$f$gan:使用变分训练生成神经采样器散度最小化

Sebastian Nowozin
Machine Intelligence and Perception Group
Microsoft Research
Cambridge, UK
Sebastian.Nowozin@microsoft.com

Botond Cseke
Machine Intelligence and Perception Group
Microsoft Research
Cambridge, UK
botcse@microsoft.com

Ryota Tomioka
Machine Intelligence and Perception Group
Microsoft Research
Cambridge, UK
ryoto@microsoft.com

Abstract

生成神经采样器是实现采样的概率模型使用前馈神经网络:它们取一个随机输入向量并产生由网络权重定义的概率分布中的样本。这些模型具有很强的表达能力，可以有效地计算样本和衍生品，但不能用于计算可能性或边缘化。生成对抗训练方法允许训练这样的模型通过使用辅助判别神经网络。我们证明了生成对抗方法是一个特例现有比较通用的变分散度估计方法。我们证明了任何f -散度都可以使用用于训练生成神经采样器。讨论了发散函数的各种选择对训练的好处生成模型的复杂度和质量。

1 简介

概率生成模型描述a上的概率分布给定域$\mathcal{X}$，例如在自然语言上的分布句子、自然图像或记录波形。

给定一个生成模型Q从一个类$\mathcal{Q}$的可能模型我们通常对执行以下一个或多个操作感兴趣操作:

- 抽样。从Q产生一个样本。通过检验样品或在一组样本上计算函数，我们可以获得重要的见解进入分配或解决决策问题。
- 估计。给定一组iid样本$\{x_1, x_2, \ldots, x_n\}$来自一个未知的真实分布$P$，找到最能描述的$Q \in \mathcal{Q}$ 真实分布。
- 逐点似然评估。给出一个示例x，求值可能性$Q(x)$。

生成对抗网络(GAN)的形式提出通过[10]是一个富有表现力的类生成模型允许精确采样和近似估计。GAN中使用的模型是一个简单的前馈神经网络，它接收a作为输入随机数的向量，例如，从均匀分布中抽样。这个随机输入通过网络的每一层和最后一层层产生所需的输出，例如，图像。显然，从GAN模型中采样是有效的，因为只有一次前向传递需要通过网络产生一个精确的样品。

首先考虑了这种概率前馈神经网络模型在[22]和[3]，这里我们称这些模型为生成神经采样器。GAN也是这种类型，变分的解码器模型也是如此自动编码器[18]。

在GAN的原始论文中，作者证明了估计是可能的神经采样器由近似最小化对称*Jensen-Shannon*散度，

$$D_{\mathrm{JS}}(P\|Q) = \tfrac{1}{2}D_{\mathrm{KL}}(P\|\tfrac{1}{2}(P+Q)) + \tfrac{1}{2}D_{\mathrm{KL}}(Q\|\tfrac{1}{2}(P+Q)), \tag{1}$$

其中D_{KL}表示Kullback-Leibler散度。在GAN训练中使用的关键技术是引入秒同时优化的“ 鉴别器”神经网络。因为$D_{\mathrm{JS}}(P\|Q)$是一个合适的散度度量这意味着真实分布P可以被近似如果有足够的训练样本和模型类$\mathcal{Q}$足够丰富，可以表示P。

在这项工作中，我们证明了gan的原理是更一般的，我们可以扩展了Nguyen提出的变分散度估计框架等[25]来恢复GAN训练客观并将其推广到任意f -分歧。

更具体地说，我们在最先进的技术上做出了以下贡献:

- 我们导出了所有f -散度和提供额外的散度函数作为示例，包括Kullback-Leibler和Pearson散度。
- 对Goodfellow等的鞍点优化过程进行了简化Al. [10]并提供理论正当理由。
- 我们提供了关于散度函数是什么的实验见解适用于估计自然图像的生成神经采样器。

2 方法

我们首先回顾了Nguyen等人的散度估计框架al - [25]基于f -散度。然后我们将这个框架从散度估计扩展到模型估计。

2.1 f散度族

统计差异，如著名的*Kullback-Leibler* 散度度量两个给定概率之间的差分布。一大类不同的分歧就是所谓的分歧f -发散[5, 21]，也是被称为阿里-西尔维距离[1]。给定两个发行版P和Q，它们分别具有绝对的连续密度函数p和q相对于一个基本测量dx 在域$\mathcal{X}$上定义，我们定义f-散度，

$$D_f(P\|Q) = \int_{\mathcal{X}} q(x) f\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x, \tag{2}$$

其中生成器函数$f : \mathbb{R}_+ \to \mathbb{R}$是一个凸，下半连续函数满足$f(1) = 0$。$f$的不同选择将大众的分歧作为特殊情况加以弥补在(2)。我们在表5中说明了常见的选择。参见补充材料更多的分歧和情节。

2.2 f -散度的变分估计

Nguyen等人[25]推导出一般变分方法估计f -散度只给样本从P和Q。我们将从仅仅估计a的散度扩展他们的方法固定模型，估计模型参数。我们将这种新方法称为变分散度最小化(VDM)和证明生成对抗训练是这方面的一个特例通用VDM框架。

为了完整起见，我们首先提供Nguyen et的自包含派生艾尔的散度估计程序。每一个凸的下半连续函数f都有一个凸共轭 功能f^*，也称为*Fenchel* 共轭[14]。这个函数定义为

$$f^*(t) = \sup_{u \in \mathrm{dom}_f} \{ut - f(u)\}. \tag{3}$$

函数f^*也是凸的，并且是下半连续的(f, f^*)与另一个是双重的，因为$f^{**} = f$。因此，我们可以也表示f为$f(u) = \sup_{t \in \mathrm{dom}_{f^*}} \{tu - f^*(t)\}$。Nguyen等人在定义中利用了$f$的上述变分表示对$f$ -散度求散度的下界，

$$\begin{aligned} D_f(P\|Q) &= \int_{\mathcal{X}} q(x) \sup_{t \in \mathrm{dom}_{f^*}} \left\{ t\tfrac{p(x)}{q(x)} - f^*(t) \right\} \mathrm{d}x \\ &\geq \sup_{T \in \mathcal{T}} \left(\int_{\mathcal{X}} p(x)\, T(x)\, \mathrm{d}x - \int_{\mathcal{X}} q(x)\, f^*(T(x))\, \mathrm{d}x \right) \\ &= \sup_{T \in \mathcal{T}} \left(\mathbb{E}_{x \sim P}\left[T(x)\right] - \mathbb{E}_{x \sim Q}\left[f^*(T(x))\right] \right), \end{aligned} \tag{4}$$

其中$\mathcal{T}$是函数的任意类$T : \mathcal{X} \to \mathbb{R}$。上面的推导产生一个下界有两个原因:首先，因为詹森不等式在交换积分和上极值时操作。其次，函数类$\mathcal{T}$可能只包含一个子集在所有可能的函数中。

通过取(4) w.r.t的下界的变化。T，我们发现在温和的条件下f[25], bound是紧的

$$T^*(x) = f'\left(\frac{p(x)}{q(x)}\right), \tag{5}$$

式中f'为f的一阶导数。这种情况可以作为选择f和设计类的指导原则功能$\mathcal{T}$。例如，流行的反向Kullback-Leibler 散度对应$f(u) = -\log(u)$，导致$T^*(x) = -q(x)/p(x)$，参见表5。

