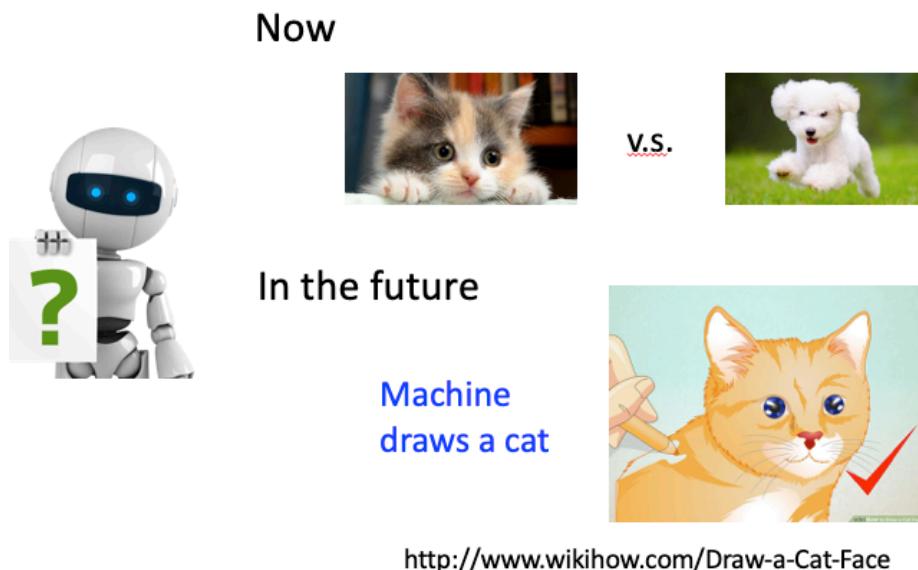


本文主要叙述了集中generative models，包括PixelRNN、VAE，GAN（生成对抗网络）；还叙述了auto-encoder和VAE的区别，以及VAE进行改进的地方，通过高斯混合模型来对VAE的原理进行了详细介绍。

Overview

Generative Models可以分为三大类：pixelRNN、autoencoder、GAN；

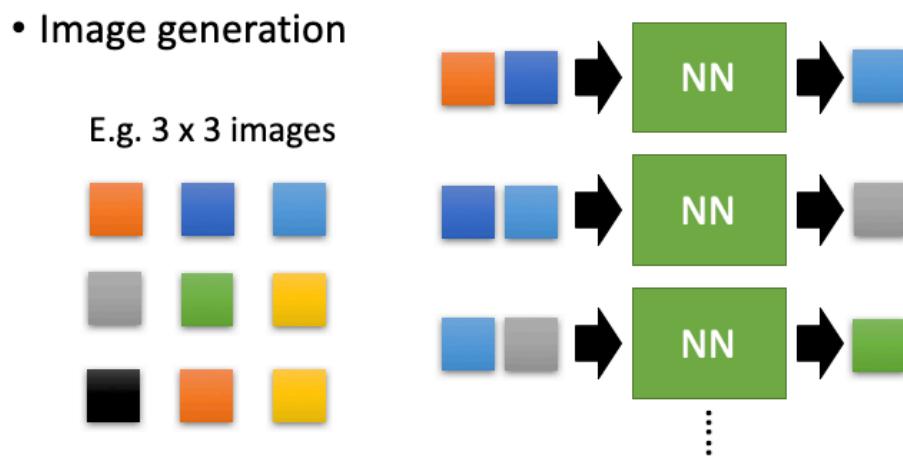
现在machine可以对猫狗进行分类，在将来machine就可以通过自己的学习画出一只猫来。



PixelRNN

Introduction

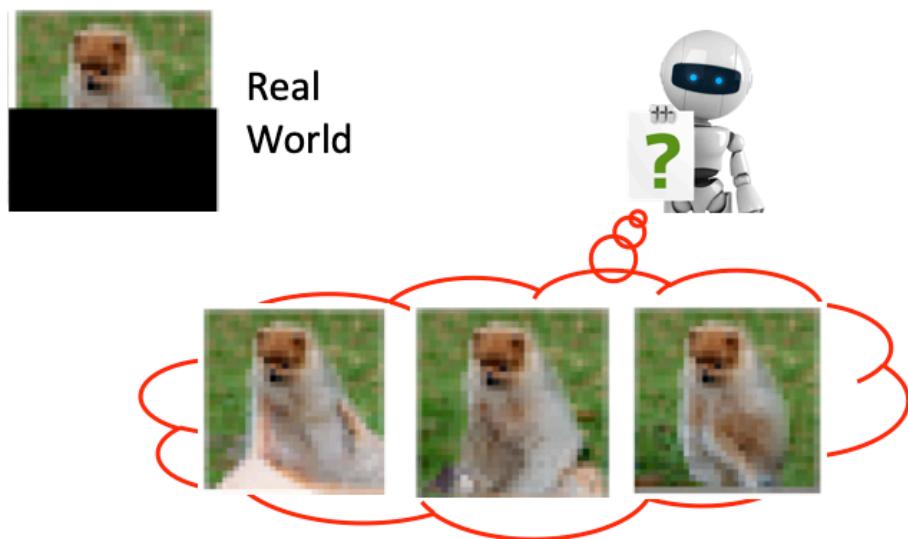
如果要生成一张图像，我们可以先生成这张图像的第一个红色pixel，再根据这个pixel生成下一个蓝色pixel，每个pixel可以用一个三维的vector来进行表示



