别器

**判别网络:** 期望能够准确的区分真假图片

采用 **minimax 的方式联合训练**

判别器输出真实图片的似然，其值在(0,1) 之间

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\substack{\text{判别器对真实} \\ \text{样本 } x \text{ 的打分}}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\substack{\text{判别器对生成} \\ \text{样本 } G(z) \text{ 的打分}}}) \right]$$

- 判别器 ( $\theta_d$ ) 希望**最大化目标函数**使得 $D(x)$ 接近于1（真实样本），而  $D(G(z))$  接近于0（假样本）
- 生成器 ( $\theta_g$ ) 希望**最小化目标函数**使得  $D(G(z))$  尽量接近于1，即希望判别器认为生成器产生的图像  $G(z)$  为真实图片。

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

# 训练GAN: 两个玩家的游戏

Minimax 目标函数:

$$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

交替完成:

- 1. Gradient ascent on discriminator

$$\max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

- 2. Gradient descent on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

2020/6/2

北京邮电大学计算机学院 鲁鹏

62

# 训练GAN: 两个玩家的游戏

Minimax 目标函数:

$$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

交替完成:

- 1. Gradient ascent on discriminator

$$\max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

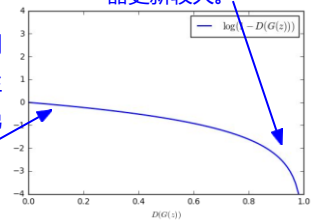
- 2. Gradient descent on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

实际上, 直接优化生成器的这个目标函数并不是很有效!

生成样本非常糟糕时, 判别器输出值都会很小, 生成器损失函数在此处得梯度很小, 使得生成器学习得非常慢。

相反, 当生成样本较好时, 判别器输出值都会很大, 生成器损失函数在此处得梯度很大, 生成器更新较大。



Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

2020/6/2

北京邮电大学计算机学院 鲁鹏

63



# 训练GAN: 两个玩家的游戏

Minimax 目标函数:

$$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

交替完成:

