深度学习最全优化方法总结比较及在tensorflow实现

梯度下降局最优的，因为此时只有唯一的局部最优点。而实际上深度学习模型是一个复杂的非结构，问题，这用梯度下降算法可能会陷入局部最优，这应该是最头疼的问题。这点和进化算法如遗很类似收敛到全局最优。因此，我们注定在这个问题上成为“高级炼丹师”。可以看到，梯度下降算法中一个重要的参数是学习速率，适当的学习重要：时收敛速发散。理想的梯度下降算法要满足两点：收敛速度要快；能全局收敛。为了这个理现了很降算法的

SGD

梯度下降神经网络模型训练最常用的优化算法。梯度下降算法背后的原理：目标函数J(θ)关于参数θ的梯度上升最快参数沿着梯度相反的方向前进一个步长(学习速率)，就可以实现目标函数的下降。参数更新公式如下

$$\theta \leftarrow$$

其中 ∇

根据计算算法又可以分为批量梯度下降算法（Batch Gradient Descent），随机梯度下降算法（Stochas escent）ient Descent）。

- 批量梯度下降算法，J(θ)是在整个训练集上计算的，如果数据集比较大，可能会面临内存不足问题，而且其收敛速度一般比较慢。
- 随机梯度下降算法，J(θ)是针对训练集中的一个训练样本计算的，又称为在线学习，即得到了一个样本，就可以执行一次参数更新。所以其收敛速度会快一些，但是有函数值震荡现象，因为高频率的参数更新导致了高方差。
- 小批量梯度下降算法，是折中方案，J(θ)选取训练集中一个小批量样本计算，这样可以保证训练过程更稳定，而且采用批量训练方法也可以利用矩阵计算的优势。这是度下降算法。

```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.001).minimize(loss)
```

momentum

SGD方法的一个缺点是，其更新方向完全依赖于当前的batch，因而其更新十分不稳定，每次迭代计算的梯度含有比较大的噪音。解决这一问题的一个是引入momentum，momentum即动量，是BorisPolyak在1964年提出的，其基于物体运动时的惯性：将一个小球从山顶滚下，其初始速率很慢，但在速率很快增加，并最终由于阻力的存在达到一个稳定速率，即更新的时候在一定程度上保留之前更新的方向，同时利用 当前batch的梯度 微调最终这样一来，可以在一定程度上增加稳定性，从而学习地更快，并且还有一定摆脱局部最优的能力。

其更新方程如下：

$$\mathbf{m} \leftarrow \gamma \cdot \mathbf{m} + \eta \cdot \nabla_\theta J(\theta)$$

$$\theta \leftarrow \theta - \mathbf{m}$$

可以看到，参数更新时不仅考虑当前梯度值，而且加上了一个动量项γm，但多了一个超参γ，通常γ设置为0.5，直到初始学习稳定，然后增加到0.9或始梯度下降算法，动量梯度下降算法有助于加速收敛。当梯度与动量方向一致时，动量项会增加，而相反时，动量项减少，因此动量梯度下降算法可以荡过程。

```
tf.train.MomentumOptimizer(learning_rate=learning_rate,momentum=0.9)
```

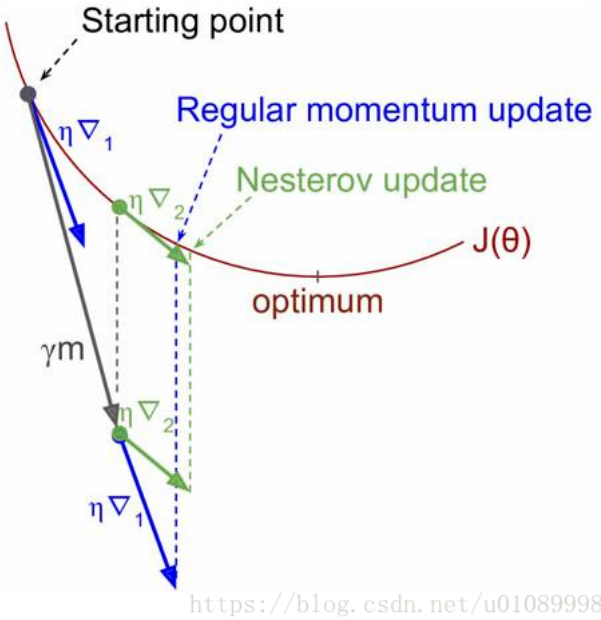
NAG

NAG（Nesterov Accelerated Gradient），，由Ilya Sutskever(2012 unpublished)在Nesterov工作的启发下提出的。对动量梯度下降算法的改进版本，其变化之处在于计算“超前梯度”更新动量项 γm，具体公式如下：