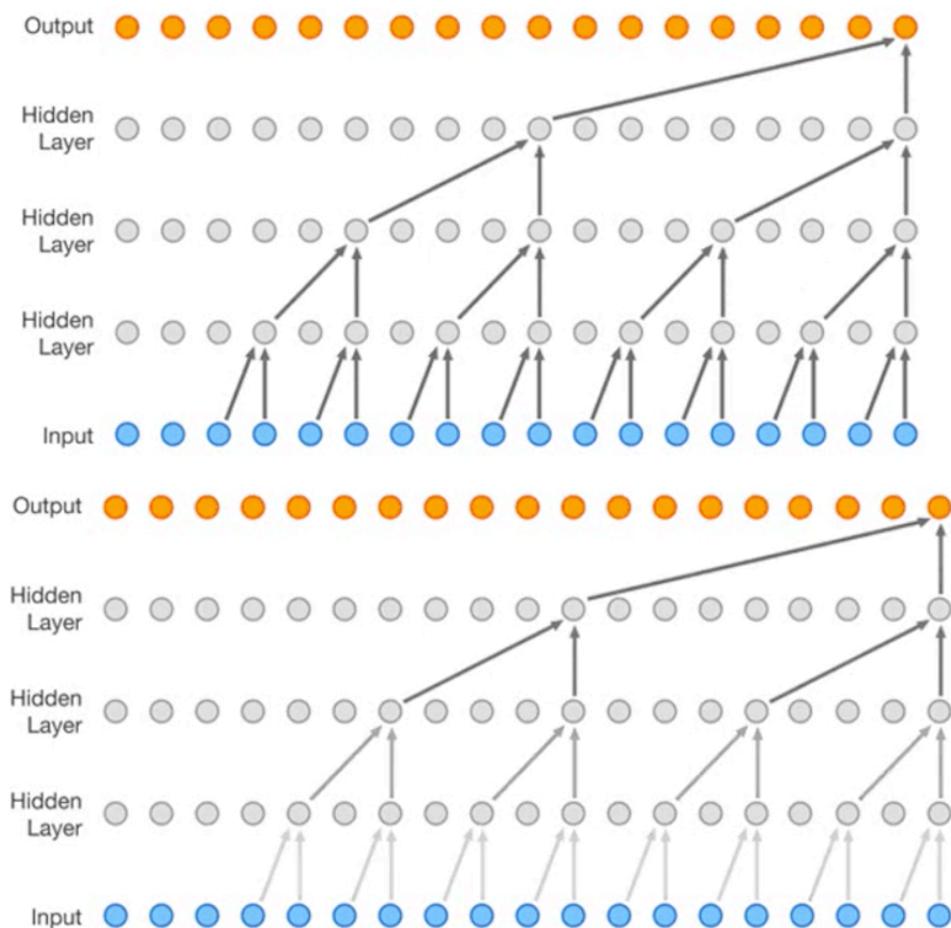
Can be trained just with a large collection of images
without any annotation

这是一个无监督学习的过程，并不需要对data进行标注

如果我们现在把下图中的狗下半身遮住，可以让machine学习出下半身的图片



也可以用到声讯号



Practicing Generation Models: Pokémon Creation

- Small images of 792 Pokémons
 - Can machine learn to create new Pokémons?

Don't catch them! Create them!

- Source of image:
[http://bulbapedia.bulbagarden.net/wiki/List_of_Pok%C3%A9mon_by_base_stats_\(Generation_VI\)](http://bulbapedia.bulbagarden.net/wiki/List_of_Pok%C3%A9mon_by_base_stats_(Generation_VI))

Original image is 40 x 40

Making them into 20 x 20



做这个实验时，有一些tips：

由于每个pixel都使用三维的vector来表示（RGB），只有三个channel的值相差特别大时，才会有颜色特别鲜明的图片，但学习结果并不能保证这一点，因此图片会比较灰蒙蒙的，最后得出来的结果会不太好；

因此，每个pixel最好使用1-of-N encoding来表示，输入一个绿色的方块，vector中只有绿色方块对应的dimensions为1，其他都是0；但有 256×256 种颜色，颜色种类非常繁多，可以先对类似的颜色做一个clustering，类似的颜色都用同一种颜色来表示，可以把颜色数量缩小到167种

- Tips (?)

➤ Each pixel is represented by 3 numbers (corresponding to RGB)



R=50, G=150, B=100

➤ Each pixel is represented by a 1-of-N encoding feature



Clustering the similar color → 167 colors in total

下面开始正式做实验

- Original image (40 x 40):
http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2016/Pokemon_creation/image.rar
- Pixels (20 x 20):
http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2016/Pokemon_creation/pixel_color.txt
 - Each line corresponds to an image, and each number corresponds to a pixel
 - http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2016/Pokemon_creation/colormap.txt

```

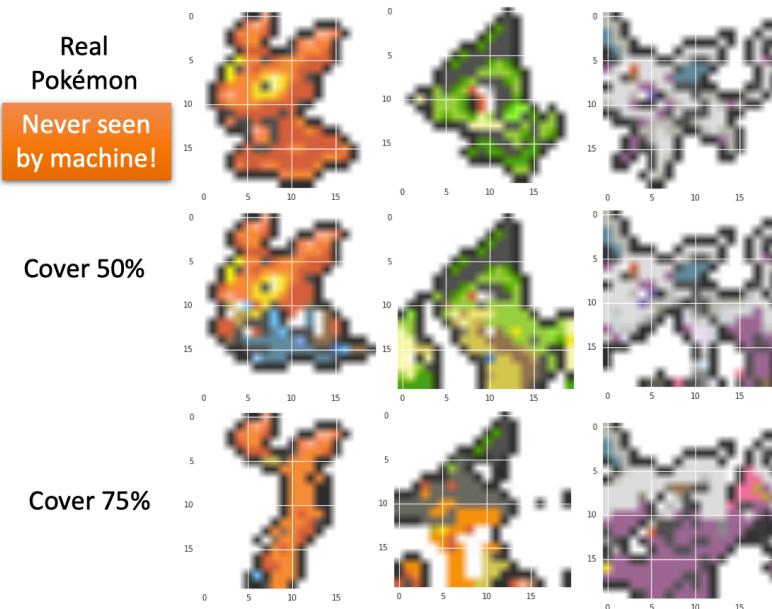
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 19 41 34 0 0 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 44 74 44 51 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 21 80 80 81 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 2 3 18 35 22 0 5 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0
93 94 93 93 85 95 38 96 97 98 99 99 67 99 9
0 0 0 0 0 0 1 106 106 106 106 61 107 0