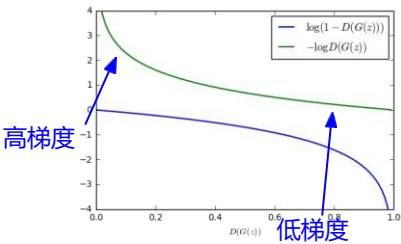
- 1. Gradient ascent on discriminator

$$\max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

- 2. Instead: Gradient ascent on generator, different objective

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

真实中很好用，大家几乎都这么用！

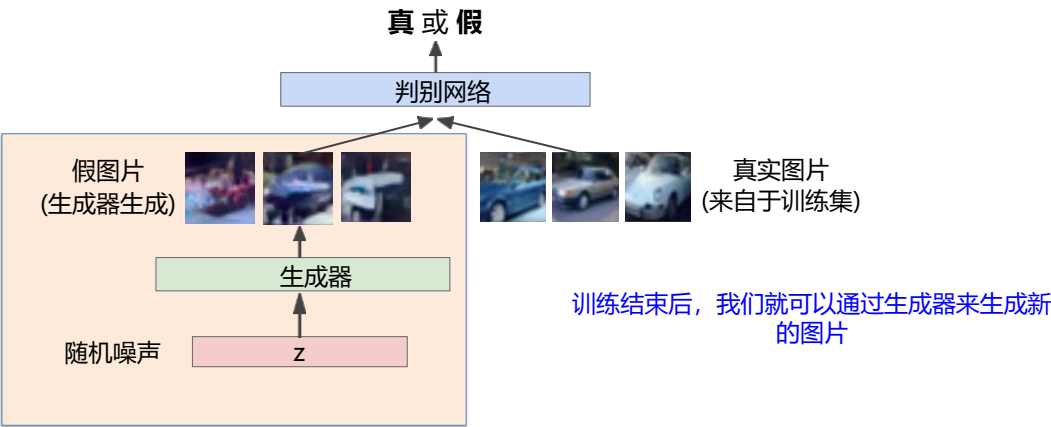


Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

# 生成对抗网络 (GAN)

问题: 希望从训练样本分布中采样新数据，但这个分布不仅维度高而且还很复杂，难以直接实现。

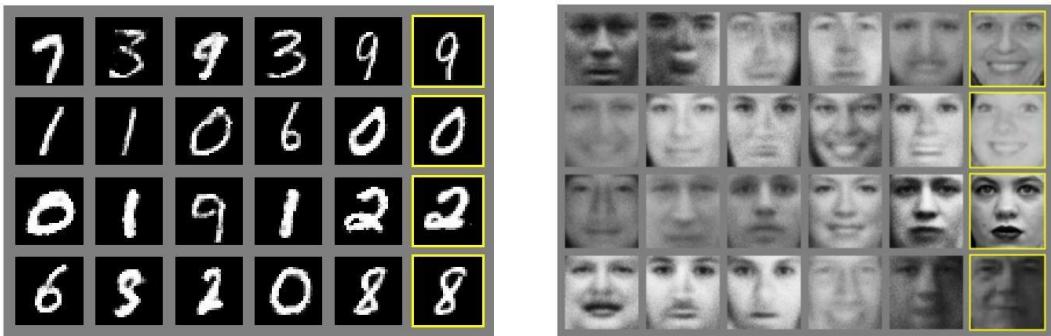
解决方案: 对一个简单的分布采样，比如均匀分布；然后，学习一种映射将其变换到训练样本分布



Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

# 生成对抗网络 (GAN)

产生的样本



在训练样本中找到的最近邻

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

# 生成对抗网络 (GAN)

产生的样本(CIFAR10)



在训练样本中找到的最近邻

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

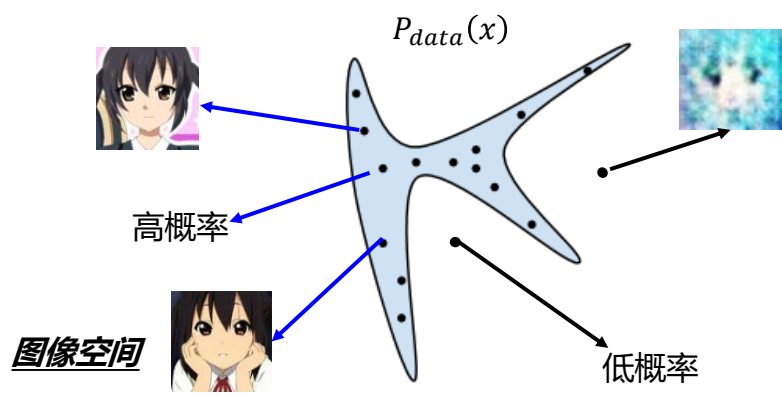
# 生成对抗网络 (GAN)

## GAN背后的数学故事!

### Generation

$x$ : 图像(高维向量)

- 我们的目标是找到数据分布  $P_{data}(x)$



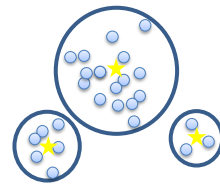
# 极大似然估计

- 给定数据分布  $P_{data}(x)$  (可以对其采样获得数据.)
- 我们能够得到一个分布  $P_G(x; \theta)$ , 其由参数  $\theta$  所决定
  - 目标: 确定参数  $\theta$  使得  $P_G(x; \theta)$  与  $P_{data}(x)$  相似
  - 比如  $P_G(x; \theta)$  为高斯混合模型,  $\theta$  则表示高斯分布的矩阵与协方差矩阵

从  $P_{data}(x)$  采样样本  $\{x^1, x^2, \dots, x^m\}$

计算  $P_G(x^i; \theta)$  产生这些样本的似然

$$L = \prod_{i=1}^m P_G(x^i; \theta)$$



通过极大似然估计, 我们可以求取  $\theta^*$

2020/6/2

北京邮电大学计算机学院 鲁鹏

70

## 最大化似然=最小化KL散度

$$\begin{aligned}
 \theta^* &= \arg \max_{\theta} \prod_{i=1}^m P_G(x^i; \theta) = \arg \max_{\theta} \log \prod_{i=1}^m P_G(x^i; \theta) \\
 &= \arg \max_{\theta} \sum_{i=1}^m \log P_G(x^i; \theta) \quad \text{从 } P_{data}(x) \text{ 采样 } \{x^1, x^2, \dots, x^m\} \\
 &\approx \arg \max_{\theta} E_{x \sim P_{data}} [\log P_G(x; \theta)] \\
 &= \arg \max_{\theta} \int_x P_{data}(x) \log P_G(x; \theta) dx - \int_x P_{data}(x) \log P_{data}(x) dx \\
 &= \arg \min_{\theta} KL(P_{data} || P_G) \quad \text{如何定义一个通用的 } P_G?
 \end{aligned}$$

