

CSDN新首页上线啦，邀请你来立即体验！ (http://blog.csdn.net/)

立即体验

CSDN

博客 (http://blog.csdn.net/?ref=toolbar)学院 (http://edu.csdn.net/?ref=toolbar)

下载 (http://download.csdn.net/?ref=toolbar)更多 ▾

登录 (http://passport.csdn.net/account/login?ref=toolbar)注册 (http://passport.csdn.net/account/mobileregister?ref=toolbar&action=mobileRegister)

生成对抗网络的简单介绍（TensorFlow 代码）

2017年05月15日 12:15:11

标签：深度学习 (http://so.csdn.net/so/search/s.do?q=深度学习&t=blog) / GAN (http://so.csdn.net/so/search/s.do?q=GAN&t=blog) / tensorflow (http://so.csdn.net/so/search/s.do?q=tensorflow&t=blog)

2473

原文地址：

An introduction to Generative Adversarial Networks (with code in TensorFlow) (http://blog.aylien.com/introduction-generative-adversarial-networks-code-tensorflow/)

引言

最近，研究者们对生成模型的兴趣一直很大（参见OpenAI的这篇博客 (https://openai.com/blog/generative-models/)文章）。这些生成模型是可以学习创建类似于我们给它们的数据。这样的直观感受是，如果我们可以得到一个能写出高质量的新闻文章的模型，那么它一般也会学到很多关于新闻文章的内容。换句话说，这个模型也应该有一个关于新闻文章的很好的内部表示。然后，我们可以希望使用这种表示来帮助我们进行其他相关任务，例如按主题分类新闻文章。

实际上，像这样制作数据的训练模式并不容易，但是近年来，出现了一些能够运行得很好的方法。其中之一便是生成对抗网络（GAN）。著名的深度学习研究人员和Facebook AI研究主管Yann LeCun最近引用GAN作为深度学习中最重要的新发展之一 (https://www.quora.com/What-are-some-recent-and-potentially-upcoming-breakthroughs-in-deep-learning)：

“There are many interesting recent development in deep learning...The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks). This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion.” – Yann LeCun

判别模型与生成模型

在看GAN之前，让我们简单回顾一下生成模式和判别模型之间的区别：

- 判别模型学习将输入数据（x）映射到某个所需输出类标签（y）的函数。从概率出发，它们直接学习条件分布 $P(y|x)$ 。

somTian (http://blog.csdn.net/somTian)

+关注

原创31

粉丝14

喜欢0

码云0 (https://gitee.com/somTian)

他的最新文章

更多文章 (http://blog.csdn.net/somTian)

科学研究设计七：单案例设计 (http://blog.csdn.net/somTian/article/details/7566657)

科学研究设计六：有效性威胁 (http://blog.csdn.net/somTian/article/details/7566579)

科学研究设计五：实验设计 (http://blog.csdn.net/somTian/article/details/7856656)

QUALCOMM

Unable to Connect

The Proxy was unable to connect to the remote site. responding to requests. If you feel you have reached please submit a ticket via the link provided below.

URL: http://pos.baidu.com/s?hei=250&wid=300&di=URL%2Fblog.csdn.net%2FsomTian%2Farticle%2Fdetails%2F72126328

在线课程

白翼引擎在WebAssembly中的实践

白翼引擎在WebAssembly中的实践

白翼引擎在WebAssembly中的实践

白翼引擎在WebAssembly中的实践

(http://edu.csdn.net/huiyiCourse/detail/72?utm_source=blog9)

