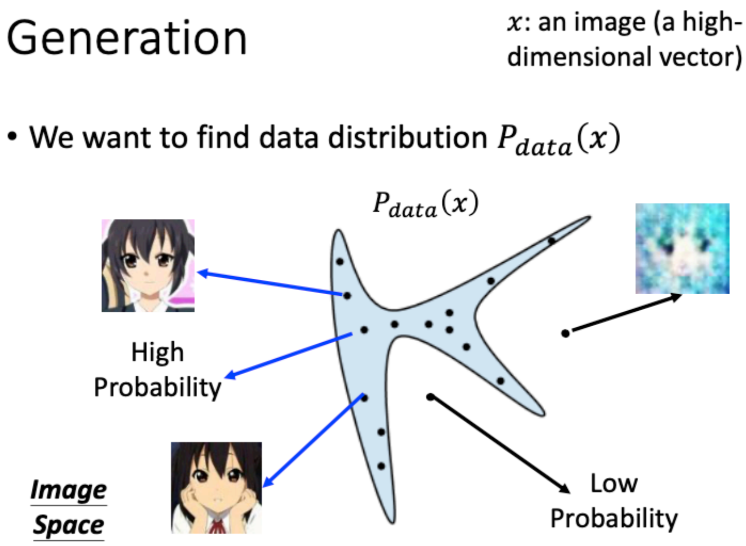
本文主要介绍了GAN的基础理论。还对似然函数和KL散度的关系进行了推导。

Generation

现在我们用x来表示一张图像，是一个高维的vector，比如图像大小是$64*64$维的，那么vector的维数就是$64*64$。每张图像都是这个高维空间中的一个点。为了方便展示，下图中我们假设图像是二维空间中的一个点。

对于我们要产生的图像，有一个固定的distribution $P_{data}(x)$。在整个图像所构成的高维空间中，只有一小部分sample出来的图像和人脸接近，其他部分都不像人脸。比如我们从下图中蓝色的distribution中进行sample，看起来就很像是人脸，在其他区域就不像人脸。

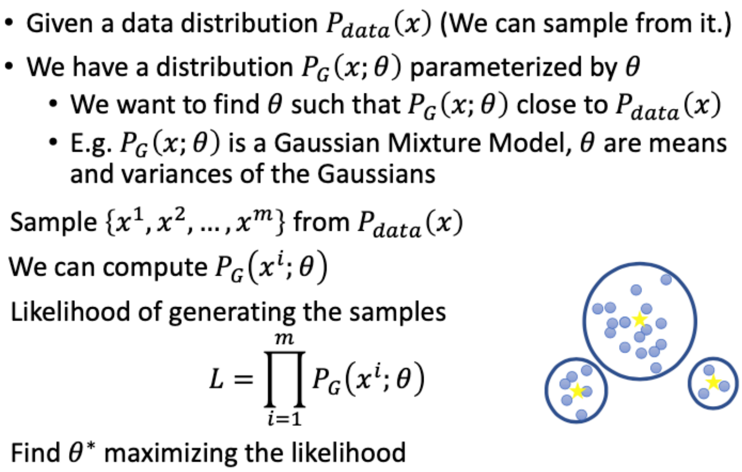
那么我们现在的目标就是找到这个distribution。



在有GAN之前，我们通常用Maximum Likelihood Estimation来做这件事。

Maximum Likelihood Estimation

1. 我们可以从这个distribution中sample图像，但我们并不知道其对应的formula长什么样子；
2. 那么我们现在就可以找到另外一个distribution $P_G(x;\theta)$，比如其对应的参数可以是$\mu,\sum$，来使这个distribution的参数和原来的相接近。



具体做法如下，

- 先从原来的distribution中sample出$\{x^1, x^2, ... x^m\}$；
- 把x^i代入现在的已知的distribution $P_G(x^i;\theta)$，表示x^i是从现在这个distribution中sample出来的概率；
- 把这些概率相乘，得到似然函数L；最后找到对应的参数θ，使似然函数取得最大值。