Name	$D_f(P\|Q)$	Generator $f(u)$	$T^*(x)$
Kullback-Leibler	$\int p(x) \log \frac{p(x)}{q(x)} \mathrm{d}x$	$u \log u$	$1 + \log \frac{p(x)}{q(x)}$
Reverse KL	$\int q(x) \log \frac{q(x)}{p(x)} \mathrm{d}x$	$-\log u$	$-\frac{q(x)}{p(x)}$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)} \mathrm{d}x$	$(u-1)^2$	$2(\frac{p(x)}{q(x)} - 1)$
Squared Hellinger	$\int \left(\sqrt{p(x)} - \sqrt{q(x)}\right)^2 \mathrm{d}x$	$(\sqrt{u}-1)^2$	$(\sqrt{\frac{p(x)}{q(x)}} - 1) \cdot \sqrt{\frac{q(x)}{p(x)}}$
Jensen-Shannon	$\frac{1}{2} \int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} \mathrm{d}x$	$-(u+1) \log \frac{1+u}{2} + u \log u$	$\log \frac{2p(x)}{p(x)+q(x)}$
GAN	$\int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} \mathrm{d}x - \log(4)$	$u \log u - (u+1) \log(u+1)$	$\log \frac{p(x)}{p(x)+q(x)}$

Table 1: f -发散度列表$D_f(P\|Q)$连同生成器函数。发散列表的一部分及其产生器是基于在[26]上。对于所有的散度我们有$f : \mathrm{dom}_f \to \mathbb{R} \cup \{+\infty\}$，其中$f$是凸的并且是下半连续的。我们还有$f(1) = 0$确保$D_f(P\|P) = 0$对于任何发行版$P$。如[10]所示GAN与Jensen-Shannon散度有关$D_{\mathrm{GAN}} = 2D_{\mathrm{JS}} - \log(4)$。

我们在表5中列出了常见的f -差异并提供它们的Fenchel共轭物f^*和结构域表6中的dom_{f^*}。给出了生成函数及其共轭函数的图补充资料。

2.3 变分发散最小化(VDM)

我们现在使用变分下界(4) f -divergence $D_f(P\|Q)$以估计生成模型Q给出一个真正的分布P。

为此，我们遵循生成对抗法方法[10]和使用两个神经网络，Q和T。Q 我们的生成模型，输入是一个随机向量，输出是a吗感兴趣的样本。我们通过一个向量θ参数化Q，然后写Q_θ。T 我们的变分函数，以样本作为输入并返回a吗标量。我们使用一个向量ω来参数化T，并写出T_ω。

我们可以通过找到的鞍点来学习生成模型Q_θ 以下f -GAN目标函数，其中我们相对于最小化θ和ω的最大值，

$$F(\theta, \omega) = \mathbb{E}_{x \sim P} \left[T_\omega(x)\right] - \mathbb{E}_{x \sim Q_\theta} \left[f^*(T_\omega(x))\right]. \tag{6}$$

为了在给定的有限训练数据集上优化(6)，我们近似使用小批量样本的期望。为了近似$\mathbb{E}_{x \sim P}[\cdot]$，我们对$B$实例进行采样，但没有从训练集替换。为了近似$\mathbb{E}_{x \sim Q_\theta}[\cdot]$，我们采样$B$ 来自当前生成模型Q_θ的实例。

2.4 变分函数的表示

应用变分目标(6)为不同f -散度，我们需要考虑共轭的定义域dom_{f^*} 功能f^*。为此，我们假设变分函数T_ω表示在$T_\omega(x) = g_f(V_\omega(x))$的表单中重写鞍目标(6)如下:

$$F(\theta, \omega) = \mathbb{E}_{x \sim P} \left[g_f(V_\omega(x))\right] + \mathbb{E}_{x \sim Q_\theta} \left[-f^*(g_f(V_\omega(x)))\right], \tag{7}$$

哪里$V_\omega : \mathcal{X} \to \mathbb{R}$没有任何范围限制在输出上，$g_f : \mathbb{R} \to \mathrm{dom}_{f^*}$是一个输出激活函数特定于$f$ -发散使用。在表6中，我们提出了合适的输出激活各种共轭函数f^*和它们的函数域。[1]虽然选择g_f有点任意的，我们选择它们都是单调递增函数一个大的输出$V_\omega(x)$对应的信念变分函数表示样本x来源于数据分布P与GAN的情况一样;参见图1。看看第二个任期$-f^*(g_f(v))$也很有启发意义鞍座目标(7)。这个术语通常是(除了) Pearson (χ^2散度)呈递减趋势输出函数$V_\omega(x)$倾向于变分函数从生成器输出的样本是负数。

我们可以看到GAN的目标，

$$F(\theta, \omega) = \mathbb{E}_{x \sim P} \left[\log D_\omega(x)\right] + \mathbb{E}_{x \sim Q_\theta} \left[\log(1 - D_\omega(x))\right], \tag{8}$$

作为(7)的特殊实例通过标识每一项的期望值分别为(7)和(8)。特别地，在鉴别器中选择最后一个非线性如乙状结肠$D_\omega(x) = 1/(1 + e^{-V_\omega(x)})$，对应输出激活函数为$g_f(v) = -\log(1 + e^{-v})$;见表6。

Name	Output activation g_f	dom_{f^*}	Conjugate $f^*(t)$	$f'(1)$
Kullback-Leibler (KL)	v	$\mathbb{R}$	$\exp(t-1)$	1
Reverse KL	$-\exp(-v)$	$\mathbb{R}_-$	$-1-\log(-t)$	-1
Pearson χ^2	v	$\mathbb{R}$	$\frac{1}{4}t^2+t$	0
Squared Hellinger	$1-\exp(-v)$	$t<1$	$\frac{t}{1-t}$	0
Jensen-Shannon	$\log(2)-\log(1+\exp(-v))$	$t<\log(2)$	$-\log(2-\exp(t))$	0
GAN	$-\log(1+\exp(-v))$	$\mathbb{R}_-$	$-\log(1-\exp(t))$	$-\log(2)$

Table 2: 推荐最终层激活函数和临界变分功能级别由$f'(1)$定义。临界值$f'(1)$可以解释为分类阈值应用于$T(x)$以区分真实样本和生成样本。

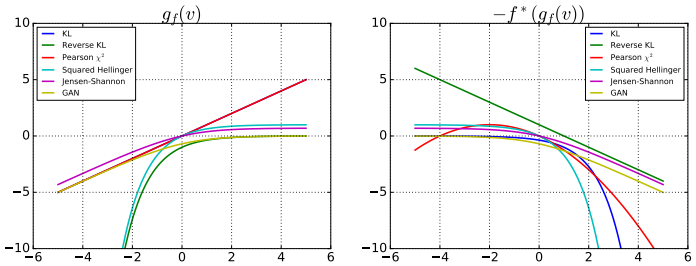

Figure 1: 鞍形目标中的两项(7) 绘制为变分函数$V_\omega(x)$的函数。

2.5 示例:高斯函数的单变量混合

为了证明不同的f -散度和为了验证变分发散估计框架，我们执行了一个实验类似于[24]。

设置。 我们通过学习高斯分布来近似高斯分布的混合物。我们用线性表示我们的模型Q_θ 函数，它接收一个随机的$z\sim\mathcal{N}(0,1)$并输出$G_\theta(z)=\mu+\sigma z$，其中$\theta=(\mu,\sigma)$是要学习的两个标量参数。对于变分函数T_ω，我们使用两个神经网络每个隐藏层都有64单位和tanh激活。我们使用单步优化目标$F(\omega,\theta)$ 梯度方法见3节。在每个步骤中，我们分别为$p(x)$和$p(z)$采样大小为1024的批次我们使用$\eta=0.01$步长来更新ω和θ。我们将结果与精确优化提供的最佳拟合进行比较$D_f(P\|Q_\theta)$ W.R.T. θ，在这种情况下是可行的在(2)中求积分。我们使用$(\hat{\omega},\hat{\theta})$(学习)和$\theta^*$(最佳拟合)来区分其中使用的参数集两种方法。