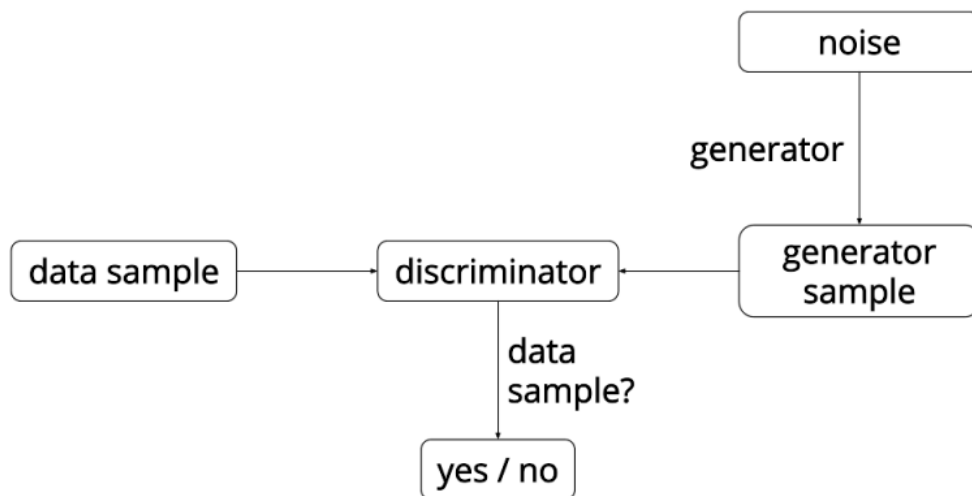
(http://edu.csdn.net/huiyiCourse/detail/602?utm_source=blog9)

- 生成模型尝试同时学习输入数据和标签的联合概率, 即 $P(x, y)$ 。这可以通过贝叶斯规则转换为 $P(y|x)$ 进行分类, 但生成能力也可以用于别的东西, 例如创建可能的新 (x, y) 样本。

两种类型的模型都是有用的, 但是生成模型比判别模型有一个有趣的优势 - 即使没有标签, 它们也有潜力理解和解释输入数据的基础结构。在现实世界中处理数据建模问题时, 这是非常可取的, 因为未标记的数据当然是丰富的, 但是获得标签数据通常是非常昂贵, 不切实际的。

生成对抗网络

GANs (<https://arxiv.org/abs/1406.2661>) 是一个有趣的想法, 由德国蒙特利尔大学的Ian Goodfellow (现OpenAI) 领导的一组研究人员于2014年首次推出。 GAN的主要思想是拥有两个竞争的神经网络模型。 一个将噪声数据作为输入, 并产生样本 (所谓的生成器)。 另一个模型 (称为判别器) 从生成器和训练数据接收样本, 并且必须能够区分两个来源。 这两个网络进行连续的博弈, 生成器学习产生越来越多的现实样本, 鉴别器正在学习越来越好地区分生成的数据和实际数据。 这两个网络同时进行训练, 最后的希望是竞争能够使生成器生成的样本与实际数据不可区分。



<http://blog.csdn.net/somTian>

GAN overview. Source: <https://ishmaelbelghazi.github.io/ALI>
(<https://ishmaelbelghazi.github.io/ALI>)

这里经常使用的类比是, 生成器就像伪造的一些物品, 而判别器就像警方试图检测伪造的物品。这种设置也可能似乎让人联想到强化学习, 其中生成器从判别器接收到奖励信号, 让它知道生成的数据是否准确。然而, 与GAN的关键区别在于, 我们可以将梯度信息从判别器反向传播回生成器网络, 因此生成器知道如何调整其参数, 以产生可以欺骗判别器的输出数据。

到目前为止, GAN主要应用于建模自然图像。他们现在在图像生成任务中产生出色的结果, 产生的图像比基于最大可能性训练目标的其他领先的生成方法训练有素的图像更加尖锐。以下是GAN生成的图像示例:

热门文章

当推荐系统遇上深度学习 (<http://blog.csdn.net/somTian/article/details/71516613>)

5490

Tensorflow实现的CNN文本分类 (<http://blog.csdn.net/somTian/article/details/69359498>)

5254

tensorboard 使用报错 (<http://blog.csdn.net/somTian/article/details/52131623>)

4101

生成对抗网络的简单介绍 (TensorFlow 代码) (<http://blog.csdn.net/somTian/article/details/72126328>)