```

0	255	255	255
1	53	53	53
2	49	49	49
	186	186	186
	51	51	51
	54	54	54
	187	187	187
	83	83	83
	50	51	52
	251	251	251
	52	52	52

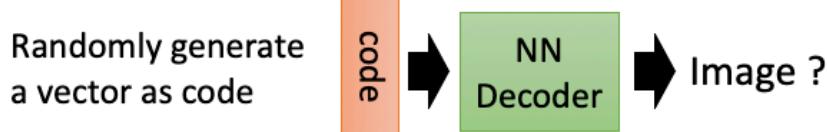
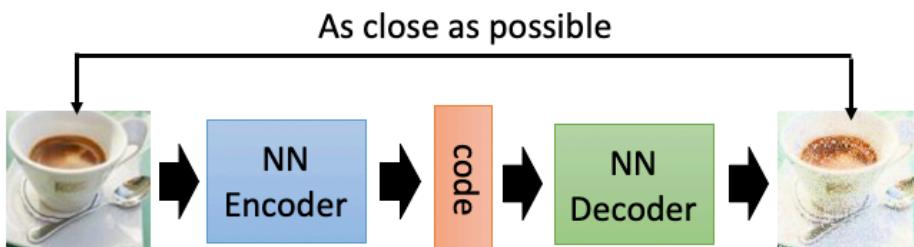
⋮

- Following experiment: 1-layer LSTM, 512 cells



VAE

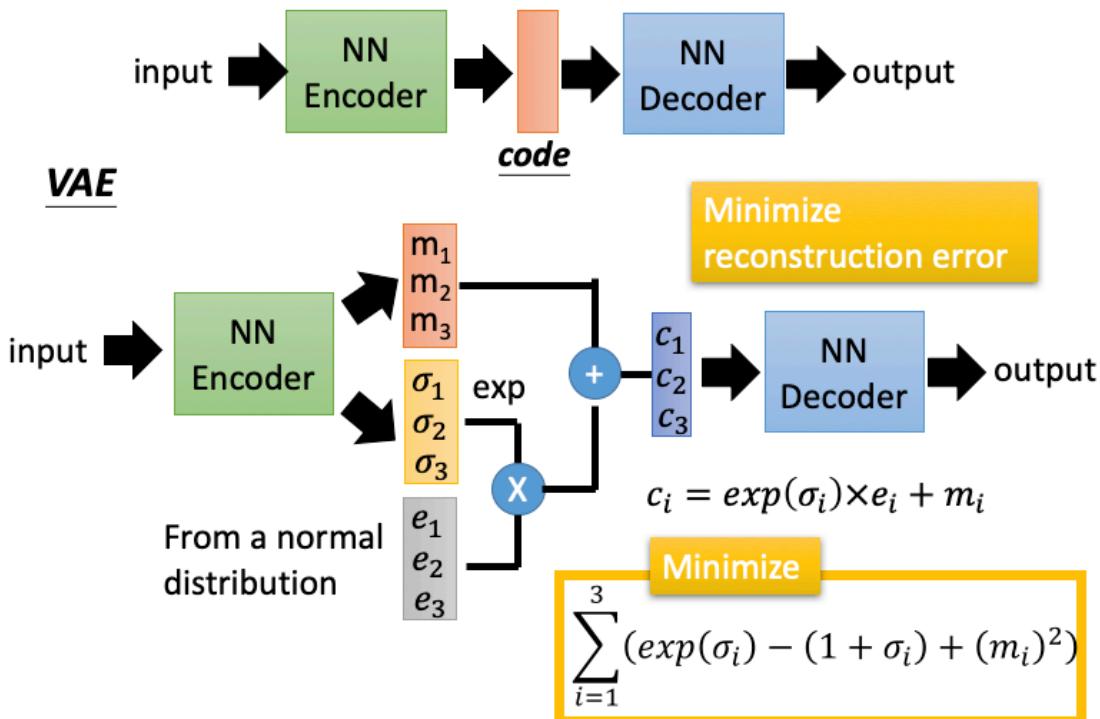
Auto-encoder



VAE

VAE不像autoencoder那样，直接得出code，经过了一些变换，即 $c_i = \exp(\sigma_i) \times e_i + m_i$ ，也是来最小化reconstruction error