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

Minimize KL Divergence

最大似然估计也就等同于来最小化KL divergence。

$$\begin{aligned}\theta^* &= arg \max_\theta \prod_{i=1}^{m} P_G(x^i;\theta) = arg \max_\theta log \prod_{i=1}^{m} P_G(x^i;\theta) \\ &= arg \max_\theta \sum_{i=1}^{m} logP_G(x^i;\theta) \quad \boxed{\{x^1, x^2, \dots, x^m\} \text{ from } P_{data}(x)} \\ &\approx arg \max_\theta E_{x \sim P_{data}}[logP_G(x;\theta)] \\ &= arg \max_\theta \int_x P_{data}(x) logP_G(x;\theta)dx - \int_x P_{data}(x) logP_{data}(x)dx \\ &= arg \min_\theta KL(P_{data}||P_G) \quad \text{How to define a general } P_G\text{?}\end{aligned}$$

现在我们的问题是找到参数θ^*，使得$E_{x\sim P_{data}}[logP_G(x;\theta)]$可以取得最大值。$\{x^1, x^2, ... x^m\}$是从distribution P_{data}中sample出来的，我们把这里的$E_{x\sim P_{data}}$展开，从离散值变到连续值，即

$$E_{x\sim P_{data}}[logP_G(x;\theta)] = \int_x P_{data}(x)[logP_G(x;\theta)]dx$$

由于我们的目标是找到P_G分布对应的参数，现在加入一个常数项$\int_x P_{data}(x)[logP_{data}(x;\theta)]$，对最大化的问题也不会产生影响，即

$$\begin{aligned}&arg \max_\theta E_{x\sim P_{data}}[logP_G(x;\theta)] \\ =&arg \max_\theta \int_x P_{data}(x)[logP_G(x;\theta)]dx \\ =&arg \max_\theta \int_x P_{data}(x)[logP_G(x;\theta)]dx - \int_x P_{data}(x)[logP_{data}(x;\theta)]dx \\ =&arg \max_\theta \int_x P_{data}(x)[logP_G(x;\theta) - logP_{data}(x;\theta)]dx \\ =&arg \max_\theta \int_x P_{data}(x)[log\frac{P_G(x;\theta)}{P_{data}(x;\theta)}]dx \\ =&arg \max_\theta - \int_x P_{data}(x)[log\frac{P_{data}(x;\theta)}{P_G(x;\theta)}]dx \\ =&arg \min_\theta \int_x P_{data}(x)[log\frac{P_{data}(x;\theta)}{P_G(x;\theta)}]dx \\ =&arg \min_\theta KL(P_{data}||P_G)\end{aligned}$$

就把这个最大化似然函数问题转化为了最小化KL divergence的问题。

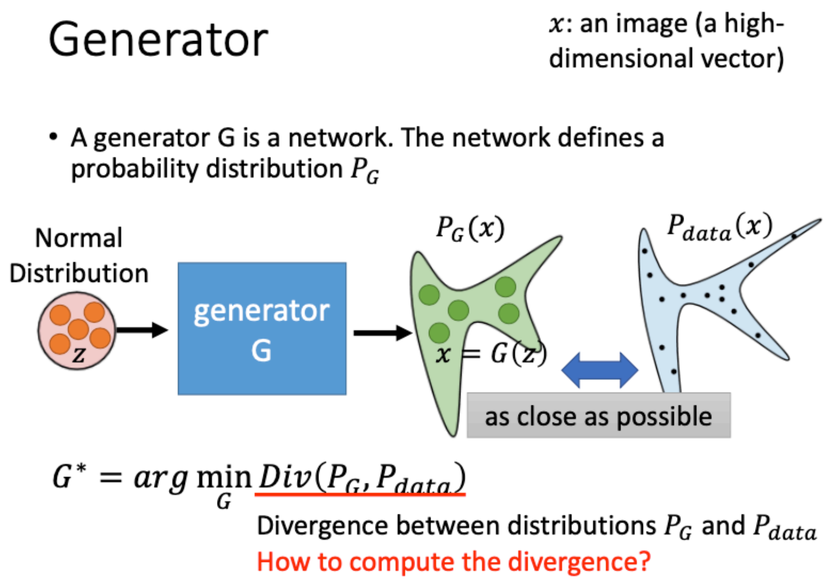
那么我们如何来定义P_G的表达式呢?