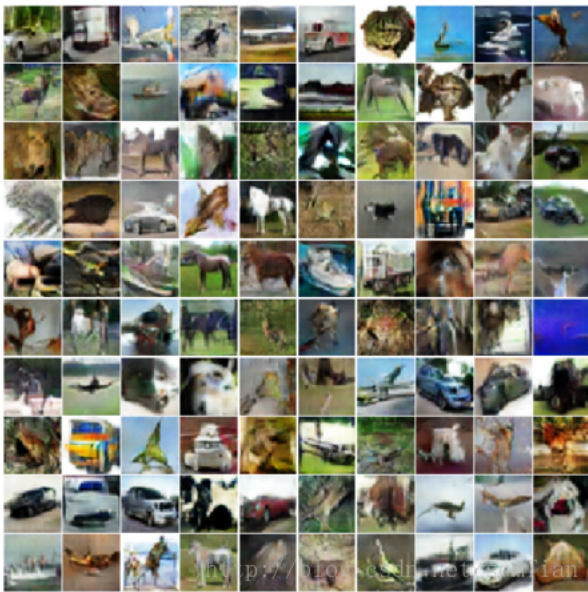
2453

centos7 同时安装python2、python3和pip3以及各种包遇到的坑 (<http://blog.csdn.net/somTian/article/details/72954705>)

2008



Generated bedrooms. Source: "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" <https://arxiv.org/abs/1511.06434v2>* (<https://arxiv.org/abs/1511.06434v2>)



Generated CIFAR-10 samples. Source: "Improved Techniques for Training GANs" <https://arxiv.org/abs/1606.03498>* (<https://arxiv.org/abs/1606.03498>)

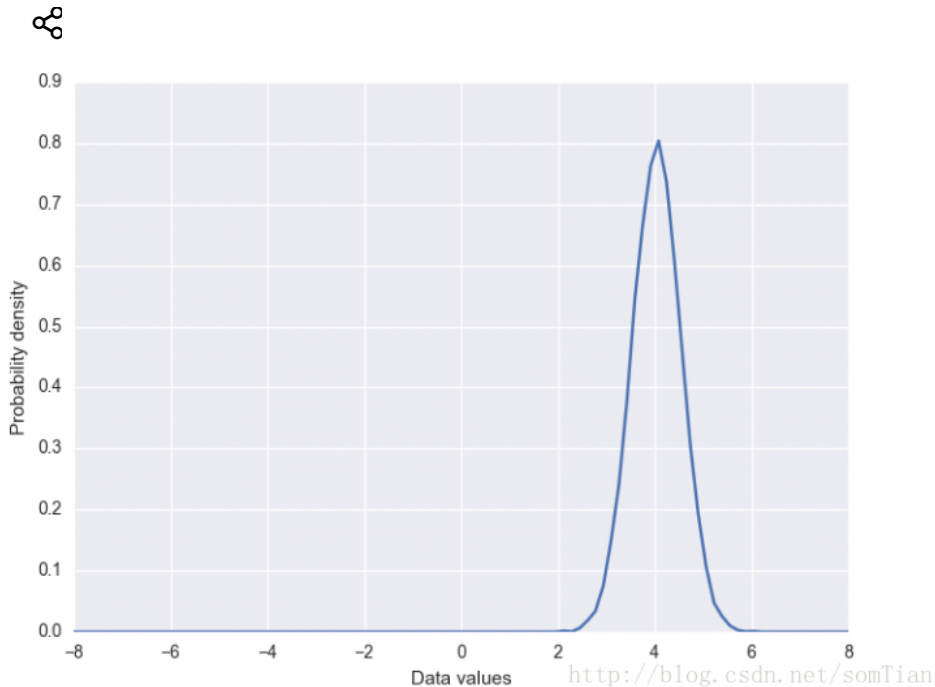
近似一维高斯分布

为了更好地了解这一切如何工作，我们将在TensorFlow中使用GAN来解决一个简单的问题 - 学习近似一维高斯分布。这是基于Eric Jang的类似目标的博文 (<http://blog.evjang.com/2016/06/generative-adversarial-nets-in.html>)。我们的演示的完整源代码可以在Github (<https://github.com/AYLIEN/gan-intro> (<https://github.com/AYLIEN/gan-intro>)) 上找到，在这里我们将专注于一些更有趣的部分代码。

首先我们创建“真实”数据分布，一个简单的高斯，平均值为4，标准偏差为0.5。它具有一个样本函数，它从分布返回给定数量的样本（按值排序）。

```
1 class DataDistribution(object):
2     def init(self):
3         self.mu = 4
4         self.sigma = 0.5
5     def sample(self, N):
6         samples = np.random.normal(self.mu, self.sigma, N)
7         samples.sort()
8         return samples
```