Auto-encoder



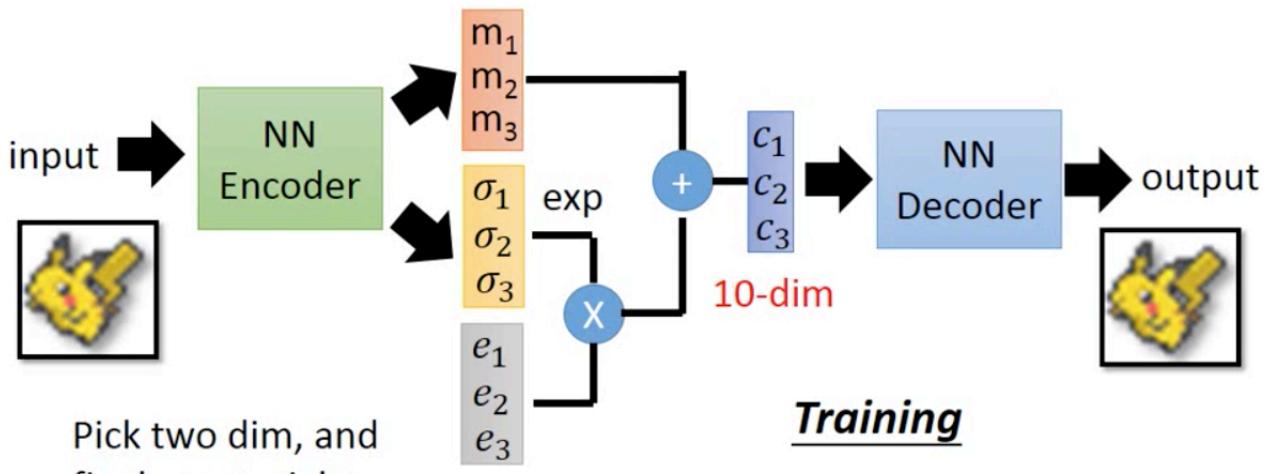
minimize的目标还有

$$\sum_{i=1}^3 (\exp(\sigma_i) - (1 + \sigma_i) + (m_i)^2)$$

Pokémon Creation

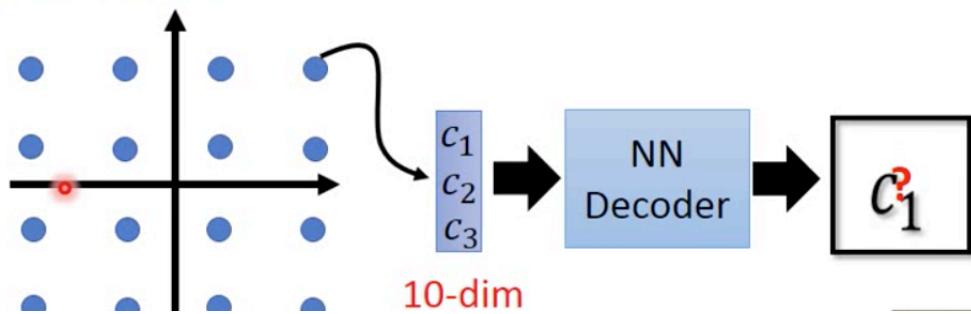
input一个宝可梦图像，进行encoder、decoder，得到reconstruct之后的图像，其中code是10-dim的；

现在我们只选择其中的2个纬度出来，其他纬度的值都固定不变，对这个二维的坐标轴，放入不同的点，再输入对应的Decoder，观察其合成出来的image；如果使用不同维度的点，就可以观察到不同维度生成的image；根据不同的维度，我们就可以根据自己的需要，调整这些维度的值，从而得出我们想要的结果



Pick two dim, and fix the rest eight

Training



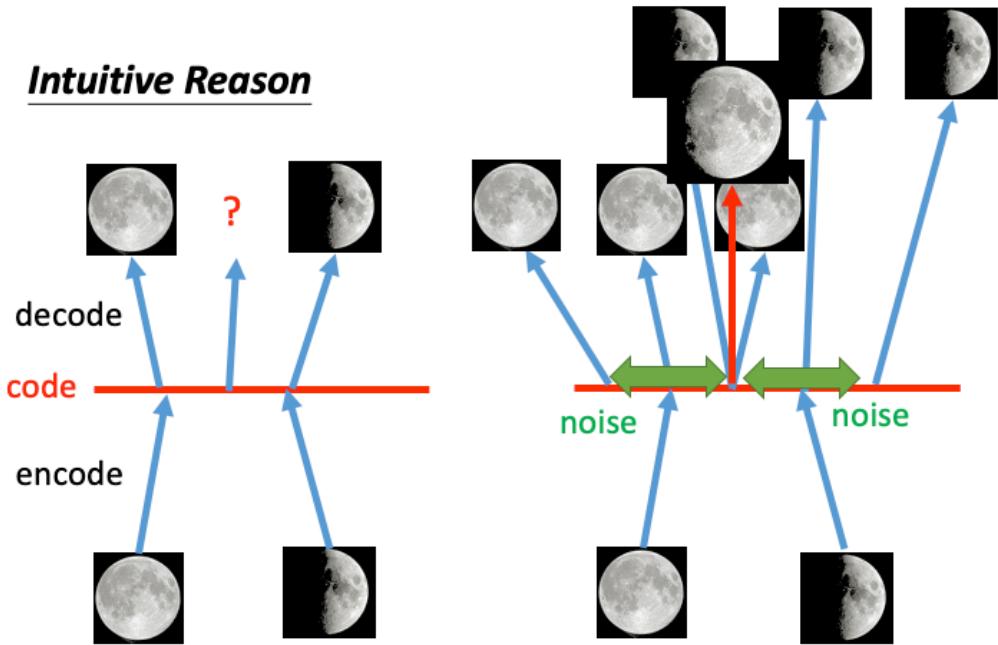
部分结果展示



Why VAE?