$$\mathbf{m} \leftarrow \gamma \cdot \mathbf{m} + \eta \cdot \nabla_\theta J(\theta - \gamma \cdot \mathbf{m})$$

$$\theta \leftarrow \theta - \mathbf{m}$$

既然参数要沿着动量项 γm更新，不妨计算未来位置（θ -γm）的梯度，然后合并两项作为最终的更新项，其具体效果如图1所示，可以看[illegible]的加速



momentum基础上设置 use_nesterov=True

```
tf.train.MomentumOptimizer(learning_rate=learning_rate,momentum=0.9, use_nesterov=True)
```

AdaGrad

AdaGrad是Duchi在2011年提出的一种学习速率自适应的梯度下降算法。在训练迭代过程，其学习速率是逐渐衰减的，经常更新的参数其学习速率衰减种自适应算法。其更新过程如下：

$$\epsilon_n = \frac{\epsilon}{\delta + \sqrt{\sum_{i=1}^{n-1} g_i \odot g_i}}$$

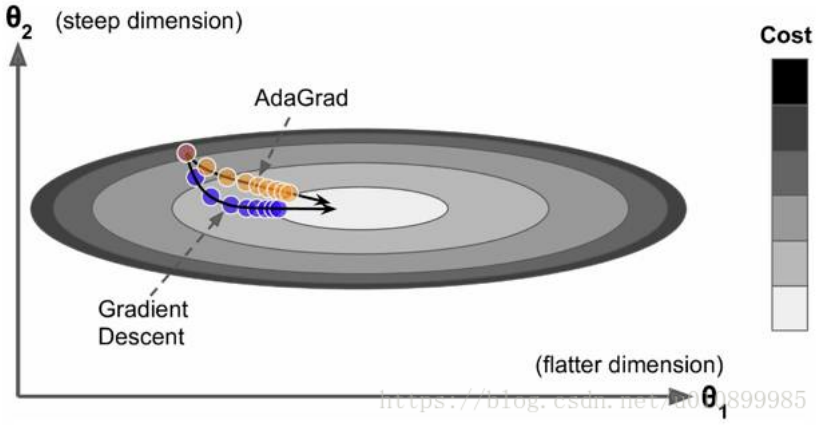
每步迭代过程:

1. 从训练集中的随机抽取一批容量为m的样本{x1,…,xm},以及相关的输出yi
2. 计算梯度和误差,更新r,再根据r和梯度计算参数更新量:

$$\hat{g} \leftarrow + \frac{1}{m} \nabla_\theta \sum_i L(f(x_i;\theta), y_i)$$
$$r \leftarrow r + \hat{g} \odot \hat{g}$$
$$\triangle\theta = -\frac{\epsilon}{\delta + \sqrt{r}} \odot \hat{g}$$
$$\theta \leftarrow \theta + \triangle\theta$$

其中，全局学习速率 ϵ, 初始参数 θ，梯度平方的累计量r初始化为0)， **δ（通常为10^−7）是为了防止分母的为 0**。

由于梯度平方的累计量**r**逐渐增加的，那么学习速率是衰减的。考虑下图所示的情况，目标函数在两个方向的坡度不一样，如果是原始的梯度下降算法时收敛速度比较慢。而当采用AdaGrad，这种情况可以被改善。由于比较陡的方向梯度比较大，其学习速率将衰减得更快，这有利于参数沿着更接近动，从而加速收敛。对于每个参数，随着其更新的总距离增多，其学习速率也随之变慢。



缺点: 任然要设置一个变量ϵ ,经验表明，在普通算法中也许效果不错，但在深度学习中，深度过深时会造成训练提前结束。

```
tf.train.AdagradOptimizer(learning_rate=0.001).minimize(loss)
```