结果。 Table的左侧3 给出了最优散度和客观价值$D_f(P||Q_{\theta^*})$ 和$F(\hat{\omega},\hat{\theta})$ 以及由此产生的手段和标准偏差。注意，结果符合下界性质，即我们有$D_f(P||Q_{\theta^*})\geq F(\hat{\omega},\hat{\theta})$. 在目标的差距和差异之间有很好的对应关系在拟合均值和标准差之间。桌子的右边3 显示以下结果实验:(1)我们训练T_ω 和Q_θ 使用特定的散度，然后(2)我们估计散度并重新训练T_ω 同时保持Q_θ 固定。不出所料，Q_θ 在训练时的散度上表现最好。进一步的细节显示了拟合高斯分布和最优分布的详细图变分函数在补充材料中给出。

综上所述，以上结果表明，当生成模型为不正确，不包含真正的分布，散度用于估计的函数对学习的模型有很大的影响。

3 变分发散最小化(VDM)算法

我们现在讨论求鞍点的数值方法目标(6)。为此，我们区分了两种方法;第一，交替法最初由古德费罗等人提出Al. [10]，第二，更直接的单步优化程序。

在我们的变分框架中，交替梯度法可以描述为双环法; 内部的环收紧了底部而外环则改进了生成器模型。虽然这种方法的动机似乎是合理的，但在实践中选择在内部循环中走一步很受欢迎。再见。Al. [10]提供局部收敛我保证。

[1]请注意，对于数值实现，我们建议直接健壮地实现标量函数$f^*(g_f(\cdot))$，而不是按顺序对两个函数求值;见图1。

	KL	KL-rev	JS	Jeffrey	Pearson
$D_f(P\|\|Q_{\theta^*})$	0.2831	0.2480	0.1280	0.5705	0.6457
$F(\hat{\omega}, \hat{\theta})$	0.2801	0.2415	0.1226	0.5151	0.6379
μ^*	1.0100	1.5782	1.3070	1.3218	0.5737
$\hat{\mu}$	1.0335	1.5624	1.2854	1.2295	0.6157
σ^*	1.8308	1.6319	1.7542	1.7034	1.9274
$\hat{\sigma}$	1.8236	1.6403	1.7659	1.8087	1.9031

train \ test	KL	KL-rev	JS	Jeffrey	Pearson
KL	**0.2808**	0.3423	0.1314	0.5447	0.7345
KL-rev	0.3518	**0.2414**	0.1228	0.5794	1.3974
JS	0.2871	0.2760	**0.1210**	0.5260	0.92160
Jeffrey	0.2869	0.2975	0.1247	**0.5236**	0.8849
Pearson	0.2970	0.5466	0.1665	0.7085	**0.648**

Table 3: 混合高斯函数的高斯近似。左:最优目标，学习均值和标准差: $\hat{\theta} = (\hat{\mu}, \hat{\sigma})$(学识渊博)和$\theta^* = (\mu^*, \sigma^*)$(最适合)。右:每个训练模型的真实分布的客观值。对于每一次散度，模型的目标函数值最小训练这种发散。

3.1 单步梯度法

受单内步交替梯度法的成功启发，我们提出一个更简单的算法，见算法1。该算法与原算法有所不同一是没有内循环，关于ω的梯度θ是在一次反向传播中计算的。

Algorithm 1 单步梯度法

function SINGLESTEPGRADIENTITERATION($P, \theta^t, \omega^t, B, \eta$)
  Sample $X_P = \{x_1, \ldots, x_B\}$ and $X_Q = \{x'_1, \ldots, x'_B\}$, from P and Q_{θ^t}, respectively.
  Update: $\omega^{t+1} = \omega^t + \eta \nabla_\omega F(\theta^t, \omega^t)$.
  Update: $\theta^{t+1} = \theta^t - \eta \nabla_\theta F(\theta^t, \omega^t)$.
end function

分析。 这里我们以几何方式展示算法1 收敛到一个鞍点(θ^*, ω^*)如果有鞍点周围的邻域，其中F是强凸的θ和强烈凹在ω。这些条件类似于假设在[10]和可以正式形式如下:

$$\nabla_\theta F(\theta^*, \omega^*) = 0, \quad \nabla_\omega F(\theta^*, \omega^*) = 0, \quad \nabla^2_\theta F(\theta, \omega) \succeq \delta I, \quad \nabla^2_\omega F(\theta, \omega) \preceq -\delta I. \tag{9}$$

这些假设是必要的，除了"强"部分定义鞍点的类型变分框架的有效解。注意，虽然可能有很多鞍点它们来自于深度网络的结构[6]，他们没有资格作为我们的解决方案这些假设下的变分框架。

为方便起见，我们定义$\pi^t = (\theta^t, \omega^t)$。现在，算法1的收敛性可以表示为以下(证明见补充材料):

Theorem 1. *Suppose that there is a saddle point $\pi^* = (\theta^*, \omega^*)$ with a neighborhood that satisfies conditions* (9)*. Moreover, we define $J(\pi) = \frac{1}{2}\|\nabla F(\pi)\|_2^2$ and assume that in the above neighborhood, F is sufficiently smooth so that there is a constant $L > 0$ such that $\|\nabla J(\pi') - \nabla J(\pi)\|_2 \leq L\|\pi' - \pi\|_2$ for any π, π' in the neighborhood of π^*. Then using the step-size $\eta = \delta/L$ in Algorithm 1, we have*

$$J(\pi^t) \leq \left(1 - \frac{\delta^2}{2L}\right)^t J(\pi^0)$$

That is, the squared norm of the gradient $\nabla F(\pi)$ decreases geometrically.

3.2 实际考虑

在这里，我们讨论启发式的原则扩展[10]和真实/虚假的统计数据Larsen和S ø nderby [2]。此外我们讨论一些实用的建议稍微偏离原则性的观点。

Goodfellow等[10]注意到了训练GAN可以通过最大化$\mathbb{E}_{x\sim Q_\theta}[\log D_\omega(x)]$而不是最小化$\mathbb{E}_{x\sim Q_\theta}[\log(1 - D_\omega(x))]$以更新发电机。在更一般的$f$ -GAN算法(1)中，这意味着我们用更新替换4行

$$\theta^{t+1} = \theta^t + \eta \nabla_\theta \mathbb{E}_{x\sim Q_{\theta^t}}[g_f(V_{\omega^t}(x))], \tag{10}$$

从而最大化发电机输出。这不仅在直观上是正确的，而且我们还可以证明平稳点通过使用与在[10];我们发现这对其他分歧。

[2] http://torch.ch/blog/2015/11/13/gan.html

Larsen和S ø nderby建议监控真实和虚假的统计数据，哪个被定义为?的真正和真负速率变分函数，将其视为二元分类器。因为我们的输出激活g_f都是单调的，我们可以推导出类似的统计任何f -发散仅通过改变决策阈值。由于密度比和变分函数之间的联系(5)，阈值为$f'(1)$(见表) 6)。也就是说，我们可以解释的输出变分函数将输入x分类为true 抽样如果是变分函数$T_\omega(x)$比$f'(1)$大，将其归类为否则，从生成器中采样。

我们发现亚当[17]和渐变剪辑是有用的特别是在LSUN数据集的大规模实验中。

4 实验

我们现在训练基于VDM的生成神经采样器在MNIST和LSUN数据集。

MNIST数字。 我们使用MNIST训练数据集(60,000个样本，28 × 28像素图像)来提出了训练生成器和变分函数模型在[10]中表示各种f -散度。以$z \sim \text{Uniform}_{100}(-1, 1)$作为输入，生成器模型有两个线性层，每个层后面是批规范化和ReLU 激活和最后的线性层接着是s型函数。变分函数$V_\omega(x)$有三个线性层指数线性单位[4]在两者之间。决赛激活是特定于每个发散的，列在表6中。在[27]中，我们使用Adam的学习率为$\alpha = 0.0002$和更新权重$\beta = 0.5$。我们使用的批量为4096，从未替换的训练集中抽样，并训练每个模型为一个小时。我们还比较了变分自编码器[18] 有20个潜在维度。