首先P_G是类似于高斯分布这样的distribution，很容易计算出其对应的likelihood；但如果是neural network这样的distribution，就很难算出这个likelihood。

Generator

现在分布是neural network，如果我们还是用高斯分布的公式来进行调整，那么我们不管怎么变化mean和variance，其分布都不可能和neural network相接近。因此我们需要一个更加general的方式来学习generative这件事。

现在我们有一个generator G，input z是从normal distribution中sample出来的，output为$x = G(z)$，不同的z就会有不同的x，x就组成了一个新的distribution $P_G(x)$。这个distribution可以非常复杂，比如neural network。



我们希望通过G得出的这个distribution $P_G(x)$，与目标$P_{data}(x)$可以越接近越好，即P_G, P_{data}之间的divergence可以越小越好，可以是KLdivergence，也可以是其他的divergence，即

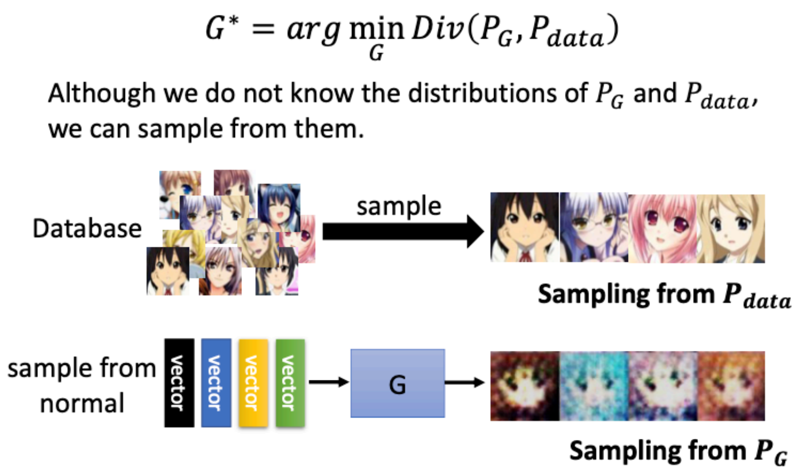
$$G^* = arg\min_\theta Div(P_G, P_{data})$$

那么我们怎么来minimize这个divergence呢?

如果我们知道P_G, P_{data}的formulation，那么我们就可以计算出divergence，再使用gradient descent算法。但现在我们并不知道他们的formulation，就不能使用gradient descent算法。

Discriminator

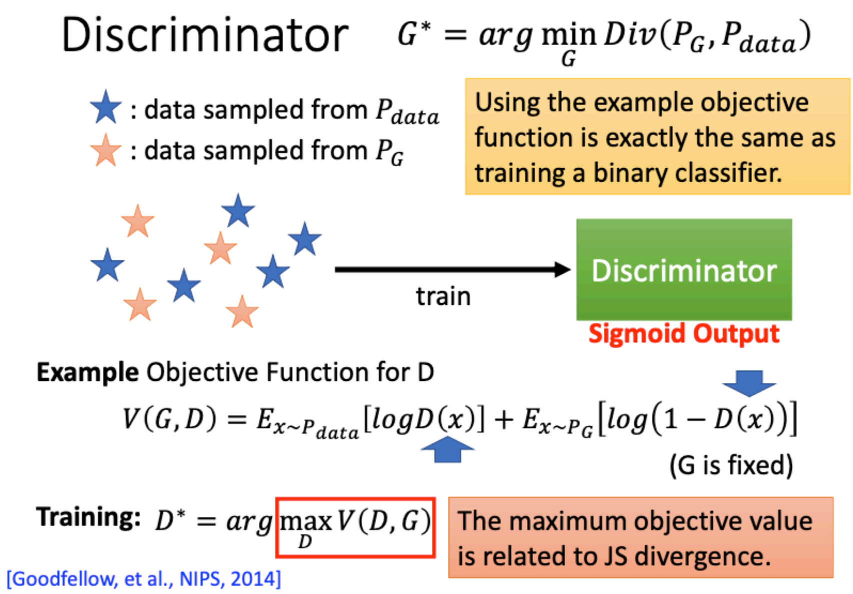
虽然我们并不知道这两者的distribution，但我们可以从这两个分布sample 很多data出来。



从这两个分布中sample很多data出来，又如何来计算分布之间的divergence呢?