对于一张满月的图像，先进行encode，再进行decode，使得input和output之间的差值最小化；对半月的图像输入也如此

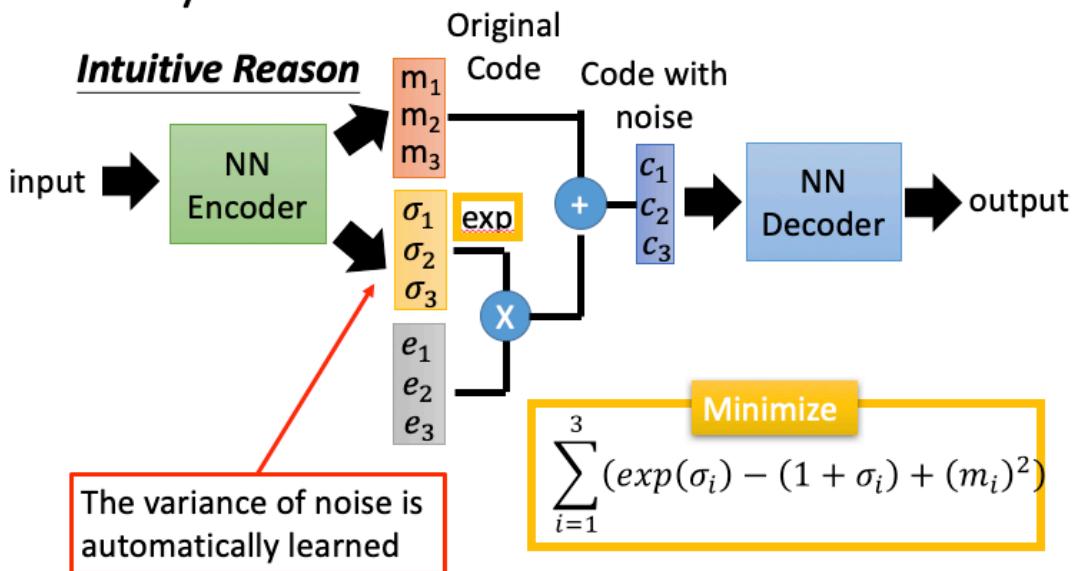
Intuitive Reason



对于encoder的其中一个输出 σ_i , 表示noise的variance, 需要再输入exp函数进行计算, 保证其值是大于0的, 是machine自己学习的

Why VAE?

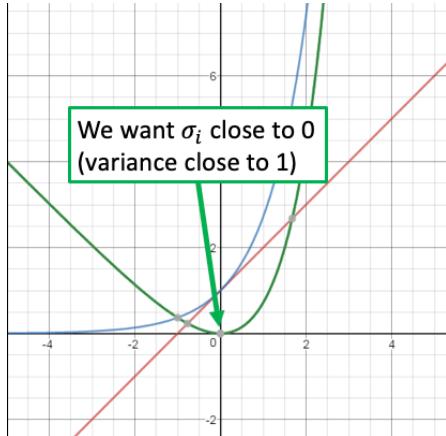
What will happen if we only minimize reconstruction error?



但只有noise是不够的, 还需要加一些限制

$$\sum_{i=1}^3 (exp(\sigma_i) - (1 + \sigma_i) + (m_i)^2)$$

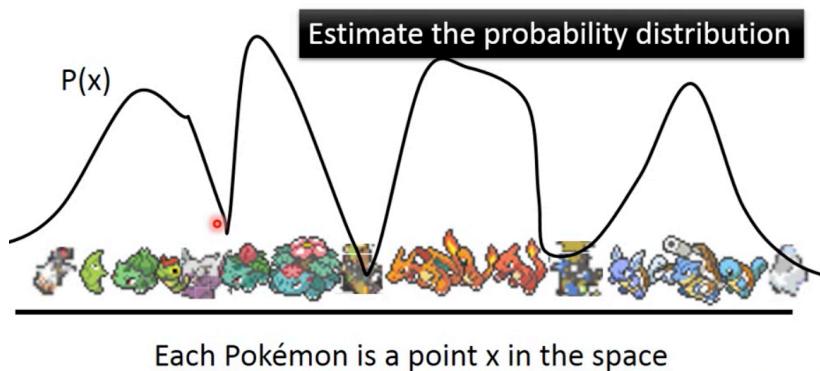
在下图中, 蓝色的线表示 $exp(\sigma_i)$, 红色的线表示 $(1 + \sigma_i)$, 绿色的线表示两者的差值, 可以发现在原点处差值是最小的, σ_i 的值接近1, variance的值接近1



$(m_i)^2$ 为正则项 regularization term, 让结果不那么 overfitting

在下图中, 为一个高维的坐标 (用一维进行了展示), 曲线表示 image 是宝可梦的概率, 可以在图中看到, 对于一些很像宝可梦的 image, $P(x)$ 值很高, 但对于一些比较模棱两可的 image, $P(x)$ 值就很低

- Back to what we want to do



那么我们到底要怎么来评估这个 probability distribution 呢? 答案是高斯混合模型

Gaussian Mixture Model

对于下图中的曲线图, 蓝色表示多个高斯模型, 黑色曲线表示这些模型通过一定的 weight 进行叠加之后的结果

现在有很多个 gaussian model (1, 2, 3, 4, 5...), 且都对应了自己的 weight $P(m)$ (蓝色方块), 要先根据 weight 选对应的 gaussian, 再决定到底要从 gaussian 中 sample 哪些 data; 我们使用 $P(m)$ 表示每个 gaussian 的 weight, $P(x|m)$ 表示从对应的 gaussian 中选出 data x 的概率

x 并不是代表着某个类别, 而是用一个 vector 来进行表示, 每个维度表示不同的特征, 即 Distributed representation is better than cluster