结果和讨论。我们使用核密度估计(Parzen)来评估性能窗口)方法用于[10]。为此，我们从模型中采样16k图像并估计一个Parzen窗口估计使用各向同性高斯核带宽使用三倍交叉验证。最终密度模型用于计算平均值MNIST测试集(10k个样本)上的对数似然。我们在表4中显示了结果，并从我们的图2中的模型。

已知使用KDE方法进行对数似然估计缺陷[31]。特别是，对于MNIST ($d = 784$)中使用的维数获得准确对数似然估计所需的模型样本为大得不可思议。我们发现多次重复之间存在很大的差异(高达50个纳特)。因此，结果并不完全是决定性的。我们还在MNIST训练集上训练了相同的KDE估计器，获得了a 明显更高的抵抗可能性。然而，令人欣慰的是，该模型是为相比之下，Kullback-Leibler散度确实实现了较高的抵抗可能性到GAN模型。

Training divergence	KDE $\langle LL \rangle$ (nats)	± SEM
Kullback-Leibler	416	5.62
Reverse Kullback-Leibler	319	8.36
Pearson χ^2	429	5.53
Neyman χ^2	300	8.33
Squared Hellinger	-708	18.1
Jeffrey	-2101	29.9
Jensen-Shannon	367	8.19
GAN	305	8.97
Variational Autoencoder [18]	445	5.36
KDE MNIST train (60k)	502	5.99

Table 4: Kernel Density Estimation evaluation on the MNIST test data set. Each KDE model is build from 16,384 samples from the learned generative model. We report the mean log-likelihood on the MNIST test set ($n = 10,000$) and the standard error of the mean. The KDE MNIST result is using 60,000 MNIST training images to fit a single KDE model.

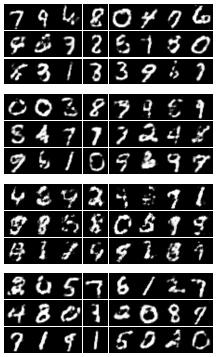

Figure 2: MNIST model samples trained using KL, reverse KL, Hellinger, Jensen from top to bottom.

LSUN自然图像。 通过DCGAN的工作[27]生成对抗该方法在生成自然外观的图像方面显示出了真正的希望。这里我们使用与[27]和replace相同的架构GAN目标和我们更一般的f -GAN目标。

我们使用大规模的LSUN数据库[34]的自然图像不同的类别。为了说明不同发散的不同行为，我们训练同一模型上的教室类图片，包含168,103张教室环境的图像，重新缩放和中心裁剪为96 × 96 像素。

设置。我们使用中提出的生成器架构和训练设置DCGAN[27]。模型接收$z \in \text{Uniform}_{d_{\text{rand}}}(-1, 1)$ 并通过一个线性层和三个反卷积层进行输入中间有批处理规范化和ReLU激活。变分函数与鉴别器结构相同在[27]和遵循卷积神经的结构网络具有批归一化，指数线性单位[4]和最后一个线性层。

结果。图3显示了来自神经网络的16个随机样本使用GAN, KL和平方海灵格散度训练的采样器。这三种差异产生的样本同样真实。请注意学习分布Q_θ的差异只有在发电机模型不够丰富。

(a) GAN (b) KL (c) Squared Hellinger

Figure 3: 来自三个不同发散点的样本。

5 相关工作

现在我们讨论我们的方法如何与现有的工作联系起来。构建真实世界分布的生成模型是一个基本目标机器学习和很多相关的工作都存在。我们只讨论适用于神经网络模型的工作。

混合密度网络[2]是直接回归有限参数参数的神经网络混合模型。当与递归神经网络结合使用时，结果是令人印象深刻的手写文本生成模型[11]。

NADE[19]和*RNADE*[33]执行a 使用预定义的任意排序对输出进行因式分解的输出维数。得到的模型每次对一个变量进行采样在过去变量的整个历史上。这些模型提供了易于处理的可能性评估和令人信服的结果但是在许多应用中，如何选择分解顺序尚不清楚。

扩散概率模型[29] 定义一个目标分布作为一个学习扩散过程的结果从一个平凡的已知分布开始。学习模型提供了精确的样本和近似对数似然评估。

噪声对比估计(NCE) [13]就是一种方法该方法通过执行从非线性逻辑回归中区分数据人为制造的噪音。NCE可以看作GAN的一个特例，其中鉴别器被限制为依赖于模型的特定形式(逻辑回归) 分类器)和发电机(保持固定)是人工提供的产生的噪声(见补充资料)。

生成神经采样器模型的[22]和[3]则没有提供满意的学习方法; [22]采用重要性抽样和[3]期望最大化。GAN和我们工作的主要区别在于学习目标，这是有效的和计算廉价。

变分自编码器(VAE) [18, 28]是将样本映射到a的概率编码器和解码器模型对潜表示和返回，使用变分贝叶斯学习进行训练目标。VAEs的优势在于编码器模型允许从观察到潜在表征的有效推理是f - gan的一个令人信服的替代品，最近的工作研究了两种方法的结合[23]

作为GAN训练目标的替代方案，工作[20]和独立考虑了[7]内核的使用最大平均差(MMD) [12, 9]作为培训目标为概率模型。与GAN模型相比，这个目标更容易训练，因为有没有显式表示的变分函数。但是，它需要选择一个内核功能报告的结果是远比GAN略逊一筹。MMD是一个更大概率类的特殊实例指标[30]都采用这种形式$D(P,Q) = \sup_{T\in\mathcal{T}} |\mathbb{E}_{x\sim P}[T(x)] - \mathbb{E}_{x\sim Q}[T(x)]|$，其中选择函数类$\mathcal{T}$ 以一种特定于散度的方式。除了MMD其他流行的指标这种形式是总变差度量(也是f -散度) Wasserstein距离和Kolmogorov距离。

在[16]中GAN的概括目标是通过使用另一种简森-香农方法提出的类似于*KL*和反向之间的插值的散度 KL散度，以Jensen-Shannon为中点。可以看出，当π接近0和1时，它会导致一种行为类似于由KL和反向KL分歧产生的目标(见补充资料)。

6 讨论

生成神经采样器提供了一种强大的方法来表示复杂没有限制因子分解假设的分布。然而，当纯生成神经采样器被用于此纸是有趣的，它们的使用是有限的，因为经过训练后它们不能以观察到的数据为条件，因此不能提供推论。

我们相信，在未来真正的好处神经采样为在判别模型和我们的通过提供额外的输入，所提出的方法很容易扩展到这种情况和条件GAN中的变分函数一样模型[8]。

致谢我们感谢费伦茨Huszár就生成对抗方法。

References

[1] S. M. Ali and S. D. Silvey. A general class of coefficients of divergence of one distribution from another. *JRSS (B)*, pages 131–142, 1966.

[2] C. M. Bishop. Mixture density networks. Technical report, Aston University, 1994.

[3] C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM: The generative topographic mapping. *Neural Computation*, 10(1):215–234, 1998.

[4] D. A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv:1511.07289*, 2015.

[5] I. Csiszár and P. C. Shields. Information theory and statistics: A tutorial. *Foundations and Trends in Communications and Information Theory*, 1:417–528, 2004.

[6] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *NIPS*, pages 2933–2941, 2014.

[7] G. K. Dziugaite, D. M. Roy, and Z. Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. In *UAI*, pages 258–267, 2015.

[8] J. Gauthier. Conditional generative adversarial nets for convolutional face generation. Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester 2014, 2014.

[9] T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. *JASA*, 102(477): 359–378, 2007.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[11] A. Graves. Generating sequences with recurrent neural networks. *arXiv:1308.0850*, 2013.

[12] A. Gretton, K. Fukumizu, C. H. Teo, L. Song, B. Schölkopf, and A. J. Smola. A kernel statistical test of independence. In *NIPS*, pages 585–592, 2007.

[13] M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, pages 297–304, 2010.

[14] J. B. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of convex analysis*. Springer, 2012.