我们可以使用GAN的discriminator来完成这个任务。

在下图中，我们使用蓝色星星表示从P_{data}中sample出来的数据，红色星星表示从P_G中sample出来的数据。再来训练我们的discriminator D，D对P_{data}中sample出来的数据会给高分，从P_G中sample出来的数据给低分。这个训练的结果就可以告诉我们P_G, P_{data}之间的divergence。



先fix掉G的参数，再来训练discriminator D，D得出的分数越大越好。找到D^*，使得$V(G,D)$最大化。

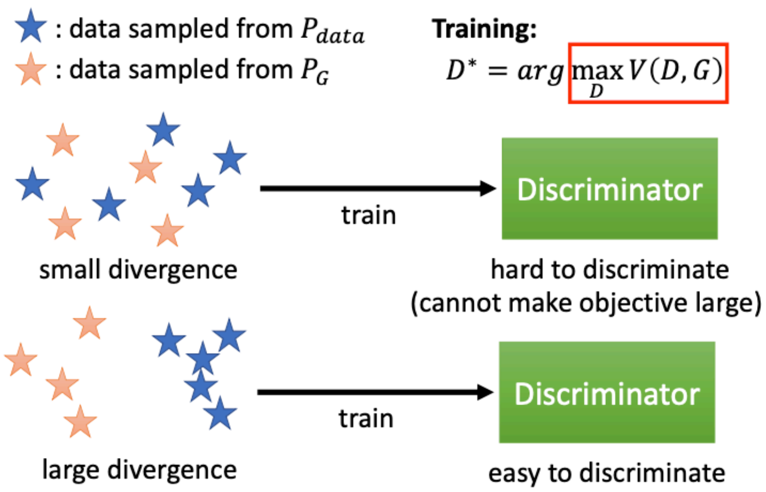
$$V(G,D) = E_{x\sim P_{data}}[log(D(x))] + E_{x\sim P_G}[1 - log(D(x))]$$

其中$E_{x\sim P_{data}}[log(D(x))]$表示真实图像所得到的分数，D的目标就是使真实图像获得的分数越大越好；而$E_{x\sim P_G}[log(D(x))]$表示G生成的图像所得到的分数，应该越小越好，所以前面加了负号。

这个$V(D,G)$的表达式其实和二分类的问题是一样的，红色星星表示class 1，蓝色星星表示class 2，discriminator的任务就是对这两个class进行分类，来最小化cross entropy，也就相当于在解这个问题

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

我们最后找到的D^*，能使objective function $V(D,G)$取得最大值。这个objective function和divergence是有一定关系的。如果这两个类别之间很接近、很难区分，分类器训练的时候loss就会很大，对应的V的值就会很低，对应的divergence的值也会很低；如果这两个类别很好区分，discriminator就很容易找到D^*，使得V取得很大的值，从而divergence的值就会很大。



V(G,D)和divergence之间的关系

根据大数定律，

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

其中我们假设$D(x)$可以是任意函数，现在我们的目标是找到最优值D^*使V最大化。我们可以把积分中的x分开来算，对于其中的任意一个x，都可以分配一个最好的D函数。那么现在的问题就变成：对于给定的x，来找到最优值D^*使V最大化，即最大化

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

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

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

- Given G, what is the optimal D* maximizing

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

Assume that D(x) can be any function

- Given x, the optimal D* maximizing

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

现在我们用$a = P_{data}, b = P_G, D = D(x)$，来简化式子，即找到$D^*$来最大化

$$
f(D) = \mathrm{a}\; log(D) + \mathrm{b}\; log(1-D)
$$

对D求导，并令成0，可以得到

$$D^* = \frac{a}{a+b} \quad \rightarrow \quad D^*(x) = \frac{P_{data}}{P_{data}+P_G}$$

$$\max_D V(G,D) \qquad \boxed{\begin{aligned} V = & E_{x\sim P_{data}}[logD(x)] \\ & +E_{x\sim P_G}[log(1-D(x))] \end{aligned}}$$

- **Given x, the optimal D* maximizing**

$$\underset{a}{P_{data}(x)}\,log\underset{D}{D(x)} + \underset{b}{P_G(x)}\,log(1-\underset{D}{D(x)})$$