Gaussian Mixture Model

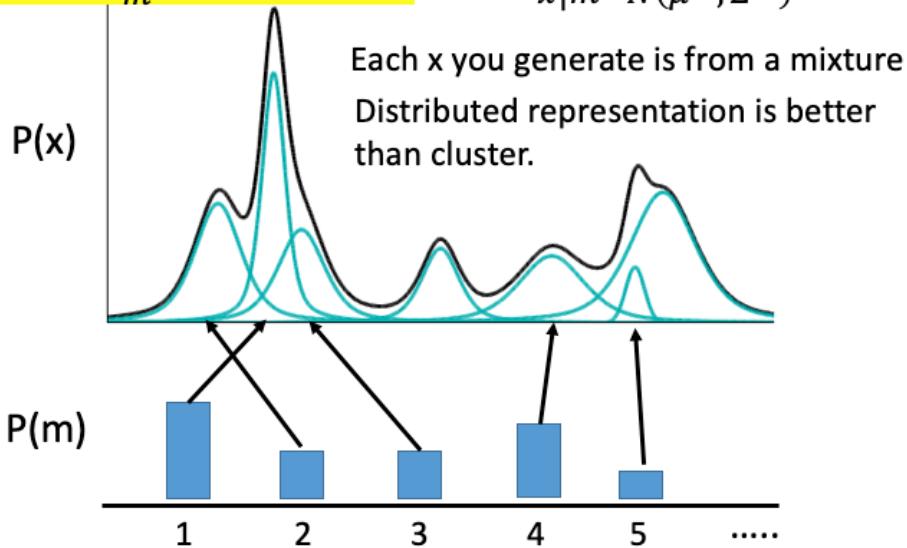
$$P(x) = \sum_m P(m)P(x|m)$$

How to sample?

$m \sim P(m)$ (multinomial)

m is an integer

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



我们现在从正态分布中sample一个data z , z 的每个dimension就表示某种attribute;

根据 z 我们就可以得出高斯分布mean和variance, 即 $\mu(z), \sigma(z)$, z 有无穷多个可能, mean和variance也有无穷多个可能

VAE

$$z \sim N(0, I)$$

z is a vector from normal distribution

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

Each dimension of z represents an attribute

$z \rightarrow \text{NN} \quad \begin{matrix} \mu(z) \\ \sigma(z) \end{matrix}$

P(x)

$$P(x) = \int_z P(z)P(x|z)dz$$

Even though z is from $N(0, I)$, $P(x)$ can be very complex

Infinite Gaussian

z

在最下方的图中, 我们可以认为最中间的圆点被sample到的几率最大, 其他圆点被sample到的几率就相对较小; 每个 z 被sample到之后, 根据某个function, 计算出对应的mean和variance, 都对应着不同的gaussian model

这个function可以是一个neural network, input为 z , output为 $\mu(z), \sigma(z)$

由于现在是连续的 z , $P(z)$ 的形式也发生了变化,

$$P(x) = \int_z P(z)P(x|z)dz$$

Maximizing Likelihood

现在我们需要找到 z 和 $\mu(z), \sigma(z)$ 之间的关系，用network来进行计算的时候也需要一个评估手段，这里我们采用的是最大化 $L = \sum_x \log P(x)$ ，即最大化我们已经看到的image x 的likelihood；

那么我们现在就有一个NN的参数需要调整，来最大化likelihood L；

Maximizing Likelihood

$$P(x) = \int_z P(z)P(x|z)dz$$

$$L = \sum_x \log P(x)$$

$P(z)$ is normal distribution

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

$\mu(z), \sigma(z)$ is going to be estimated

Maximizing the likelihood of the observed x

Tuning the parameters to
maximize likelihood L



We need another
distribution $q(z|x)$

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



现在我们还有另外一个distribution $q(z|x)$ ，和前一个是相反的，其中 $z|x$ 表示给出 x ，再把 x 输入网络 NN' ，得到属于 z 的gaussian distribution，其mean和variance，即 $\mu'(x), \sigma'(x)$

$P(x|z)$ 表示 z 决定了 x 的distribution， $q(z|x)$ 表示 x 决定了 z 的distribution

Maximizing Likelihood

$$P(x) = \int_z P(z)P(x|z)dz$$

$P(z)$ is normal distribution

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

$\mu(z), \sigma(z)$ is going to be estimated

$$L = \sum_x \log P(x) \quad \text{Maximizing the likelihood of the observed } x$$

$$\log P(x) = \int_z q(z|x) \log P(x) dz \quad q(z|x) \text{ can be any distribution}$$

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

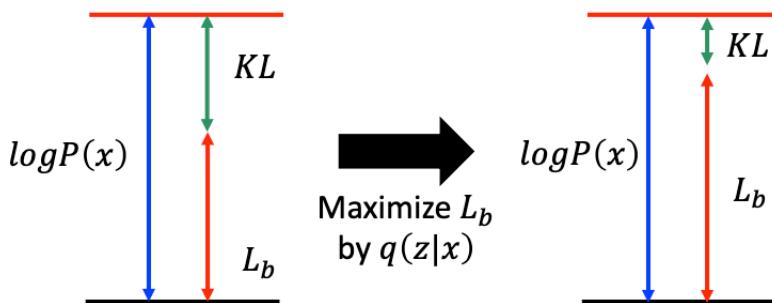
$$= \int_z q(z|x) \log \left(\frac{P(z,x)}{q(z|x)} \right) dz + \int_z q(z|x) \log \left(\frac{q(z|x)}{P(z|x)} \right) dz$$

$$\geq \int_z q(z|x) \log \left(\frac{P(x|z)P(z)}{q(z|x)} \right) dz \quad \text{lower bound } L_b \quad \geq 0$$

其中 $P(z,x) = P(x)P(z|x)$

$$\log P(x) = L_b + KL(q(z|x)||P(z|x))$$

$$L_b = \int_z q(z|x) \log \left(\frac{P(x|z)P(z)}{q(z|x)} \right) dz \quad \text{Find } P(x|z) \text{ and } q(z|x) \text{ maximizing } L_b$$



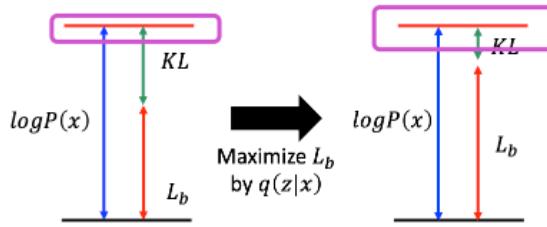
$q(z|x)$ will be an approximation of $p(z|x)$ in the end

我们本来需要调整的参数是 $P(x|z)$, 使得 $P(x)$ 取得最大值, 从而使 likelihood 取得最大值; 但现在的 lower bound 为 L_b , 是 $\log(P(x))$ 的最小值, 其中 $P(z)$ 是已知的, 不知道的参数是 $P(x|z), q(z|x)$, 两项都需要调整

Q: 这里为什么突然多了一项需要调整的参数?