Adadelta

Adadelta是对Adagrad的扩展，最初方案依然是对学习率进行自适应约束，但是进行了计算上的简化。
Adagrad会累加之前所有的梯度平方，而**Adadelta**只累加固定大小的项，并且也不直接存储这些项，仅仅是近似计算对应的平均值。即：

$$n_t = \nu * n_{t-1} + (1-\nu) * g_t^2$$

$$\Delta\theta_t = -\frac{\eta}{\sqrt{n_t + \epsilon}} * g_t$$

其中，η是学习率，gt 是梯度

在此处Adadelta其实还是依赖于全局学习率的，但是作者做了一定处理，经过近似牛顿迭代法之后：

$$E|g^2|_t = \rho * E|g^2|_{t-1} + (1-\rho) * g_t^2$$

$$\Delta x_t = -\frac{\sqrt{\sum_{r=1}^{t-1}\Delta x_r}}{\sqrt{E|g^2|_t + \epsilon}}$$

其中，E代表求期望。此时，可以看出Adadelta已经不用依赖于全局学习率了。

```
tf.train.AdadeltaOptimizer(learning_rate=0.001).minimize(loss)
```

RMSProp

RMSprop是对Adagrad算法的改进，主要是解决。其实思路很简单，类似Momentum思想，引入一个衰减系数，让梯度平方的累计量r 每回合都衰减一

$$\hat{g} \leftarrow +\frac{1}{m}\nabla_\theta \sum_i L(f(x_i;\theta), y_i)$$
$$r \leftarrow \rho r + (1-\rho)\hat{g} \odot \hat{g}$$
$$\triangle\theta = -\frac{\epsilon}{\delta + \sqrt{r}} \odot \hat{g}$$
$$\theta \leftarrow \theta + \triangle\theta$$

其中，衰减系数ρ

decay: 衰减率
epsilon: 设置较小的值，防止分母的为 **0.**

```
tf.train.RMSPropOptimizer(learning_rate=0.001,momentum=0.9, decay=0.9, epsilon=1e-10)
```

优点：
- 相比于AdaGrad,这种方法有效减少了出现梯度爆炸情况，因此避免了学习速率过快衰减的问题。
- 适合处理非平稳目标，对于RNN效果很好

缺点：
- 又引入了新的超参—衰减系数ρ
- 依然依赖于全局学习速率，

总结：RMSprop算是Adagrad的一种发展，和Adadelta的变体，效果趋于二者之间。

Adam

自适应矩估计（daptive moment estimation，Adam），是Kingma等在2015年提出的一种新的优化算法，本质上是带有动量项的RMSprop，结合了RMSprop算法的思想。它利用梯度的一阶矩估计 和 二阶矩估计 动态调整每个参数的学习率。

具体实现每步迭代过程:

1. 从训练集中的随机抽取一批容量为m的样本{x1,…,xm},以及相关的输出yi
2. 计算梯度和误差,更新r和s,再根据r和s以及梯度计算参数更新量：

$$g \leftarrow +\frac{1}{m}\nabla_\theta \sum_i L(f(x_i;\theta),y_i)$$
$$s \leftarrow \rho_1 s + (1-\rho_1)g$$
$$r \leftarrow \rho_2 r + (1-\rho_2)g \odot g$$
$$\hat{s} \leftarrow \frac{s}{1-\rho_1}$$
$$\hat{r} \leftarrow \frac{r}{1-\rho_2}$$
$$\triangle\theta = -\epsilon\frac{\hat{s}}{\sqrt{\hat{r}}+\delta}$$
$$\theta \leftarrow \theta + \triangle\theta$$

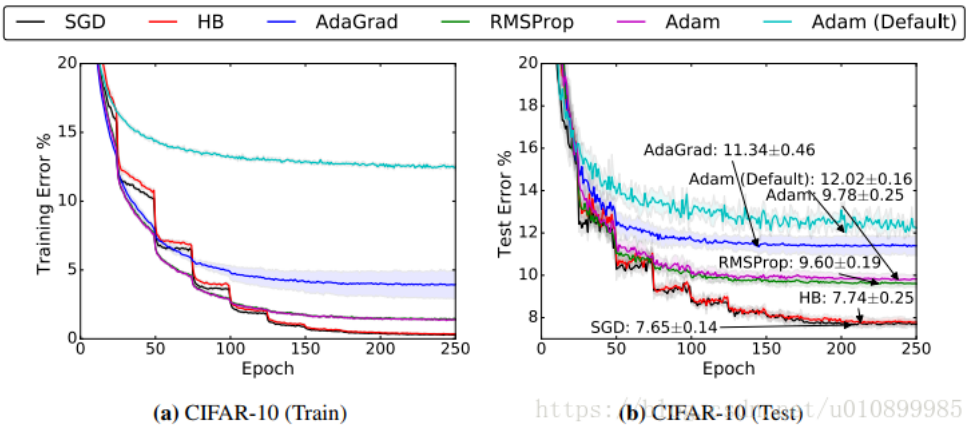
其中，一阶动量s，二阶动量r（初始化为0），一阶动量衰减系数ρ1, 二阶动量衰减系数ρ2