我们将尝试学习的数据分布如下所示：



我们还定义生成器输入噪声分布（具有相似的采样功能）。在Eric Jang的例子之后，我们还采用分层采样方法对生成器输入噪声进行分析 - 样本首先在指定范围内均匀生成，然后随机扰动。

```
1 class GeneratorDistribution(object):
2     def __init__(self, range):
3         self.range = range
4
5     def sample(self, N):
6         return np.linspace(-self.range, self.range, N) + \
7             np.random.random(N) * 0.01
```

我们的生成器和判别器网络非常简单。生成器是通过非线性（softplus函数）的线性变换，接着是另一个线性变换。

```
1 def generator(input, hidden_size):
2     h0 = tf.nn.softplus(linear(input, hidden_size, 'g0'))
3     h1 = linear(h0, 1, 'g1')
4     return h1
```

在这种情况下，我们发现重要的是确保判别器比发生器更强大，否则它们没有足够的能力学习从而准确区分生成的和实际的样本。所以我们做了一个更深层的神经网络，具有更大的维度。它使用除了最后一个层之外的所有层中的tanh非线性，其是sigmoid（其输出可以被解释为概率）。

```
1 def discriminator(input, hidden_size):
2     h0 = tf.tanh(linear(input, hidden_size * 2, 'd0'))
3     h1 = tf.tanh(linear(h0, hidden_size * 2, 'd1'))
4     h2 = tf.tanh(linear(h1, hidden_size * 2, 'd2'))
5     h3 = tf.sigmoid(linear(h2, 1, 'd3'))
6     return h3
```

然后，我们可以在TensorFlow图中连接这些部件。我们还为每个网络定义损失函数，生成器的目的是简单地愚弄判别器。

```
1 with tf.variable_scope('G'):
2     z = tf.placeholder(tf.float32, shape=(None, 1))
3     G = generator(z, hidden_size)
4
5 with tf.variable_scope('D') as scope:
6     x = tf.placeholder(tf.float32, shape=(None, 1))
7     D1 = discriminator(x, hidden_size)
8     scope.reuse_variables()
9     D2 = discriminator(G, hidden_size)
10
11 loss_d = tf.reduce_mean(-tf.log(D1) - tf.log(1 - D2))
12 loss_g = tf.reduce_mean(-tf.log(D2))
```

我们使用TensorFlow中的普通GradientDescentOptimizer以指数学习速率衰减为每个网络创建优化器。我们还应该注意，在这里找到好的优化参数需要一些调整。

```
1 def optimizer(loss, var_list):
2     initial_learning_rate = 0.005
3     decay = 0.95
4     num_decay_steps = 150
5     batch = tf.Variable(0)
6     learning_rate = tf.train.exponential_decay(
7         initial_learning_rate,
8         batch,
9         num_decay_steps,
10        decay,
11        staircase=True
12    )
13    optimizer = GradientDescentOptimizer(learning_rate).minimize(
14        loss,
15        global_step=batch,
16        var_list=var_list
17    )
18    return optimizer
19
20 vars = tf.trainable_variables()
21 d_params = [v for v in vars if v.name.startswith('D/')]
22 g_params = [v for v in vars if v.name.startswith('G/')]
23
24 opt_d = optimizer(loss_d, d_params)
25 opt_g = optimizer(loss_g, g_params)
```