A: 这里并不知道 lower bound 和 likelihood 之间的关系到底是怎样的, 有可能升高了 lower bound, 但 likelihood 反而下降了; 引入 q 就可以解决这个问题

根据原式子，即 $P(x) = \int_z P(z)P(x|z)dz$ ，只和 $P(x|z)$ 有关，与 $q(z|x)$ 是无关的，即下图中 **方框** 部分是不变的，



如果我们现在固定 $P(x|z)$ ，想通过 $q(z|x)$ 来使 L_b 最大化，那么对应的 **KL divergence** 就会最小化，到一种很极端的情况，**KL divergence** 会变为 0；如果 L_b 继续上升，那么肯定会超出该区域，因此对应的 $\log P(x)$ 也会继续上升；**KL divergence** 不断变小的过程， $q(z|x), P(z|x)$ 之间的差距也会不断缩小

因此，这个过程不仅让 likelihood 变得越来越大，也会找到和 $P(x|z)$ 接近的 $q(z|x)$

$$\begin{aligned}
 L_b &= \int_z q(z|x) \log \left(\frac{P(z,x)}{q(z|x)} \right) dz = \int_z q(z|x) \log \left(\frac{P(x|z)P(z)}{q(z|x)} \right) dz \\
 &= \underbrace{\int_z q(z|x) \log \left(\frac{P(z)}{q(z|x)} \right) dz}_{-KL(q(z|x)||P(z))} + \int_z q(z|x) \log P(x|z) dz \\
 &\quad z|x \sim N(\mu'(x), \sigma'(x)) \\
 &\quad x \rightarrow \text{NN'} \quad \begin{matrix} \nearrow \mu'(x) \\ \searrow \sigma'(x) \end{matrix}
 \end{aligned}$$

对于 $q(z|x)$ ，其中 $z|x$ 表示给出 x ，再把 x 输入网络 NN' ，得到属于 z 的 gaussian distribution，就可以知道 z 是从什么样的 gaussian distribution 得出来的

Connection with Network

现在我们的目标就是最小化 $q(z|x), P(z)$ 之间的 **KL divergence**，即使 $q(z|x)$ 和 normal distribution $P(z)$ 越接近越好，即 minimize

$$\sum_{i=1}^3 (\exp(\sigma_i) - (1 + \sigma_i) + (m_i)^2)$$

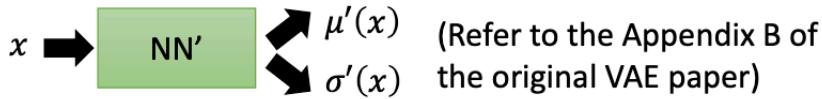
还需要 maximizing

$$P(x) = \int_z P(z)P(x|z)dz$$

Connection with Network

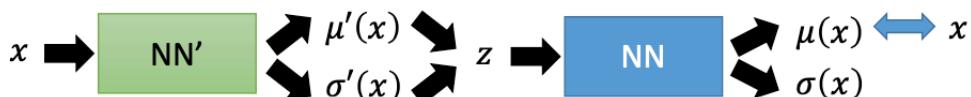
Minimizing $KL(q(z|x)||P(z))$

$$\text{Minimize} \quad \sum_{i=1}^3 (\exp(\sigma_i) - (1 + \sigma_i) + (m_i)^2)$$



Maximizing

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



This is the auto-encoder

对于input的x，我们先输入网络 NN' ，得到获取z的gaussian distribution；sample得到z之后，再输入网络 NN ，得到获取x的gaussian distribution，使得新分布的mean和x越接近越好

KL散度

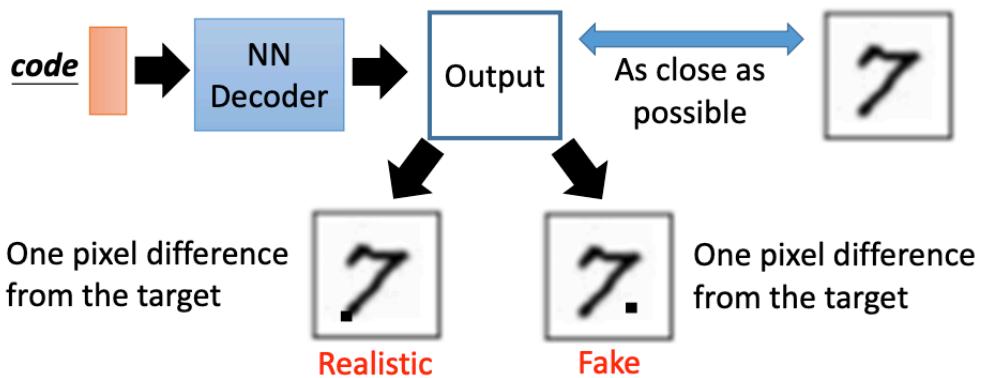
所谓KL散度，是指当某分布 $q(x)$ 被用于近似 $p(x)$ 时的信息损失。