- **Find D* maximizing:** $f(D) = alog(D) + blog(1-D)$

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

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

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

把$D^*(x)$代入objective function，可以得到

$$\begin{aligned} & maxV(G,D) = V(G,D^*) \\ & = E_{x\sim P_{data}}[log\frac{P_{data}}{P_{data}+P_G}] + E_{x\sim P_G}[log\frac{P_G}{P_{data}+P_G}] \\ & = \int_x [P_{data}(x)log\frac{P_{data}}{P_{data}+P_G} + P_G(x)log\frac{P_G}{P_{data}+P_G}]dx \end{aligned}$$

对式子中log部分的分子分母同时除以2，把分子上的1/2提出来，可得

$$\begin{aligned} & maxV(G,D) \\ & = -2log2 + \int_x [P_{data}(x)log\frac{P_{data}}{(P_{data}+P_G)/2} + P_G(x)log\frac{P_G}{(P_{data}+P_G)/2}]dx \\ & = -2log2 + KL(P_{data}||\frac{P_{data}+P_G}{2}) + KL(P_G||\frac{P_{data}+P_G}{2}) \\ & = -2log2 + 2JSD(P_{data}||P_G) \qquad (1) \end{aligned}$$

也就得到了我们的 Jensen-Shannon divergence，即JSD。

我们希望通过G得出的这个distribution $P_G(x)$，与目标$P_{data}(x)$可以越接近越好，即P_G, P_{data}之间的divergence可以越小越好，即

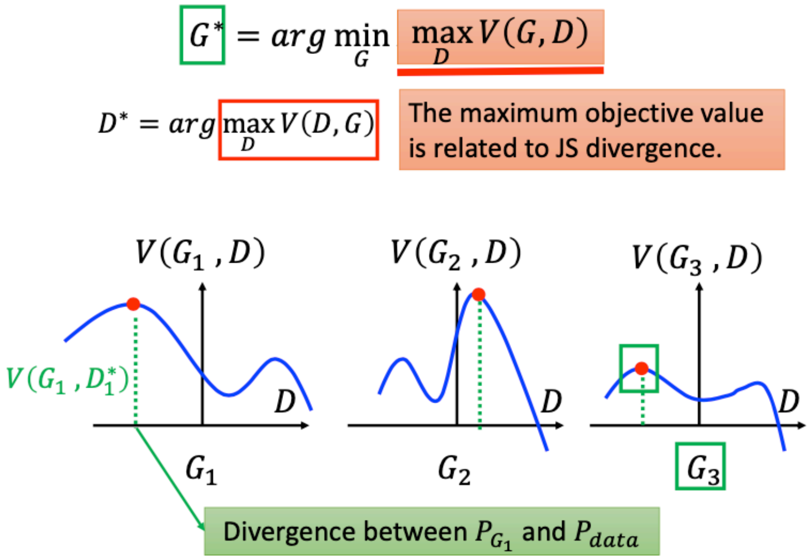
$$G^* = arg\min_G Div(P_G, P_{data})$$

那么我们到底怎么算P_G, P_{data}之间的divergence $Div(P_G, P_{data})$呢?

根据公式(1)，我们知道了JS divergence和$maxV(G,D)$之间的关系，是成正比的。那么现在我们找到D，使objective function取得最大值，这个最大值就是divergence。那么现在的式子就变成了，

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

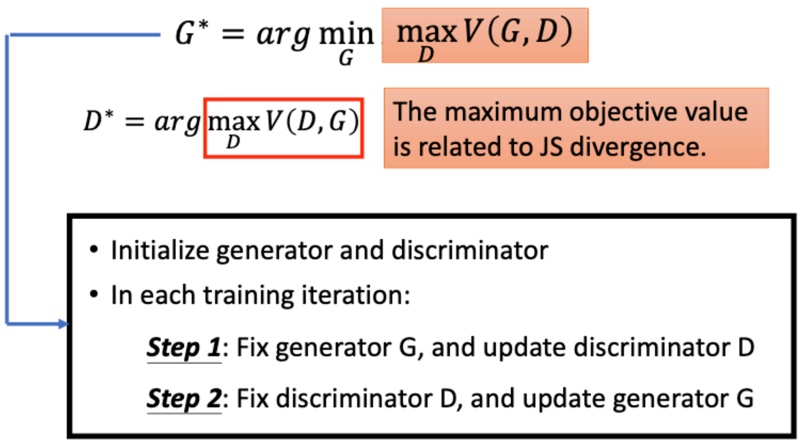
其中要最大化discriminator得出的分数，最小化generator生成的数据与P_{data}之间的差距。