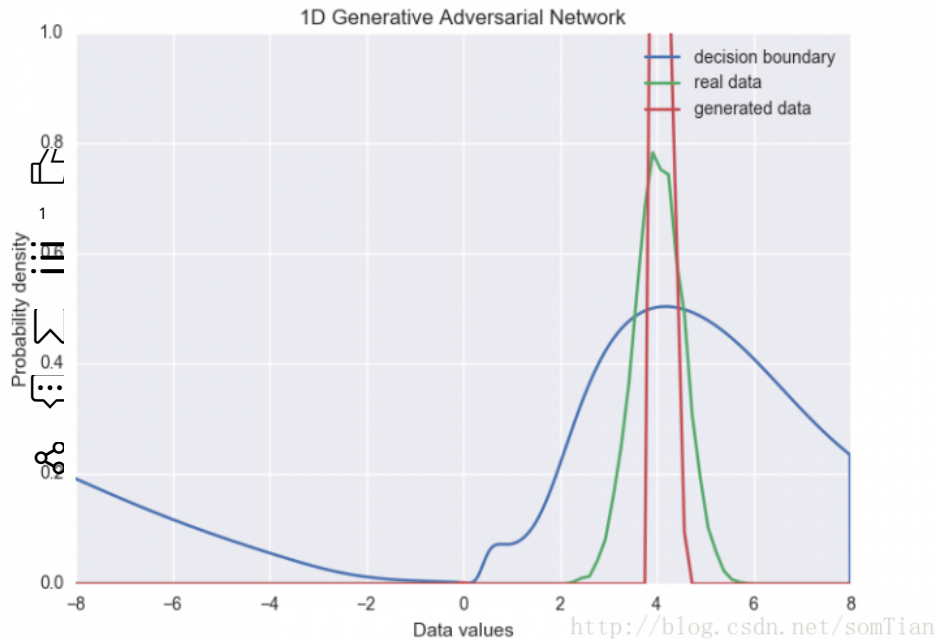
为了训练模型，我们从数据分布和噪声分布中抽取样本，并优化判别器和生成器的参数。

```
1 with tf.Session() as session:
2     tf.initialize_all_variables().run()
3
4     for step in xrange(num_steps):
5         # update discriminator
6         x = data.sample(batch_size)
7         z = gen.sample(batch_size)
8         session.run([loss_d, opt_d], {
9             x: np.reshape(x, (batch_size, 1)),
10            z: np.reshape(z, (batch_size, 1))
11        })
12
13        # update generator
14        z = gen.sample(batch_size)
15        session.run([loss_g, opt_g], {
16            z: np.reshape(z, (batch_size, 1))
17        })
```

以下动画显示了生成器如何在训练过程中学习如何近似数据分布：

<https://youtu.be/mObnwR-u8pc> (<https://youtu.be/mObnwR-u8pc>)

我们可以看到，在训练过程开始时，生成器正在产生与实际数据非常不同的分布。它最终终于学会了相当接近与真实数据（在750帧附近），然后收敛到一个较窄的分布集中在输入分布的平均值。训练后，这两个分布看起来像这样：



这是很直观的。生成器正在从实际数据和我们的判别器中查看各个样本。如果生成器只是在这个简单的例子中产生实际数据的平均值，那么很可能会愚弄判别器。

这个问题有很多可能的解决方案。在这种情况下，我们可以添加一些提前停止的标准，当达到两个分布之间的相似性阈值时暂停训练。然而，如果将这个概念化为更大的问题，即使在简单的情况下也可能难以保证，我们的生成器将始终达到提前停止的意义。一个更有吸引力的解决方案是通过给判别者一次性检查多个示例的能力来直接解决问题。

提高样本多样性

根据Tim Salimans和OpenAI的合作者最近的一篇文章 (<https://arxiv.org/abs/1606.03498>)，生成器崩溃到其输出非常窄的点分布的参数设置的问题是GAN的主要失败模式之一。幸运的是，他们还提出了一个解决方案：允许判别器同时查看多个样本，这是一种称之为“小批量判别”的技术。

在这篇文章中，小批量判别被定义为任何判别器能够查看整批样本以便确定它们是来自生成器还是实际数据的方法。他们还提出了一种更具体的算法，通过建模给定样品与同一批次中的所有其他样品之间的距离来工作。然后将这些距离与原始样品组合并通过鉴别器，因此可以选择在分类过程中使用距离测量值和样品值。

该方法可以大致地总结如下：

- 取出判别器的一些中间层的输出。
- 将其乘以3D张量以产生矩阵（在下面的代码中大小为num_kernels x kernel_dim）。
- 在批量中的所有样本之间计算该矩阵中的行之间的L1距离，然后应用负指数。
- 样本的minibatch 特征是这些取幂距离的总和。
- 将原始输入连接到新建的最小匹配特征的最小匹配层（前一个判别层的输出），并将其作为输入传递给判别的下一层。