[15] F. Huszár. An alternative update rule for generative adversarial networks. `http://www.inference.vc/an-alternative-update-rule-for-generative-adversarial-networks/`.

[16] F. Huszár. How (not) to train your generative model: scheduled sampling, likelihood, adversary? *arXiv:1511.05101*, 2015.

[17] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[18] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *arXiv:1402.0030*, 2013.

[19] H. Larochelle and I. Murray. The neural autoregressive distribution estimator. In *AISTATS*, 2011.

[20] Y. Li, K. Swersky, and R. Zemel. Generative moment matching networks. In *ICML*, 2015.

[21] F. Liese and I. Vajda. On divergences and informations in statistics and information theory. *Information Theory, IEEE*, 52(10):4394–4412, 2006.

[22] D. J. C. MacKay. Bayesian neural networks and density networks. *Nucl. Instrum. Meth. A*, 354(1):73–80, 1995.

[23] A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow. Adversarial autoencoders. *arXiv:1511.05644*, 2015.

[24] T. Minka. Divergence measures and message passing. Technical report, Microsoft Research, 2005.

[25] X. Nguyen, M. J. Wainwright, and M. I. Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *Information Theory, IEEE*, 56(11):5847–5861, 2010.

[26] F. Nielsen and R. Nock. On the chi-square and higher-order chi distances for approximating f-divergences. *Signal Processing Letters, IEEE*, 21(1):10–13, 2014.

[27] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv:1511.06434*, 2015.

[28] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, pages 1278–1286, 2014.

[29] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using non-equilibrium thermodynamics. *ICML*, pages 2256—-2265, 2015.

[30] B. K. Sriperumbudur, A. Gretton, K. Fukumizu, B. Schölkopf, and G. Lanckriet. Hilbert space embeddings and metrics on probability measures. *JMLR*, 11:1517–1561, 2010.

[31] L. Theis, A. v.d. Oord, and M. Bethge. A note on the evaluation of generative models. *arXiv:1511.01844*, 2015.

[32] S. Tokui, K. Oono, S. Hido, and J. Clayton. Chainer: a next-generation open source framework for deep learning. In *NIPS*, 2015.

[33] B. Uria, I. Murray, and H. Larochelle. RNADE: The real-valued neural autoregressive density-estimator. In *NIPS*, pages 2175–2183, 2013.

[34] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv:1506.03365*, 2015.

Supplementary Materials

A Introduction

We provide additional material to support the content presented in the paper. The text is structured as follows. In Section B we present an extended list of f-divergences, corresponding generator functions and their convex conjugates. In Section C we provide the proof of Theorem 2 from Section 3. In Section D we discuss the differences between current (to our knowledge) GAN optimisation algorithms. Section E provides a proof of concept of our approach by fitting a Gaussian to a mixture of Gaussians using various divergence measures. Finally, in Section F we present the details of the network architectures used in Section 4 of the main text.

B f-divergences and Generator-Conjugate Pairs

In Table 5 we show an extended list of f-divergences $D_f(P\|Q)$ together with their generators $f(u)$ and the corresponding optimal variational functions $T^*(x)$. For all divergences we have $f : \mathrm{dom}_f \to \mathbb{R} \cup \{+\infty\}$, where f is convex and lower-semicontinuous. Also we have $f(1) = 0$ which ensures that $D_f(P\|P) = 0$ for any distribution P. As shown by [10] GAN is related to the Jensen-Shannon divergence through $D_{\mathrm{GAN}} = 2D_{\mathrm{JS}} - \log(4)$. The GAN generator function f does not satisfy $f(1) = 0$ hence $D_{\mathrm{GAN}}(P\|P) \neq 0$.

Table 6 lists the convex conjugate functions $f^*(t)$ of the generator functions $f(u)$ in Table 5, their domains, as well as the activation functions g_f we use in the last layers of the generator networks to obtain a correct mapping of the network outputs into the domains of the conjugate functions.

The panels of Figure 4 show the generator functions and the corresponding convex conjugate functions for a variety of f-divergences.

Name	$D_f(P\|Q)$	Generator $f(u)$	$T^*(x)$
Total variation	$\frac{1}{2}\int \lvert p(x) - q(x)\rvert \,\mathrm{d}x$	$\frac{1}{2}\lvert u-1\rvert$	$\frac{1}{2}\mathrm{sign}(\frac{p(x)}{q(x)} - 1)$
Kullback-Leibler	$\int p(x) \log \frac{p(x)}{q(x)} \,\mathrm{d}x$	$u \log u$	$1 + \log \frac{p(x)}{q(x)}$
Reverse Kullback-Leibler	$\int q(x) \log \frac{q(x)}{p(x)} \,\mathrm{d}x$	$-\log u$	$-\frac{q(x)}{p(x)}$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)} \,\mathrm{d}x$	$(u-1)^2$	$2(\frac{p(x)}{q(x)} - 1)$
Neyman χ^2	$\int \frac{(p(x)-q(x))^2}{q(x)} \,\mathrm{d}x$	$\frac{(1-u)^2}{u}$	$1 - \left[\frac{q(x)}{p(x)}\right]^2$
Squared Hellinger	$\int \left(\sqrt{p(x)} - \sqrt{q(x)}\right)^2 \,\mathrm{d}x$	$(\sqrt{u}-1)^2$	$(\sqrt{\frac{p(x)}{q(x)}} - 1) \cdot \sqrt{\frac{q(x)}{p(x)}}$
Jeffrey	$\int (p(x) - q(x)) \log\left(\frac{p(x)}{q(x)}\right) \,\mathrm{d}x$	$(u-1)\log u$	$1 + \log \frac{p(x)}{q(x)} - \frac{q(x)}{p(x)}$
Jensen-Shannon	$\frac{1}{2}\int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} \,\mathrm{d}x$	$-(u+1)\log \frac{1+u}{2} + u \log u$	$\log \frac{2p(x)}{p(x)+q(x)}$
Jensen-Shannon-weighted	$\int p(x)\pi \log \frac{p(x)}{\pi p(x)+(1-\pi)q(x)} + (1-\pi)q(x) \log \frac{q(x)}{\pi p(x)+(1-\pi)q(x)} \,\mathrm{d}x$	$\pi u \log u - (1-\pi+\pi u)\log(1-\pi+\pi u)$	$\pi \log \frac{p(x)}{(1-\pi)q(x)+\pi p(x)}$
GAN	$\int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} \,\mathrm{d}x - \log(4)$	$u \log u - (u+1)\log(u+1)$	$\log \frac{p(x)}{p(x)+q(x)}$
α-divergence ($\alpha \notin \{0,1\}$)	$\frac{1}{\alpha(\alpha-1)} \int \left(p(x)\left[\left(\frac{q(x)}{p(x)}\right)^\alpha - 1\right] - \alpha(q(x)-p(x))\right) \,\mathrm{d}x$	$\frac{1}{\alpha(\alpha-1)}\left(u^\alpha - 1 - \alpha(u-1)\right)$	$\frac{1}{\alpha-1}\left[\left[\frac{p(x)}{q(x)}\right]^{\alpha-1} - 1\right]$

Table 5: f -发散度列表$D_f(P\|Q)$连同生成器函数和最优变分函数。

C Proof of Theorem 1

In this section we present the proof of Theorem 2 from Section 3 of the main text. For completeness, we reiterate the conditions and the theorem.

We assume that F is strongly convex in θ and strongly concave in ω such that

$$\nabla_\theta F(\theta^*, \omega^*) = 0, \quad \nabla_\omega F(\theta^*, \omega^*) = 0, \tag{11}$$

$$\nabla^2_\theta F(\theta, \omega) \succeq \delta I, \quad \nabla^2_\omega F(\theta, \omega) \preceq -\delta I. \tag{12}$$

These assumptions are necessary except for the "strong" part in order to define the type of saddle points that are valid solutions of our variational framework.