假设现在只有三个generator G_1, G_2, G_3可供选择，对应的三个objective function变化如上图所示，横坐标表示选择了不同的discriminator，纵坐标表示$V(G_i, D)$的值。第一幅图表示选择固定G1，变化discriminator，V的值的变化曲线。

图中红色圆点所在的横坐标，表示$V(G_i, D)$值最大的位置，一共有三个。现在已经找到了使V最大的discriminator，接下来需要找使$maxV(G, D)$最小的generator（G_1, G_2, G_3）。毫无疑问是第三幅图中的G_3，是可以使得$maxV(G, D)$最小的generator。

红色圆点的纵坐标$V(G_i, D)$表示P_{G_i}, P_{data}之间的divergence，也是第三幅图中的divergence最小。



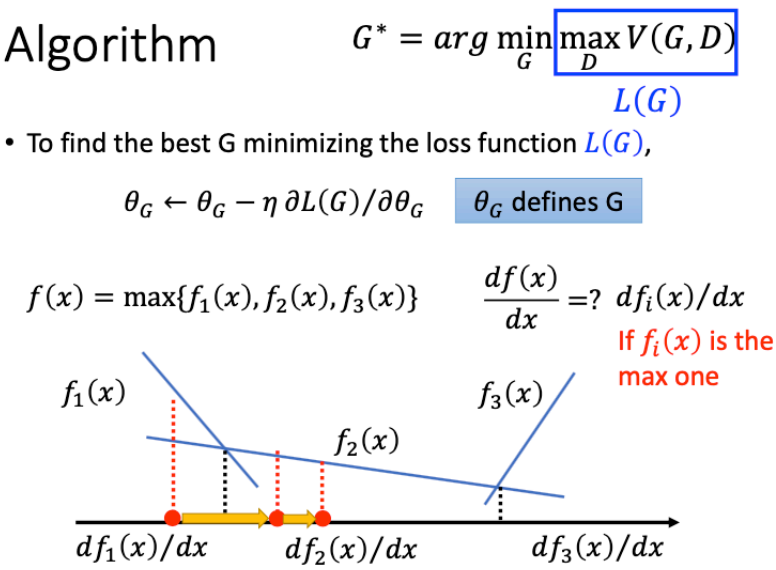
其实GAN的两个训练步骤就是在解决这个最大最小化问题。

Algorithm

我们把目标式子简化一下，现在D是一个给定的值，可以让$V(G, D)$的值最大化，$\max_D V(G, D)$可以表示为$L(G)$，即找到G^*，

$$G^* = arg \min_G L(G)$$

先计算出gradient $\frac{\partial L(G)}{\partial \theta_G}$，再来更新$\theta_G$的参数。



Q：其中$L(G)=\max_D V(G,D)$，我们可以对这个函数求微分吗？

A：可以。如果现在有一个函数$f(x)=max\{f_1(x),f_2(x),f_3(x)\}$，其对应的函数图像如上图所示，是个分段函数。我们假设$f_1(x)$的函数值是最大的，那么$\frac{df(x)}{dx}=\frac{df_1(x)}{dx}$，梯度对应的是在该区域内，函数值最大的那个梯度。更次参数更新都要注意自己在哪个region内，不同的region求导的函数不一样。