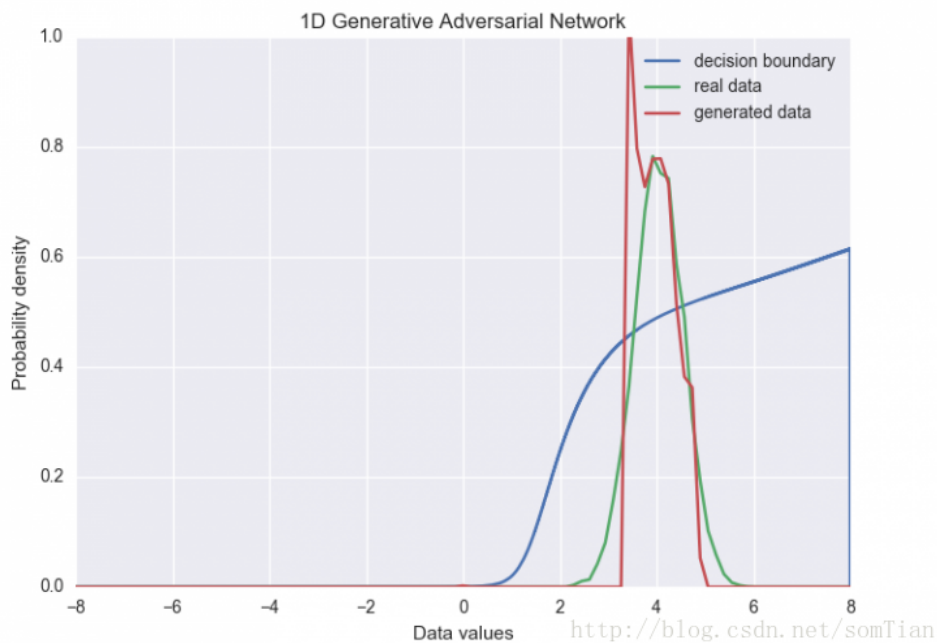
在TensorFlow中：

```
1 def minibatch(input, num_kernels=5, kernel_dim=3):
2     x = linear(input, num_kernels * kernel_dim)
3     activation = tf.reshape(x, (-1, num_kernels, kernel_dim))
4     diffs = tf.expand_dims(activation, 3) - \
5         tf.expand_dims(tf.transpose(activation, [1, 2, 0]), 0)
6     abs_diffs = tf.reduce_sum(tf.abs(diffs), 2)
7     minibatch_features = tf.reduce_sum(tf.exp(-abs_diffs), 2)
8     return tf.concat(1, [input, minibatch_features])
```

我们实施这种 minibatch discrimination 技术，看看它是否有助于我们的示例中生成器输出分布的崩溃。训练期间生成器网络的新行为如下所示。

<https://youtu.be/0r3g7-4bMYU> (<https://youtu.be/0r3g7-4bMYU>)

在这种情况下，很明显，添加 minibatch discrimination 会导致生成器维持原始数据分布的大部分宽度（尽管仍然不完美）。收敛后，分布现在看起来像这样：



minibatch discrimination 的最后一点是，使批量大小作为超参数更为重要。在我们的示例中，我们不得不保持批量相当小（不到16个左右）进行训练。也许仅仅限制对每个距离测量有贡献的样本数量，而不是使用完整批次，但是再次调整另一个参数就足够了。

最后的想法

生成对抗网络是一个有趣的发展，为我们提供了一种新的无监督学习方法。GAN的大部分成功应用一直处于计算机视觉领域，但在这里，我们正在研究将这些技术应用于自然语言处理的方法。如果您正在处理相同的想法，并希望比较想法，请联系我们。

在这方面的一个大问题是如何最好地评估这些模型。在图像域中，至少看看生成的样本是很容易的，虽然这显然不是令人满意的解决方案。在文本领域，这甚至不太有用（除非你的目标是产生散文）。使用基于最大似然训练的生成模型，我们通常可以根据看不见的测试数据的可能性（或可能性的一些下限）产生一些度量，但这在这里是不适用的。一些GAN论文根据生成的样本的核密度估计值产生了似然估计，但是这种技术似乎在较高维空间中分解。另一个解决方案是只评估一些下游任务（如分类）。如果您有任何其他建议，我们很乐意听到您的意见。