Name	Output activation g_f	dom_{f^*}	Conjugate $f^*(t)$	$f'(1)$
Total variation	$\frac{1}{2}\tanh(v)$	$-\frac{1}{2} \leq t \leq \frac{1}{2}$	t	0
Kullback-Leibler (KL)	v	$\mathbb{R}$	$\exp(t-1)$	1
Reverse KL	$-\exp(v)$	$\mathbb{R}_-$	$-1-\log(-t)$	-1
Pearson χ^2	v	$\mathbb{R}$	$\frac{1}{4}t^2+t$	0
Neyman χ^2	$1-\exp(v)$	$t<1$	$2-2\sqrt{1-t}$	0
Squared Hellinger	$1-\exp(v)$	$t<1$	$\frac{t}{1-t}$	0
Jeffrey	v	$\mathbb{R}$	$W(e^{1-t})+\frac{1}{W(e^{1-t})}+t-2$	0
Jensen-Shannon	$\log(2)-\log(1+\exp(-v))$	$t<\log(2)$	$-\log(2-\exp(t))$	0
Jensen-Shannon-weighted	$-\pi\log\pi-\log(1+\exp(-v))$	$t<-\pi\log\pi$	$(1-\pi)\log\frac{1-\pi}{1-\pi e^{t/\pi}}$	0
GAN	$-\log(1+\exp(-v))$	$\mathbb{R}_-$	$-\log(1-\exp(t))$	$-\log(2)$
α-div. ($\alpha<1, \alpha\neq 0$)	$\frac{1}{1-\alpha}-\log(1+\exp(-v))$	$t<\frac{1}{1-\alpha}$	$\frac{1}{\alpha}(t(\alpha-1)+1)^{\frac{\alpha}{\alpha-1}}-\frac{1}{\alpha}$	0
α-div. ($\alpha>1$)	v	$\mathbb{R}$	$\frac{1}{\alpha}(t(\alpha-1)+1)^{\frac{\alpha}{\alpha-1}}-\frac{1}{\alpha}$	0

Table 6: 推荐最终层激活函数和临界变分功能级别由$f'(1)$定义。训练生成神经采样器的目标函数Q_θ 给定真分布P和辅助变分函数T是$\min_\theta \max_\omega F(\theta,\omega) = \mathbb{E}_{x\sim P}[T_\omega(x)] - \mathbb{E}_{x\sim Q_\theta}[f^*(T_\omega(x))]$。对于任何样例$x$，变分函数产生标量$v(x) \in \mathbb{R}$。输出激活提供一个可微映射$g_f : \mathbb{R} \to \text{dom}_{f^*}$，定义$T(x) = g_f(v(x))$。临界值$f'(1)$可以解释为分类阈值应用于$T(x)$以区分真实样本和生成样本。$W$ 为兰伯特- W积对数函数。

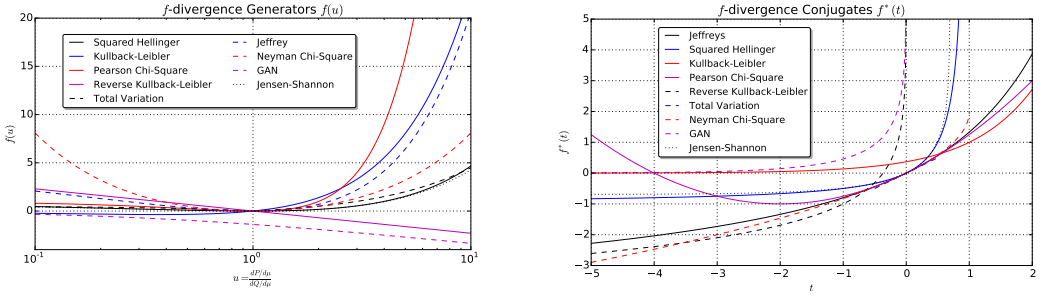

Figure 4: 变分框架下的发生器共轭(f, f^*)对Nguyen等人[25]。左:f -divergence中使用的生成器函数f $D_f(P\|Q) = \int_{\mathcal{X}} q(x) f\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x$。右:变分的共轭函数$f^*$ 散度下界$D_f(P\|Q) \geq \sup_{T\in\mathcal{T}} \int_{\mathcal{X}} p(x)\,T(x) - q(x) f^*(T(x))\,\mathrm{d}x$。

We define $\pi^t = (\theta^t, \omega^t)$ and use the notation

$$\nabla F(\pi) = \begin{pmatrix} \nabla_\theta F(\theta,\omega) \\ \nabla_\omega F(\theta,\omega) \end{pmatrix}, \quad \tilde{\nabla} F(\pi) = \begin{pmatrix} -\nabla_\theta F(\theta,\omega) \\ \nabla_\omega F(\theta,\omega) \end{pmatrix}.$$

With this notation, Algorithm 1 in the main text can be written as

$$\pi^{t+1} = \pi^t + \eta \tilde{\nabla} F(\pi^t).$$

Given the above assumptions and notation, in Section 3 of the main text we formulate the following theorem.

Theorem 2. *Suppose that there is a saddle point $\pi^* = (\theta^*, \omega^*)$ with a neighborhood that satisfies conditions* (11) *and* (12)*. Moreover we define $J(\pi) = \frac{1}{2}\|\nabla F(\pi)\|_2^2$ and assume that in the above neighborhood, F is sufficiently smooth so that there is a constant $L > 0$ and*

$$J(\pi') \leq J(\pi) + \langle \nabla J(\pi), \pi' - \pi \rangle + \frac{L}{2}\|\pi' - \pi\|_2^2 \tag{13}$$

for any π, π' in the neighborhood of π^*. Then using the step-size $\eta = \delta/L$ in Algorithm 1, we have

$$J(\pi^t) \leq \left(1 - \frac{\delta^2}{2L}\right)^t J(\pi^0)$$

where L is the smoothness parameter of J. That is, the squared norm of the gradient $\nabla F(\pi)$ decreases geometrically.

Proof. First, note that the gradient of J can be written as

$$\nabla J(\pi) = \nabla^2 F(\pi) \nabla F(\pi).$$

Therefore we notice that,

$$\begin{aligned}
\left\langle \tilde{\nabla} F(\pi), \nabla J(\pi) \right\rangle &= \left\langle \tilde{\nabla} F(\pi), \nabla^2 F(\pi) \nabla F(\pi) \right\rangle \\
&= \left\langle \begin{pmatrix} -\nabla_\theta F(\theta,\omega) \\ \nabla_\omega F(\theta,\omega) \end{pmatrix}, \begin{pmatrix} \nabla^2_\theta F(\theta,\omega) & \nabla_\theta \nabla_\omega F(\theta,\omega) \\ \nabla_\omega \nabla_\theta F(\theta,\omega) & \nabla^2_\omega F(\theta,\omega) \end{pmatrix} \begin{pmatrix} \nabla_\theta F(\theta,\omega) \\ \nabla_\omega F(\theta,\omega) \end{pmatrix} \right\rangle \\
&= -\left\langle \nabla_\theta F(\theta,\omega), \nabla^2_\theta F(\theta,\omega) \nabla_\theta F(\theta,\omega) \right\rangle + \left\langle \nabla_\omega F(\theta,\omega), \nabla^2_\omega F(\theta,\omega) \nabla_\omega F(\theta,\omega) \right\rangle \\
&\leq -\delta \left(\|\nabla_\theta F(\theta,\omega)\|_2^2 + \|\nabla_\omega F(\theta,\omega)\|_2^2 \right) = -\delta \|\nabla F(\pi)\|_2^2 \qquad (14)
\end{aligned}$$

In other words, Algorithm 1 decreases J by an amount proportional to the squared norm of $\nabla F(\pi)$.