超参数的建议值是ρ1=0.9，ρ2 =0.999，epsilon: 设置较小的值，防止分母的为 0。

```
class AdamOptimizer
def __init__(self, learning_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-8, use_locking=False, name="Adam")
Construct a new Adam optimizer.
optimizer = tf.train.AdamOptimizer(learning_rate=0.001).minimize(loss)
```

实践

各种优化方法在CIFAR-10图像识别上比较：



(a) CIFAR-10 (Train) (b) CIFAR-10 (Test)

图片来源：The Marginal Value of Adaptive Gradient Methods in Machine Learning

- 自适应优化算法在训练前期阶段在训练集上收敛的更快，但是在测试集上反而并不理想。
- 用相同数量的超参数来调参，SGD和SGD +momentum 方法性能在测试集上的额误差好于所有的自适应优化算法，尽管有时自适应优化算法在训练集上的loss更小，但在测试集上的loss却依然比SGD方法高，

总结：

1. 对于稀疏数据，优先选择学习速率**自适应的算法如RMSprop和Adam算法**，而且最好采用默认值，大部分情况下其效果是较好的
2. SGD通常训练时间更长，容易陷入鞍点，但是在好的初始化和学习率调度方案的情况下，结果更可靠。
3. 如果要求**更快的收敛，并且较深较复杂的网络时，推荐使用学习率自适应的优化方法。例如对于RNN之类的网络结构,Adam速度快,效果好,而对于CN 的网络结 entum 的更新方法要更好（常见国际顶尖期刊常见优化方法）.**
4. Adadelta，RMSprop，Adam是比较相近的算法，在相似的情况下表现差不多。
5. 在想使用带动量的RMSprop，或者Adam的地方，大多可以使用Nadam取得更好的效果。
6. 特别注意学习速率的问题。学习速率设置得非常大，那么训练可能不会收敛，就直接发散了；如果设置的比较小，虽然可以收敛，但是训练时间可能无 。理想 **刚开始设置较大，有很快的收敛速度，然后慢慢衰减，保证稳定到达最优**
7. 其实还有很多方面会影响梯度下降算法，如**梯度的消失与爆炸**，梯度下降算法目前无法保证全局收敛还将是一个持续性的数学难题。

参考：

[1]Adagrad
[2]RMSprop[Lecture 6e]
[3]Adadelta
[4]Adam
[5]Nadam
[6]On the importance of initialization and momentum in deep learning

深度学习中的优化不同于一般优化算法 500

一、经验风险最小化 1、机器学习中我们关注某些性能度量P，其定义在测试集上并且可能不可解。我们需要间接优化P。我们通过降低代价函数J(&#x03…

深度学习中的优化问题 136

一、优化问题的挑战 绝大多数深度学习中的目标函数都很复杂。因此，很多优化问题并不存在显示解（解析解），而需要使用基于数值方法的优化算法…

神经网络在TensorFlow实现 2028

1.引言1.1神经网络的术语 1.偏置bias： 2.激活函数：sigmoid函数;tanh函数；Relu函数。 3.损失函数：最小平方误差准则（MSE）、交叉熵（cross-...

tensorflow CIFAR-10 比较好的代码示例 182

http://www.wolfib.com/

相关热词 深度学习深度学习 深度学习和 深度学习will 深度学习（ in深度学习

常见优化算法 (tensorflow对应参数) 538

转载出处：http://www.2cto.com/kf/201612/572613.html 常见算法 SGD ? 1 x+= -learning...

深度学习优化算法总结（cs231n） 474

https://zhuanlan.zhihu.com/p/21798784?refer=intelligentunit

深度学习各种优化函数详解 9498

深度学习中有众多有效的优化函数，比如应用最广泛的SGD，Adam等等，而它们有什么区别，各有什么特征呢？下面就来详细解读一下一、先来看看有…

深度学习优化方法比较 1577

看到一篇比较不错的文章，比较了深度学习中的各种优化方法，可以看这篇博客...