2020/6/2

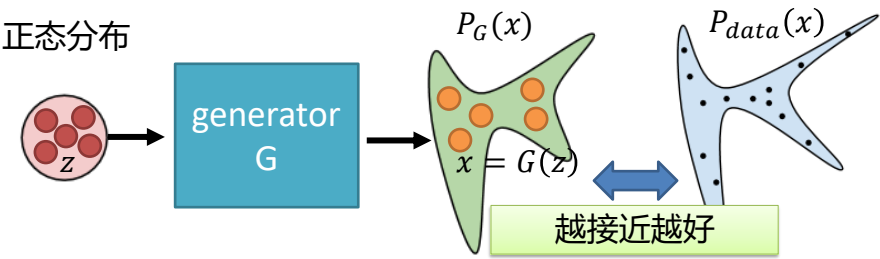
北京邮电大学计算机学院 鲁鹏

71

# 产生器 (Generator)

$x$ : 图像(高维向量)

- 利用神经网络来构建产生器  $G$ , 网络输出定义了一个密度分布  $P_G$



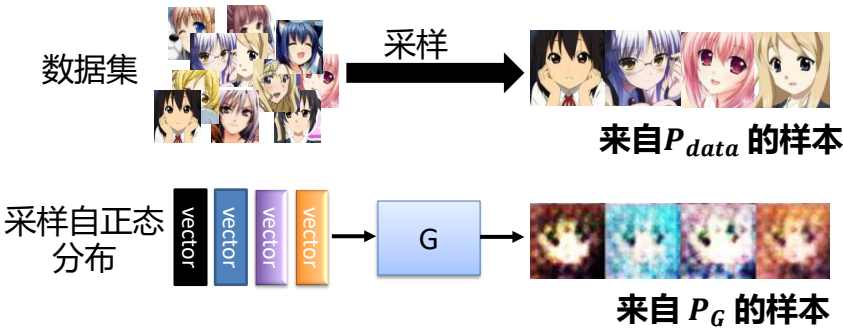
$$G^* = \arg \min_G \text{Div}(P_G, P_{data})$$
  
Div 表示  $P_G$  与  $P_{data}$  的散度

DIV如何计算?

# 判别器 (Discriminator)

$$G^* = \arg \min_G \text{Div}(P_G, P_{data})$$

虽然不知道  $P_G$  和  $P_{data}$  的分布, 但是, 可以得到它们的采样数据。

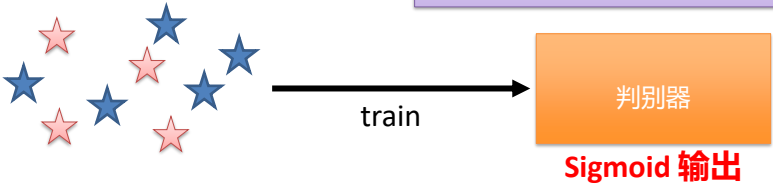


# Discriminator

$$G^* = \arg \min_G \text{Div}(P_G, P_{data})$$

★: 采样自  $P_{data}$  的样本  
★: 采样自  $P_G$  的样本

Using the example objective function is exactly the same as training a binary classifier.