Now combining the smoothness (13) with Algorithm 1, we get

$$\begin{aligned}
J(\pi^{t+1}) &\leq J(\pi^t) + \eta \left\langle \nabla J(\pi^t), \tilde{\nabla} F(\pi^t) \right\rangle + \frac{L\eta^2}{2} \|\tilde{\nabla} F(\pi^t)\|_2^2 \\
&\leq \left(1 - \delta\eta + \frac{L\eta^2}{2}\right) J(\pi^t) \\
&= \left(1 - \frac{\delta^2}{2L}\right) J(\pi^t),
\end{aligned}$$

where we used sufficient decrease (14) and $J(\pi) = \|\nabla F(\pi)\|_2^2 = \|\tilde{\nabla} F(\pi)\|_2^2$ in the second inequality, and the final equality follows by taking $\eta = \delta/L$. □

D Related Algorithms

Due to recent interest in GAN type models, there have been attempts to derive other divergence measures and algorithms. In particular, an alternative Jensen-Shannon divergence has been derived in [16] and a heuristic algorithm that behaves similarly to the one resulting from this new divergence has been proposed in [15].

In this section we summarise (some of) the current algorithms and show how they are related. Note that some algorithms use heuristics that do not correspond to saddle point optimisation, that is, in the corresponding maximization and minimization steps they optimise alternative objectives that do not add up to a coherent joint objective. We include a short discussion of [13] because it can be viewed as a special case of GAN.

To illustrate how the discussed algorithms work, we define the objective function

$$F(\theta, \omega; \alpha, \beta) = E_{x\sim P}[\log D_\omega(x)] + \alpha E_{x\sim Q_\theta}[\log(1 - D_\omega(x))] - \beta E_{x\sim Q_\theta}[\log(D_\omega(x))], \quad (15)$$

where we introduce two scalar parameters, α and β, to help us highlight the differences between the algorithms shown in Table 7.

Algorithm	Maximisation in ω	Minimisation in θ
NCE [13]	$\alpha = 1, \beta = 0$	NA
GAN-1 [10]	$\alpha = 1, \beta = 0$	$\alpha = 1, \beta = 0$
GAN-2 [10]	$\alpha = 1, \beta = 0$	$\alpha = 0, \beta = 1$
GAN-3 [15]	$\alpha = 1, \beta = 0$	$\alpha = 1, \beta = 1$

Table 7: GAN目标的优化算法(15)。

Noise-Contrastive Estimation (NCE)

NCE [13] is a method that estimates the parameters of an unnormalised model $p(x; \omega)$ by performing non-linear logistic regression to discriminate between the model and artificially generated noise.

To achieve this NCE casts the estimation problem as a ML estimation in a binary classification model where the data is augmented with artificially generated data. The "true" data items are labeled as positives while the artificially generated data items are labeled as negatives. The discriminant function is defined as $D_\omega(x) = p(x;\omega)/(p(x;\omega) + q(x))$ where $q(x)$ denotes the distribution of the artificially generated data, typically a Gaussian parameterised by the empirical mean and covariance of the true data. ML estimation in this binary classification model results in an objective that has the form (15) with $\alpha = 1$ amd $\beta = 0$, where the expectations are taken w.r.t. the empirical distribution of augmented data. As a result, NCE can be viewed as a special case of GAN where the generator is fixed and we only have maximise the objective w.r.t. the parameters of the discriminator. Another slight difference is that in this case the data distribution is learned through the discriminator not the generator, however, the method has many conceptual similarities to GAN.

GAN-1 and GAN-2

The first algorithm (GAN-1) proposed in [10] performs a stochastic gradient ascent-descent on the objective with $\alpha = 1$ and $\beta = 0$, however, the authors point out that in practice it is more advantageous to minimise $-E_{x\sim Q_\theta}[\log D_\omega(x)]$ instead of $E_{x\sim Q_\theta}[\log(1 - D_\omega(x))]$, we denote this by GAN-2. This is motivated by the observation that in the early stages of training when Q_θ is not sufficiently well fitted, D_ω can saturate fast leading to weak gradients in $E_{x\sim Q_\theta}[\log(1 - D_\omega(x))]$. The $-E_{x\sim Q_\theta}[\log D_\omega(x)]$ term, however, can provide stronger gradients and leads to the same fixed point. This heuristic can be viewed as using $\alpha = 1, \beta = 0$ in the maximisation step and $\alpha = 0, \beta = 1$ in the minimisation step[3].

GAN-3

In [15] a further heuristic for the minimisation step is proposed. Formally, it can be viewed as a combination of the minimisation steps in GAN-1 and GAN-2. In the proposed algorithm, the maximisation step is performed similarly ($\alpha = 1, \beta = 0$), but the minimisation is done using $\alpha = 1$ and $\beta = 1$. This choice is motivated by KL optimality arguments. The author makes the observation that the optimal discriminator is given by

$$D^*(x) = \frac{p(x)}{q_\theta(x) + p(x)} \tag{16}$$

and thus, close to optimality, the minimisation of $E_{x\sim Q_\theta}[\log(1 - D_\omega(x))] - E_{x\sim Q_\theta}[\log D_\omega(x)]$ corresponds to the minimisation of the reverse KL divergence $E_{x\sim Q_\theta}[\log(q_\theta(x)/p(x))]$. This approach can be viewed as choosing $\alpha = 1$ and $\beta = 1$ in the minimisation step.

Remarks on the Weighted Jensen-Shannon Divergence in [16]

The GAN/variational objective corresponding to alternative Jensen-Shannon divergence measure proposed in [16] (see Jensen-Shannon-weighted in Table 1) is

$$F(\theta, \omega; \pi) = E_{x\sim P}[\log D_\omega(x)] - (1 - \pi)E_{x\sim Q_\theta}\Big[\log \frac{1 - \pi}{1 - \pi D_\omega(x)^{1/\pi}}\Big]. \tag{17}$$

Note that we have the $T_\omega(x) = \log D_\omega(x)$ correspondence. According to the definition of the variational objective, when T_ω is close to optimal then in the minimisation step the objective function is close to the chosen divergence. In this case the optimal discriminator is

$$D^*(x)^{1/\pi} = \frac{p(x)}{(1 - \pi)q_\theta(x) + \pi p(x)}. \tag{18}$$

The objective in (17) vanishes when $\pi \in \{0, 1\}$, however, when π is only is close to 0 and 1, it can behave similarly to the KL and reverse KL objectives, respectively. Overall, the connection between GAN-3 and the optimisation of (17) can only be considered as approximate. To obtain an exact KL or reverse KL behavior one can use the corresponding variational objectives. For a simple illustration of how these divergences behave see Section 2.5 and Section E below.

[3] A somewhat similar observation regarding the artificially generated data is made in [13]: in order to have meaningful training one should choose the artificially generated data to be close the the true data, hence the choice of an ML multivariate Gaussian.