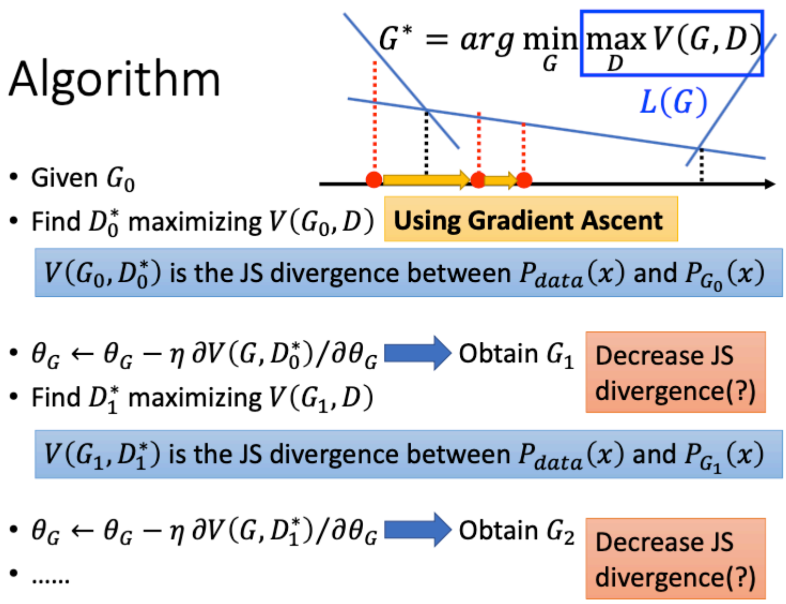
具体的算法流程如下：

- 给定一个初始的generator G_0；
- 找到D_0^*，使得$V(G_0,D)$的值最大化；
- 得到$L(G)$之后，就可以对整个式子求微分，得出对应的梯度，更新generator的参数，就得到新的generator G_1；

得到新的generator后，很可能已经进入了下一个region，因此还需要重新计算discriminator，

- 此时objective function为$V(G_1,D)$，现在是D_1^*使V取得最大值；
- $L(G)=V(G_1,D_1^*)$，再更新generator的参数。

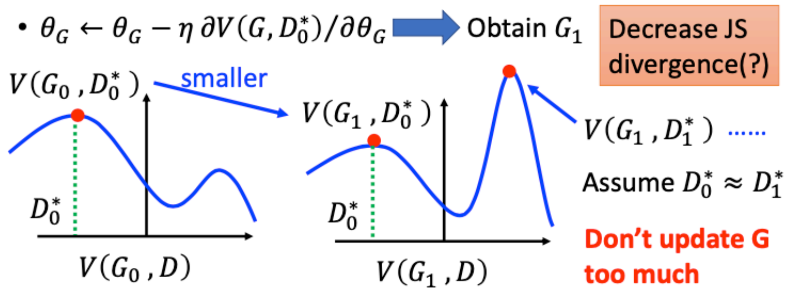
……



更新参数这个过程到底是不是在减小JS divergence呢?

首先我们找到了D_0^*，使V取得最大值$V(G_0,D_0^*)$，也就是JS divergence取得最大值；在generator更新参数之后，objective function也发生了变化$V(G_0,D) \rightarrow V(G_1,D)$，$D_0^*$对应的$V(G_0,D_0^*)$并不是现在的最大值，而$D_1^*$对应的$V(G_0,D_1^*)$才是最大值。从图中可以看出，$V(G_0,D_1^*) < V(G_0,D_0^*)$，对应的JS divergence反而减小了。

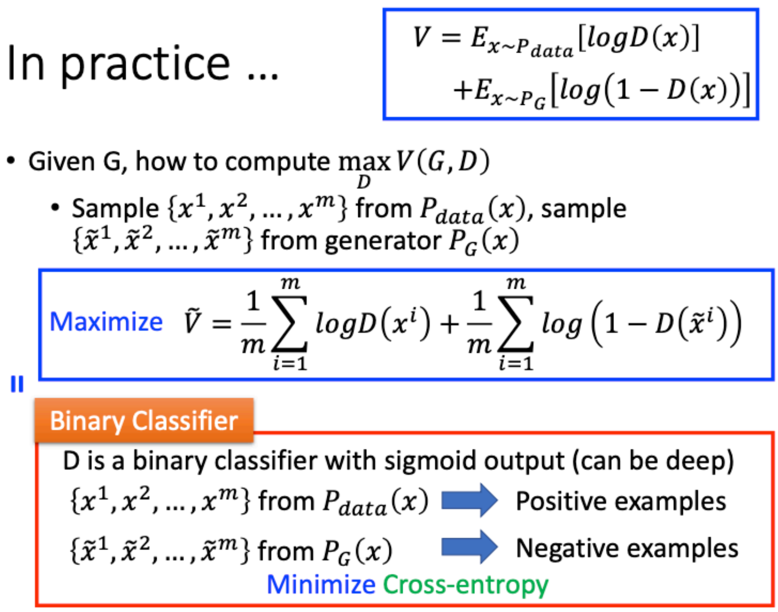
如果generator的参数变化不大，即$D_0^* \approx D_1^*$，我们就可以把这个过程看作是在减小divergence。在实际的操作中，我们应该使G迭代的次数减少， 使D迭代的次数增加。