Example Objective Function for D

$$V(G, D) = E_{x \sim P_{data}} [\log D(x)] + E_{x \sim P_G} [\log(1 - D(x))]$$

(G is fixed)

训练:  $D^* = \arg \max_D V(D, G)$  最大化红框中的目标函数等价于最小化JS散度

[Goodfellow, et al., NIPS, 2014]

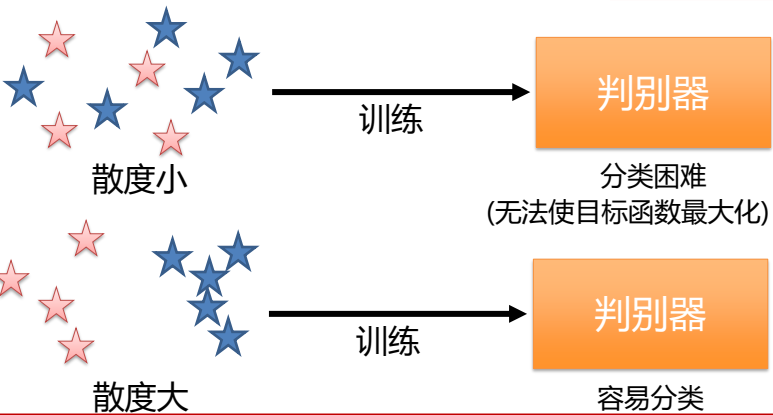
# Discriminator

$$G^* = \arg \min_G \text{Div}(P_G, P_{data})$$

★: 采样自  $P_{data}$  的样本  
★: 生成器  $P_G$  生成的样本

训练:

$$D^* = \arg \max_D V(D, G)$$



$$\max_D V(G, D)$$

$$V = E_{x \sim P_{data}}[\log D(x)] + E_{x \sim P_G}[\log(1 - D(x))]$$

- 给定 G, 我们的目标是最大化下式来寻找最优化的 D\*

$$\begin{aligned} V &= E_{x \sim P_{data}}[\log D(x)] + E_{x \sim P_G}[\log(1 - D(x))] \\ &= \int_x P_{data}(x) \log D(x) dx + \int_x P_G(x) \log(1 - D(x)) dx \\ &= \int_x [P_{data}(x) \log D(x) + P_G(x) \log(1 - D(x))] dx \end{aligned}$$

假设 D(x) 可以取任何函数

- 给定 x, 最大化下式可获得最优的判别器 D\*

$$P_{data}(x) \log D(x) + P_G(x) \log(1 - D(x))$$

2020/6/2

北京邮电大学计算机学院 鲁鹏

76

$$\max_D V(G, D)$$

- 给定 x, 最大化下式可获得最优的判别器 D\*

$$P_{data}(x) \log D(x) + P_G(x) \log(1 - D(x))$$

a                      D                      b                      D

$$V = E_{x \sim P_{data}}[\log D(x)] + E_{x \sim P_G}[\log(1 - D(x))]$$

- 求取最优 D\*, 需要最大化:  $f(D) = a \log(D) + b \log(1 - D)$

$$\frac{df(D)}{dD} = a \times \frac{1}{D} + b \times \frac{1}{1-D} \times (-1) = 0$$

$$a \times \frac{1}{D^*} = b \times \frac{1}{1-D^*} \quad a \times (1-D^*) = b \times D^* \quad a - aD^* = bD^* \quad a = (a+b)D^*$$

$$D^* = \frac{a}{a+b} \quad \Rightarrow \quad 0 < D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)} < 1$$

2020/6/2

北京邮电大学计算机学院 鲁鹏

77

$$\max_D V(G, D)$$

$$\max_D V(G, D) = V(G, D^*) \quad D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)}$$

$$V = E_{x \sim P_{data}}[\log D(x)] + E_{x \sim P_G}[\log(1 - D(x))]$$

$$= E_{x \sim P_{data}} \left[ \log \frac{P_{data}(x)}{P_{data}(x) + P_G(x)} \right] + E_{x \sim P_G} \left[ \log \frac{P_G(x)}{P_{data}(x) + P_G(x)} \right]$$

$$= \int_x P_{data}(x) \log \frac{\frac{1}{2} P_{data}(x)}{\frac{P_{data}(x) + P_G(x)}{2}} dx + 2 \log \frac{1}{2} - 2 \log 2$$

$$+ \int_x P_G(x) \log \frac{\frac{1}{2} P_G(x)}{\frac{P_{data}(x) + P_G(x)}{2}} dx$$