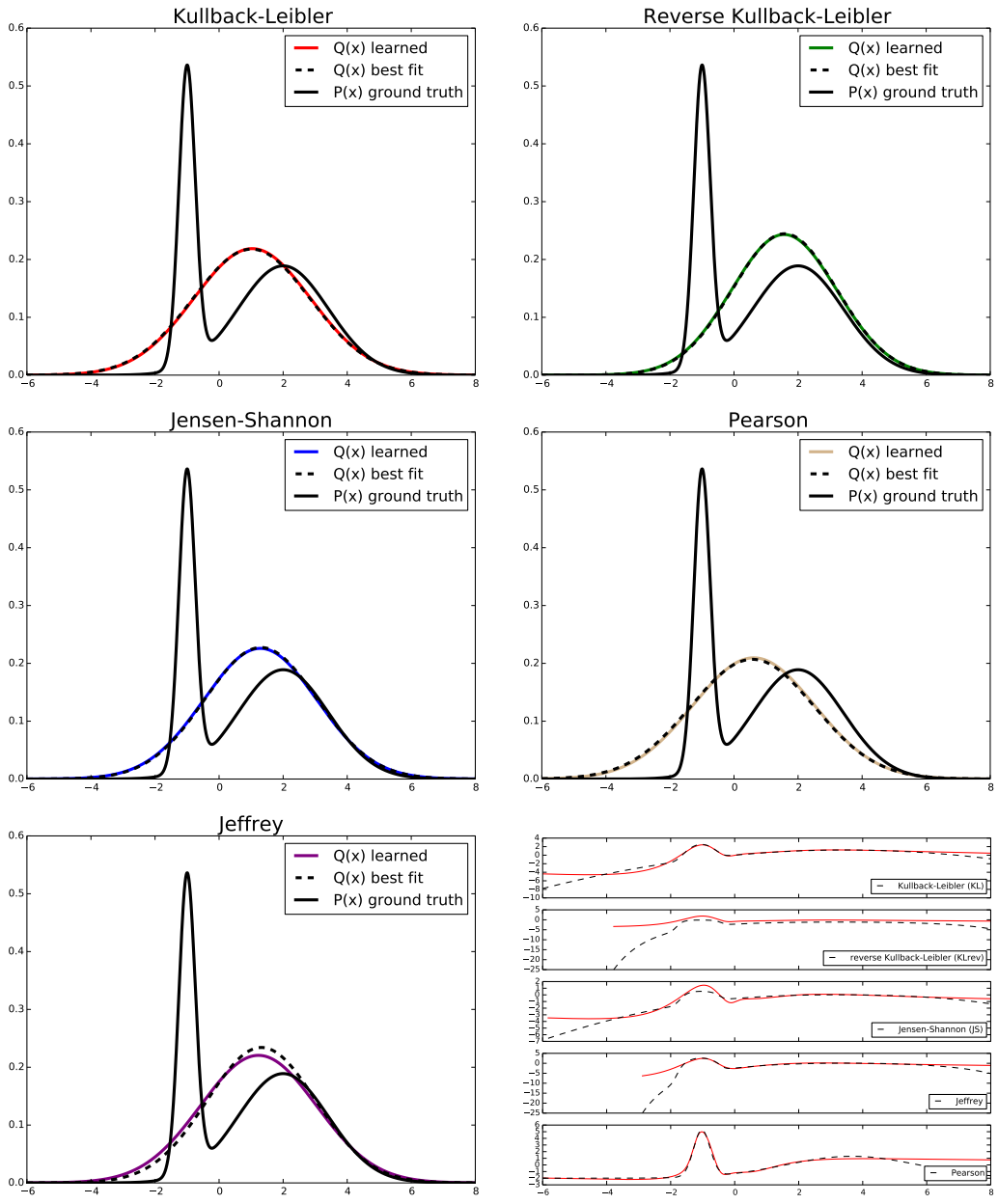

Figure 5: 混合高斯函数的高斯近似。通过直接优化$D_f(p||q_{\theta^*})$(虚线黑色)和优化$F(\hat{\omega}, \hat{\theta})$(纯色)获得的高斯近似。右下角:最优变分函数$T^*$(虚线)和$T_{\hat{\omega}}$(红色实线)。

E Details of the Univariate Example

We follow up on the example in Section 2.5 of the main text by presenting further details about the quality and behavior of the approximations resulting from using various divergence measures. For completeness, we reiterate the setup and then we present further results.

Setup. We approximate a mixture of Gaussian [4] by learning a Gaussian distribution. The model Q_θ is represented by a linear function which receives a random $z \sim \mathcal{N}(0, 1)$ and outputs

$$G_\theta(z) = \mu + \sigma z, \tag{19}$$

[4]The plots on Figure 5 correspond to $p(x) = (1-w)N(x; m_1, v_1) + wN(x; m_2, v_2)$ with $w = 0.67, m_1 = -1, v_1 = 0.0625, m_2 = 2, v_2 = 2$.

where $\theta = (\mu, \sigma)$ are the parameters to be learned. For the variational function T_ω we use the neural network

$$x \quad \rightarrow \text{Linear(1,64)} \rightarrow \text{Tanh} \rightarrow \text{Linear(64,64)} \rightarrow \text{Tanh} \rightarrow \text{Linear(64,1)}. \tag{20}$$

We optimise the objective $F(\omega, \theta)$ by using the single-step gradient method presented in Section 3.1 of the main text . In each step we sample batches of size 1024 each for both $p(x)$ and $p(z)$ and we use a step-size of 0.01 for updating both ω and θ. We compare the results to the best fit provided by the exact optimisation of $D_f(P\|Q_\theta)$ w.r.t. θ, which is feasible in this case by solving the required integrals numerically. We use $(\hat{\omega}, \hat{\theta})$ (learned) and θ^* (best fit) to distinguish the parameters sets used in these two approaches.

Results. The panels in Figure 5 shows the density function of the data distribution as well as the Gaussian approximations corresponding to a few f-divergences form Table 5. As expected, the KL approximation covers the data distribution by fitting its mean and variance while KL-rev has more of a mode-seeking behavior [24]. The fit corresponding to the Jensen-Shannon divergence is somewhere between KL and KL-rev. All Gaussian approximations resulting from neural network training are close to the ones obtained by direct optimisation of the divergence (learned vs. best fit).

In the right–bottom panel of Figure 5 we compare the variational functions $T_{\hat{\omega}}$ and T^*. The latter is defined as $T^*(x) = f'(p(x)/q_{\theta^*}(x))$, see main text. The objective value corresponding to T^* is the true divergence $D_f(P||Q_{\theta^*})$. In the majority of the cases our $T_{\hat{\omega}}$ is close to T^* in the area of interest. The discrepancies around the tails are due to the fact that (1) the class of functions resulting from the $\tanh$ activation function has limited capability representing the tails, and (2) in the Gaussian case there is a lack of data in the tails. These limitations, however, do not have a significant effect on the learned parameters.

F Details of the Experiments

In this section we present the technical setup as well as the architectures we used in the experiments described in Section 4.

F.1 Deep Learning Environment

We use the deep learning framework *Chainer* [32], version 1.8.1, running on CUDA 7.5 with CuDNN v5 on NVIDIA GTX TITAN X.

F.2 MNIST Setup

MNIST Generator

$$\begin{aligned} z \quad &\rightarrow \text{Linear(100, 1200)} \rightarrow \text{BN} \rightarrow \text{ReLU} \rightarrow \text{Linear(1200, 1200)} \rightarrow \text{BN} \rightarrow \text{ReLU} \\ &\rightarrow \text{Linear(1200, 784)} \rightarrow \text{Sigmoid} \end{aligned} \tag{21}$$

All weights are initialized at a weight scale of 0.05, as in [10].

MNIST Variational Function

$$x \quad \rightarrow \text{Linear(784,240)} \rightarrow \text{ELU} \rightarrow \text{Linear(240,240)} \rightarrow \text{ELU} \rightarrow \text{Linear(240,1)}, \tag{22}$$

where ELU is the exponential linear unit [4]. All weights are initialized at a weight scale of 0.005, one order of magnitude smaller than in [10].

Variational Autoencoders For the variational autoencoders [18], we used the example implementation included with *Chainer* [32]. We trained for 100 epochs with 20 latent dimensions.

F.3 LSUN Natural Images

$$\begin{aligned} z \quad &\rightarrow \text{Linear(100, } 6 \cdot 6 \cdot 512) \rightarrow \text{BN} \rightarrow \text{ReLU} \rightarrow \text{Reshape(512,6,6)} \\ &\rightarrow \text{Deconv(512,256)} \rightarrow \text{BN} \rightarrow \text{ReLU} \rightarrow \text{Deconv(256,128)} \rightarrow \text{BN} \rightarrow \text{ReLU} \\ &\rightarrow \text{Deconv(128,64)} \rightarrow \text{BN} \rightarrow \text{ReLU} \rightarrow \text{Deconv(64,3)}, \end{aligned} \tag{23}$$

where all Deconv operations use a kernel size of four and a stride of two.