更多信息

如果您想了解更多有关GAN的信息，我们建议您从以下出版物开始：

- Generative Adversarial Networks
- Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- Improved Techniques for Training GANs

随意重用我们的GAN代码，当然还要关注我们的博客。欢迎评论，更正和反馈。

原文地址：


An introduction to Generative Adversarial Networks (with code in TensorFlow) (<http://blog.aylien.com/introduction-generative-adversarial-networks-code-tensorflow/>)



相关文章推荐


生成对抗网络的tensorflow实现 (<http://blog.csdn.net/xiaohu2022/article/details/54234263>)

生成对抗网络的tensorflow实现原文地址：<http://blog.evjang.com/2016/06/generative-adversarial-nets-in.html> 这是关于使用te...

 xiaohu2022 (<http://blog.csdn.net/xiaohu2022>) 2017年01月08日 15:51 6038



Tensorflow(1.0)基于对抗生成网络生成明星脸 (http://blog.csdn.net/weixin_36368407/art...)

这个Demo是根据作者@斗大的熊猫和本文原始地址：<http://blog.topspeedsnail.com/archives/10977> 这篇博客来实现的。使用的数据集：Large-scale ...

 weixin_36368407 (http://blog.csdn.net/weixin_36368407) 2017年03月03日 17:52 582

tf24: GANs—生成明星脸 (<http://blog.csdn.net/u014365862/article/details/54380277>)

GANs是Generative Adversarial Networks的简写，中文翻译为生成对抗网络，它最早出现在2014年Goodfellow发表的论文中：Generative Adversar...

 u014365862 (<http://blog.csdn.net/u014365862>) 2017年01月12日 17:51  3466



tensorflow 1.01中GAN(生成对抗网络)手写字体生成例子(MINST)的测试 (<http://blog.csdn.net/sparkeexpert/article/details/72126328>)

为了更好地掌握GAN的例子，从网上找了段代码进行跑了下，测试了效果。具体过程如下：代码文件如下：import tensorflow as tf from tensorflow.examples...

 sparkeexpert (<http://blog.csdn.net/sparkeexpert>) 2017年04月12日 21:00  3258



GAN的理解与TensorFlow的实现 (<http://blog.csdn.net/kwame211/article/details/78211141>)

前言本文会从介绍生成对抗式网络的一些内容，从生成式模型开始说起。到GAN的基本原理，InfoGAN，AC-GAN的基本科普，如果有任何有错误的地方，请随时喷，我刚开始研究GAN这块的内容，希...

 kwame211 (<http://blog.csdn.net/kwame211>) 2017年10月12日 09:49  74

简述生成式对抗网络 (<http://blog.csdn.net/paminy/article/details/61201916>)

本文主要阐述了对生成式对抗网络的理解。首先谈到了什么是对抗样本，以及它与对抗网络的关系，然后解释了对抗网络的每个组成部分，再结合算法流程和代码实现来解释具体是如何实现并执行这个算法的，最后通过给出一个...

 paminy (<http://blog.csdn.net/paminy>) 2017年03月10日 17:43  1421



对抗生成网络的资料小结 (<http://blog.csdn.net/maweifei/article/details/72846316>)

*****...

 maweifei (<http://blog.csdn.net/maweifei>) 2017年06月02日 17:25  475



如何用 TensorFlow 实现生成式对抗网络(GAN) (<http://blog.csdn.net/c2a2o2/article/details/72126328>)

我们来研究一下生成式对抗网络 GAN，并且用 TensorFlow 代码实现。自从 Ian Goodfellow 在 14 年发表了 论文 Generative Adversari...

 c2a2o2 (<http://blog.csdn.net/c2a2o2>) 2017年01月13日 14:09  5560

用tensorflow改写个网络碰到的各种异常问题(数据集cifar) (<http://blog.csdn.net/wang2008start/article/details/72126328>)

用tf改写了个网络结构，这里没有使用keras，tflearn等，是自己手写的。网络结构参照他处的。训练的时候首先遇到的就是各种shape不匹配的问题，来来回回修改几次之后，把下面的几个主要用到的函数...