2020/6/2

北京邮电大学计算机学院 鲁鹏

78

$$\max_D V(G, D)$$

$$\max_D V(G, D) = V(G, D^*) \quad D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)} \quad \text{JSD}(P \parallel Q) = \frac{1}{2} D(P \parallel M) + \frac{1}{2} D(Q \parallel M)$$

$$M = \frac{1}{2}(P + Q)$$

$$= -2 \log 2 + \int_x P_{data}(x) \log \frac{P_{data}(x)}{(P_{data}(x) + P_G(x))/2} dx$$

$$+ \int_x P_G(x) \log \frac{P_G(x)}{(P_{data}(x) + P_G(x))/2} dx$$

$$= -2 \log 2 + \text{KL} \left( P_{data} \parallel \frac{P_{data} + P_G}{2} \right) + \text{KL} \left( P_G \parallel \frac{P_{data} + P_G}{2} \right)$$

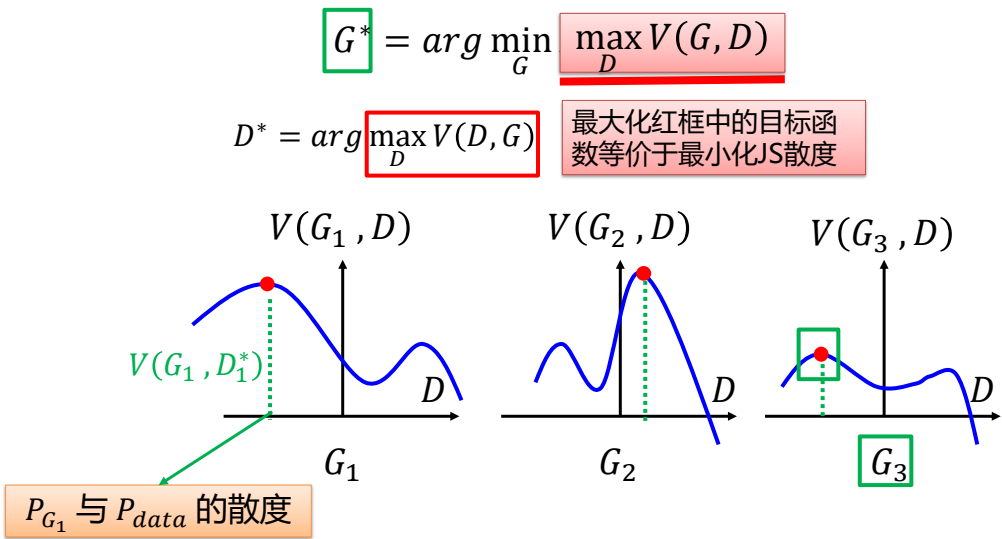
$$= -2 \log 2 + 2 \text{JSD}(P_{data} \parallel P_G) \quad \text{Jensen-Shannon divergence}$$

2020/6/2

北京邮电大学计算机学院 鲁鹏

79





[Goodfellow, et al., NIPS, 2014]

$$G^* = \arg \min_G \max_D V(G, D)$$
$$D^* = \arg \max_D V(D, G)$$

最大化红框中的目标函数等价于最小化JS散度

- 初始化生成器与判别器
- 每轮迭代完成:
  - Step 1: 固定生成器  $G$ , 更新判别器  $D$
  - Step 2: 固定判别器  $D$ , 更新生成器  $G$

# Algorithm

$$G^* = \arg \min_G \max_D V(G, D)$$

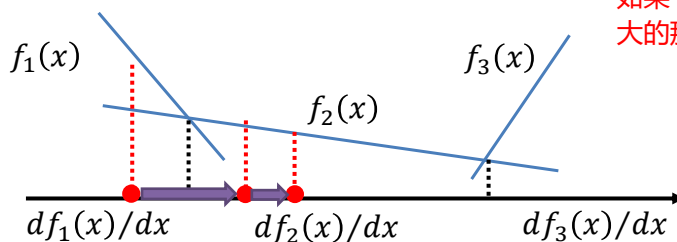
$L(G)$

- 最小化损失函数  $L(G)$  来获得最优的  $G$

$$\theta_G \leftarrow \theta_G - \eta \partial L(G) / \partial \theta_G \quad \theta_G \text{ defines } G$$

$$f(x) = \max\{f_1(x), f_2(x), f_3(x)\} \quad \frac{df(x)}{dx} = ? \quad df_i(x)/dx$$