计算公式为

$$KL(p||q) = \sum p(x) \log \frac{p(x)}{q(x)}$$

Problems of VAE

- It does not really try to simulate real images



VAE may just memorize the existing images, instead of generating new images

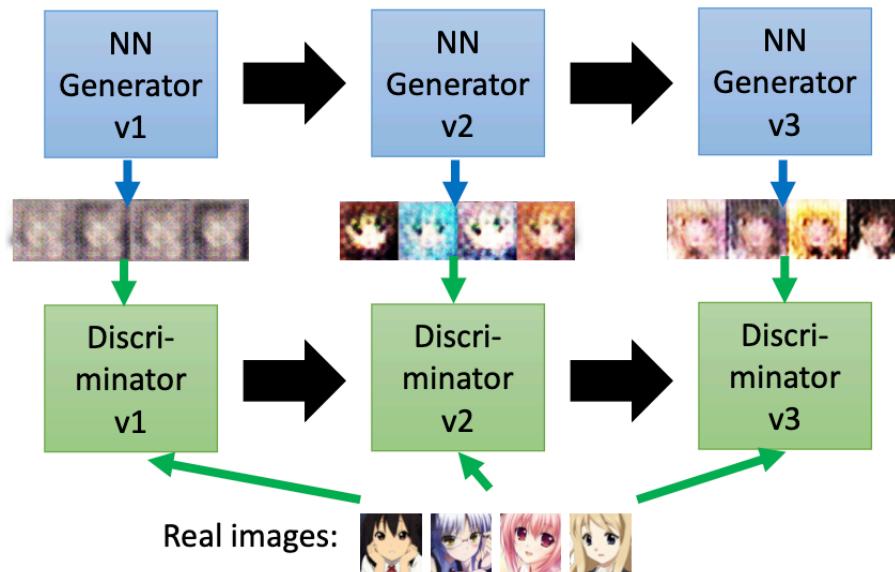
GAN

The evolution of generation

v1: 会generate一些奇怪的图片，然后会有一个第一代的discriminator，来辨别到底哪一个是real image；

V2: generator会根据上一次discriminator的结果进行调整，第二代生成的image就和真实的image更像了，再把image输入第二代discriminator，再进行比较；

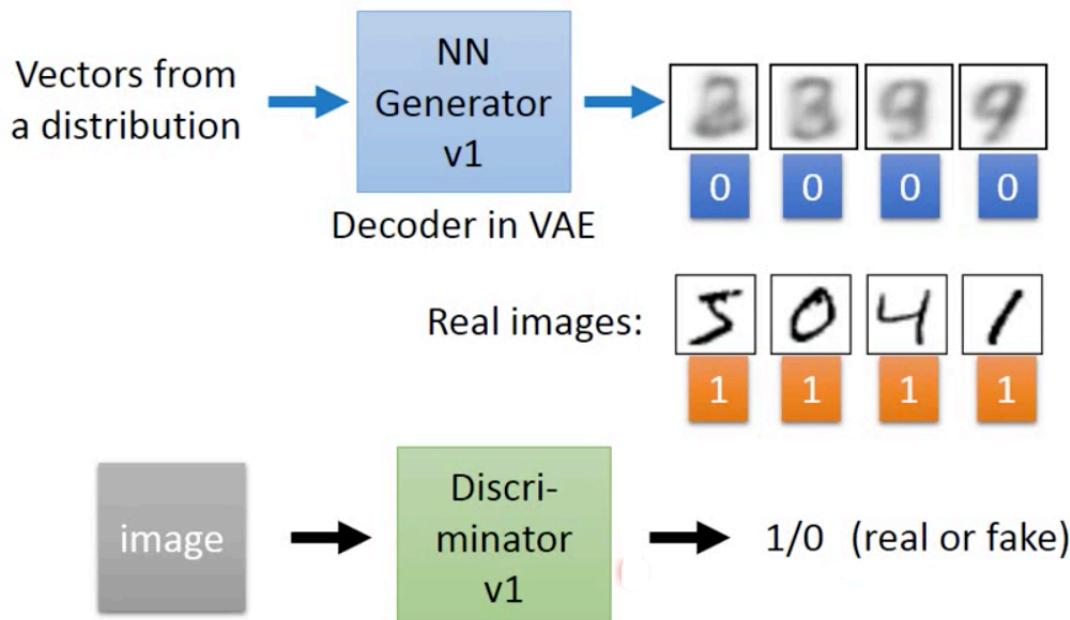
.....



Discriminator

对于generator，可以看作是VAE里面的decoder，我们通常选择从某个distribution sample出来的vector，输入到第一代的generator，再生成相对应的image，有多少个vector，就生成多少个image；

把真实的image输入discriminator，与generator生成的image进行比较；并对这些data进行标注，real image标注为1，其余标注为0



Generator

首先随机sample一个vector，作为Generator的输入，generator会生成对应的image，再把这个image输入discriminator，输出是real image的概率，对于第一代（v1），generator还不成熟，因此概率只有0.87；generator接下来会调整自己的参数，使discriminator认为其生成的image是real image；

这generator+discriminator就相当于一个network，整体目标是使discriminator认为generator生成的image是real image，就可以根据这个指标来使用梯度下降算法，进行back propagation，来不断调整generator的参数；

注：在训练过程中应该固定discriminator的参数，来调整generator的参数

GAN - Generator

“Tuning” the parameters of generator

→ The output be classified as “real”
(as close to 1 as possible)

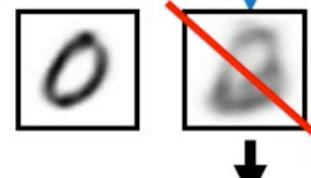
Generator + Discriminator
= a network

Using gradient descent to find the
parameters of generator

Fix the discriminator

Randomly
sample a vector

NN
Generator
v1



Discrim-
inator
v1

1.0

0.87

Why GAN is hard to train?

- No explicit signal about how good the generator is
 - In standard NNs, we monitor loss
 - In GANs, we have to keep “well-matched in a contest”
- When discriminator fails, it does not guarantee that generator generates realistic images
 - Just because discriminator is stupid
 - Sometimes generator finds a specific example that can fail the discriminator
 - Making discriminator more robust may be helpful.