 wang2008start (<http://blog.csdn.net/wang2008start>) 2017年04月28日 15:12  571

生成对抗网络简介（包含TensorFlow代码示例）【翻译】 (<http://blog.csdn.net/omnispace/article/details/72126328>)


判别模型 vs. 生成模型示例：近似一维高斯分布提高样本多样性最后的思考关于GAN的一些讨论最近，大家对生成模型的兴趣又开始出现（OpenAI关于生成模型的案例）。生成模型可以学习如何生成...

 omnispace (<http://blog.csdn.net/omnispace>) 2017年08月23日 14:48  154

生成对抗网络GANs理解（附代码） ([/sxf1061926959/article/details/54630462](http://blog.csdn.net/sxf1061926959/article/details/54630462))


对抗网络是14年Goodfellow Ian在论文Generative Adversarial Nets中提出来的。记录下自己的理解，日后忘记了也能用于

复习。生成模型和判别模型理解对抗网络，首先要...

 sxf1061926959 (<http://blog.csdn.net/sxf1061926959>) 2017-01-20 12:36 17604


【深度学习】生成对抗网络 **Generative Adversarial Nets** (/shenxiaolu1984/article/details...

介绍非监督深度学习经典论文GAN(Generative Adversarial Nets)

 shenxiaolu1984 (<http://blog.csdn.net/shenxiaolu1984>) 2016-08-17 19:05 30776


简单理解与实验生成对抗网络 **GAN** (/on2way/article/details/72773771)

之前GAN网络是近两年深度学习领域的新秀，火的不行，本文旨在浅显理解传统GAN，分享学习心得。现有GAN网络大多数代码实现使用python、torch等语言，这里，后面用matlab搭建一个简单的GA...

 on2way (<http://blog.csdn.net/on2way>) 2017-05-26 21:31 14796


生成式对抗网络 **GAN**研究进展（一） (/solomon1558/article/details/52537114)

【前言】 本文首先介绍生成式模型，然后着重梳理生成式模型（Generative Models）中生成对抗网络（Generative Adversarial Network）的研究与发展。...

 Solomon1558 (<http://blog.csdn.net/Solomon1558>) 2016-09-14 13:16 40018


对抗生成网络（ **Generative Adversarial Net**) (/stdcoutzyx/article/details/53151038)

现在，生成模型还没有体会到深度学习的利好，在Discriminative模型上，成果如雨后春笋，但在生成模型上，却并非如此。原因如下： - 在最大似然估计及相关策略上，很多概率计算的模拟非常难 - ...

 xinzhangyanxiang (<http://blog.csdn.net/xinzhangyanxiang>) 2016-11-13 19:59 14478

生成式模型 & 生成对抗网络——资料梳理（专访资料 + 论文分类） (/solomon1558/arti...

文献整理 题目主要内容 ...

 Solomon1558 (<http://blog.csdn.net/Solomon1558>) 2016-08-27 23:52 13523


GAN：生成式对抗网络介绍和其优缺点以及研究现状 (/bixiwen_liu/article/details/53909...

本博文是转载自一篇博文，介绍GAN（Generative Adversarial Networks）即生成式对抗网络的原理以及GAN的优缺点的分析和GAN网络研究发展现状

 Bixiwen_liu (http://blog.csdn.net/Bixiwen_liu) 2016-12-28 16:43 8700

生成对抗网络 (/wangli0519/article/details/73549363)

我们提出一个框架来通过对抗方式评估生成模型，

 wangli0519 (<http://blog.csdn.net/wangli0519>) 2017-06-21 15:06 238



卷积神经网络 **Lenet-5**实现 (/geekmanong/article/details/50605340)

原文地址：<http://blog.csdn.net/hjmce/article/details/47323463> 作者：hjmce 卷积神经网络算法是n年前就有的算法，只是近年来因为深度学习相...

 geekmanong (<http://blog.csdn.net/geekmanong>) 2016-01-29 10:16 19406

生成对抗网络学习笔记1----论文Generative Adversarial Nets (/liuxiao214/article/details/...

1、阅读论文：Goodfellow I J, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[C]// Internation...

 liuxiao214 (<http://blog.csdn.net/liuxiao214>) 2017-05-26 10:28  1098



1