如果  $f_i(x)$  是最大的那一个

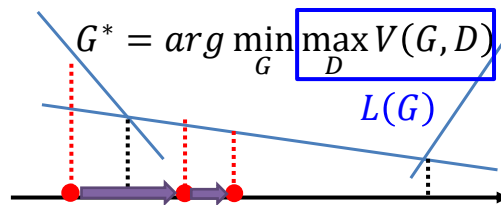


2020/6/2

北京邮电大学计算机学院 鲁鹏

82

# Algorithm



- 给定  $G_0$
- 寻找  $D_0^*$  使得  $V(G_0, D)$  最大化

使用梯度上升法

$V(G_0, D_0^*)$  为  $P_{data}(x)$  与  $P_{G_0}(x)$  的JS散度

- $\theta_G \leftarrow \theta_G - \eta \partial V(G, D_0^*) / \partial \theta_G \rightarrow$  Obtain  $G_1$  减小JS散度(?)
- 寻找  $D_1^*$  使得  $V(G_1, D)$  最大化

$V(G_1, D_1^*)$  为  $P_{data}(x)$  与  $P_{G_1}(x)$  的JS散度

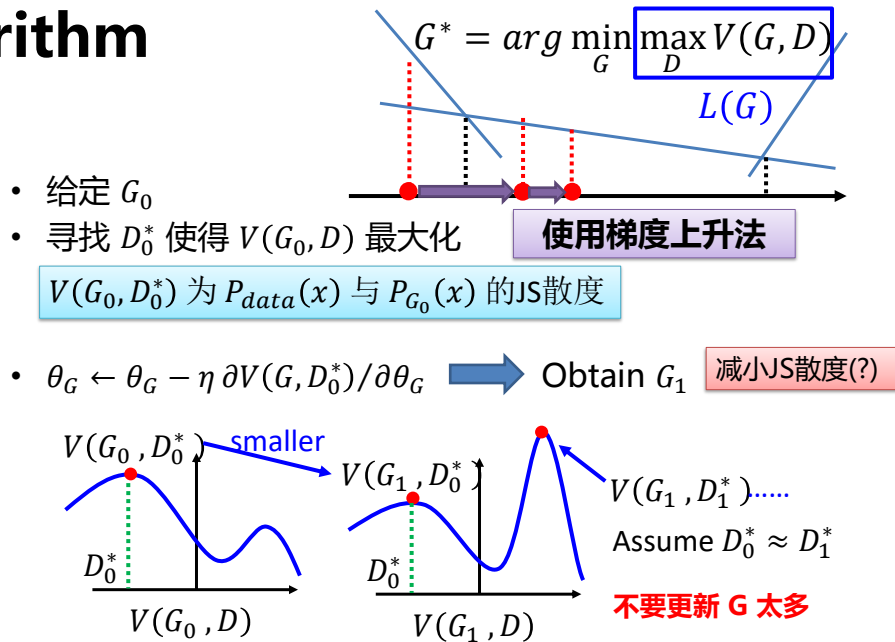
- $\theta_G \leftarrow \theta_G - \eta \partial V(G, D_1^*) / \partial \theta_G \rightarrow$  Obtain  $G_2$  减小JS散度(?)
- .....

2020/6/2

北京邮电大学计算机学院 鲁鹏

83

# Algorithm



2020/6/2

北京邮电大学计算机学院 鲁鹏

84

## In practice ...

- 给定生成器  $G$ , 如何计算  $\max_D V(G, D)$ 
  - 从  $P_{data}(x)$  采样样本  $\{x^1, x^2, \dots, x^m\}$
  - 从生成器  $P_G(x)$  产生样本  $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}$

$$V = E_{x \sim P_{data}} [\log D(x)] + E_{x \sim P_G} [\log (1 - D(x))]$$

最大化

$$\tilde{V} = \frac{1}{m} \sum_{i=1}^m \log D(x^i) + \frac{1}{m} \sum_{i=1}^m \log (1 - D(\tilde{x}^i))$$

两类分类器

$D$  是一个二分类器 (sigmoid 后输出)