In practice ...

fix G的参数，得到G生成的图像$\{\tilde{x}^1,\tilde{x}^2,...,\tilde{x}^m\}$，再输入discriminator D，不断调整$\theta_d$，使得得到的分数越大越好，

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



这个discriminator其实就是在做binary classifier；

这是整个算法的步骤：

Learning D：首先从数据库中取出m个真实图片，再根据一个分布随机产生m个vector作为输入$\{z^1,z^2,...,z^m\}$，此时fix G的参数，得到G生成的图像$\{\tilde{x}^1,\tilde{x}^2,...,\tilde{x}^m\}$，再输入discriminator D，不断调整$\theta_d$，使得得到的分数越大越好。V的值最大的时候，divergence的值才越小。

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

其中$D(x^i)$表示真实图像所得到的分数，D的目标就是使真实图像获得的分数越大越好；而$D(\tilde{x}^i)$表示G生成的图像所得到的分数，应该越小越好，所以前面加了负号。求出梯度$\Delta\tilde{V}(\theta_d)$，再更新$\theta_d$的值，

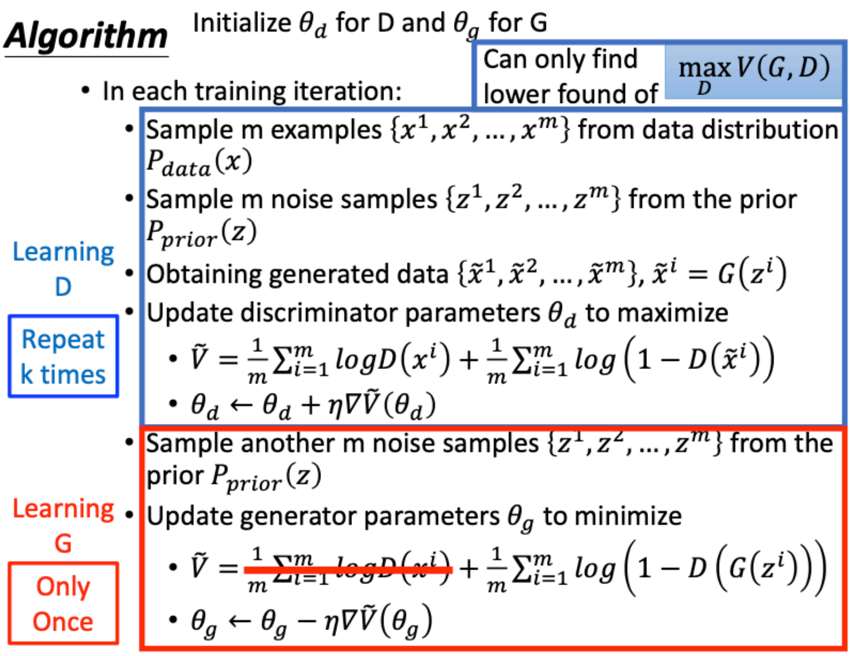
$$\theta_d \leftarrow \theta_d + \eta\Delta\tilde{V}(\theta_d)$$

Learning G：把D训练好之后，我们就可以fix D，来训练generator G的参数。首先也需要从分布中随机生成一些噪声z，再输入G，$G(z^i)$再输入D，得到相对应的分数，G的目标是想办法骗过D，不断调整参数θ_g，使下面的objective function最小化，generator不能train太多次，通常update一次就好。

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

求出梯度$\Delta\tilde{V}(\theta_g)$，再更新$\theta_g$的值，

$$\theta_g \leftarrow \theta_g - \eta\Delta\tilde{V}(\theta_g)$$



Objective Function for Generator in Real Implementation

实际上，我们使用$V = E_{x \sim P_G}[-log(D(x))]$，可以更方便进行code。

Objective Function for Generator in Real Implementation

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

Slow at the beginning

Minimax GAN (MMGAN)

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

Real implementation:
label x from P_G as positive

Non-saturating GAN (NSGAN)