从  $P_{data}(x)$  采样  $\{x^1, x^2, \dots, x^m\}$   $\rightarrow$  正样本

利用  $P_G(x)$  生成  $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}$   $\rightarrow$  负样本

最小化  
Cross-entropy

2020/6/2

北京邮电大学计算机学院 鲁鹏

85

## Algorithm

初始化判别器 D 与生成器 G 的参数  $\theta_d$  和  $\theta_g$

仅能找到下边界

$$\max_D V(G, D)$$

- 每轮迭代需要完成:

学习判别器 D

重复执行 k 次

- 从分布  $P_{data}(x)$  采样 m 个样本  $\{x^1, x^2, \dots, x^m\}$
- 从先验  $P_{prior}(z)$  采样 m 个噪声样本  $\{z^1, z^2, \dots, z^m\}$
- 利用产生器产生样本  $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}, \tilde{x}^i = G(z^i)$
- 更新判别器参数  $\theta_d$  以最大化下式
  - $\tilde{V} = \frac{1}{m} \sum_{i=1}^m \log D(x^i) + \frac{1}{m} \sum_{i=1}^m \log (1 - D(\tilde{x}^i))$
  - $\theta_d \leftarrow \theta_d + \eta \nabla \tilde{V}(\theta_d)$

学习生成器 G

仅执行 1 次

- 从先验  $P_{prior}(z)$  采样 m 个噪声样本  $\{z^1, z^2, \dots, z^m\}$
- 更新生成器参数  $\theta_g$  以最小化
  - $\tilde{V} = \frac{1}{m} \sum_{i=1}^m \log D(x^i) + \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(z^i)))$
  - $\theta_g \leftarrow \theta_g - \eta \nabla \tilde{V}(\theta_g)$

2020/6/2

北京邮电大学计算机学院 鲁鹏

86

## Objective Function for Generator in Real Implementation

$$V = \cancel{E_{x \sim P_{data}} [\log D(x)]} + E_{x \sim P_G} [\log (1 - D(x))]$$

初始时非常慢

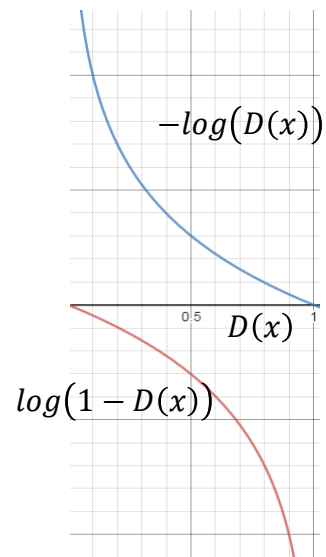
最小最大化 GAN (MMGAN)

$$V = E_{x \sim P_G} [-\log (D(x))]$$

实际上的实现:

将来自  $P_G$  的样本  $x$  标记为正样本

Non-saturating GAN (NSGAN)



2020/6/2

北京邮电大学计算机学院 鲁鹏

87

# GAN总结

- 不需要显示的密度函数定义
- 采用了游戏理论方法：利用2个玩家的博弈，来学习训练数据的分布。
- 优点：
  - 优雅，目前最好的生成效果
- 缺点：
  - 很难训练，非常不稳定
  - 无法计算出样本或者隐变量的概率，如  $p(x)$ ,  $p(z|x)$
- 领域热点：
  - 更好的损失函数，更稳定的训练方法 (Wasserstein GAN, LSGAN等等, )
  - 条件GANs (Conditional GANs)，将 GANs 应用于多种任务。

2020/6/2

Lecture 11 - 88  
北京邮电大学计算机学院 鲁鹏

88

# GAN的学习资源

1. CS 236: [Deep Generative Models](#) (Stanford)
2. CS 294-158: [Deep Unsupervised Learning](#) (Berkeley)
3. [Machine Learning](#) (台湾大学):  
[http://speech.ee.ntu.edu.tw/~tlkagk/courses\\_ML20.html](http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML20.html)

2020/6/2

北京邮电大学计算机学院 鲁鹏

89

# 今日主题

- 无监督学习 (完)
- 产生式模型 (完)
  - PixelRNN and PixelCNN
  - Variational Autoencoders (VAE)
  - Generative Adversarial Networks (GAN)