深度学习中各种优化方法详解 183

这篇文章主要参考Keras Documentation，另外增加了一些我的理解，希望尽量写的简洁明了 如何使用Optimizer 这里我们定义了一个简单的FC网络用来…

深度学习（一）——优化方法总结 40

转自：https://blog.csdn.net/u010089444/article/details/76725843 1. SGD Batch Gradient Descent 在每一轮的训...

深度学习优化方法总结比较（SGD，Adagrad，Adadelta，Adam，Adamax，Nadam） 643

作者：ycszen 转载自：https://zhuanlan.zhihu.com/p/22252270前言 （标题不能再中二了）本文仅对一些常见的优化方法进行直观介绍和简单的比较，…

深度学习中优化方法总结 1573

最近在看Google的Deep Learning一书，看到优化方法那一部分，正巧之前用tensorflow也是对那些优化方法一知半解的，所以看完后就整理了下放上来…

深度学习中的数学与技巧(0)：优化方法总结比较（sgd/momentum/Nesterov/adagrad/adadelta） 2224

reference: http://blog.csdn.net/luo123n/article/details/48239963 前言 这里讨论的优化问题指的是，给定目标函数f(x)，我们需要找...

深度学习常用优化方法 1268

深度解读最流行的优化算法：梯度下降 【本文转载自机器之心 翻译：沈泽江 原文地址：http://www.jiqizhixin.com/article/1857】梯度下降法，是当今最…

深度学习通用策略：SGD优化方法总结 1022

转载：https://zhuanlan.zhihu.com/p/22252270（标题不能再中二了）本文仅对一些常见的优化方法进行直观介绍和简单的比较，各种优化方法的详细内…

tensorflow实现最基本的神经网络 + 对比GD、SGD、batch-GD的训练方法 356

-*- coding:utf-8 -*- # 将tensorflow 引入并命名tf import tensorflow as tf # 矩阵操作库numpy，命名为np impo...

【TensorFlow】优化方法optimizer总结（SGD，Adagrad，Adadelta，Adam，Adamax，Nadam… 2479

本文仅对一些常见的优化方法进行直观介绍和简单的比较，主要是一阶的梯度法，包括SGD, Momentum, Nesterov Momentum, AdaGrad, RMSProp, Ad…

tensorflow实战系列（三）一个完整的例子 2218

tensorflow实战系列（三）一个完整的例子 链接： http://www.cnblogs.com/whu-zeng/p/6343312.html #!/usr/bin/env...

tensorflow 神经网络解决二分类问题 1109

import tensorflow as tf from numpy.random import RandomState # 定义训练数据batch大小 batch_si...

深度学习总结(五)——各优化算法 1396

一、各优化算法简介1. 批量梯度下降（Batch gradient descent，BGD）θ=θ−η·∇θJ(θ)θ = θ - η \cdot \nabla_θ J(θ) 每迭代一步，都要用到训...

基于tensorflow的深度学习框架优化 169

以下3个模型均能完成对多个类别的图像识别的过程。Part 1:Tensorflow简单框架的搭建，第一步：对图片进行预处理，包括灰度化、尺寸改变等，并构…

深度学习优化算法总结 415

本文基于目前深度学习中使用较多的优化学习算法进行总结。 1 深度学习中的优化算法 优化算法之前讨论两个问题： (1) 局部最小值问题 …

深度学习:基于梯度下降不同优化算法的比较总结 2274

这里讨论的优化问题指的是，给定目标函数f(x)，我们需要找到一组参数x，使得f(x)的值最小。 本文以下内容假设读者已经了解机器学习基本知识，和梯…

optimizer优化算法总结 1万

优化方法总结参考深度学习最全优化方法总结比较An overview of gradient descent optimization algorithms目录优化方法总结 SGD 1 Batch gr...

TensorFlow高效读取数据的方法 6.1万

概述关于Tensorflow读取数据，官网给出了三种方法： 供给数据(Feeding)： 在TensorFlow程序运行的每一步， 让Python代码来供给数据。 从文件读取…

【TensorFlow】激活函数（Activation Functions）原理解析（十二） 1216

神经网络结构的输出为所有输入的加权和，这导致整个神经网络是一个线性模型。如果将每一个神经元的输出通过一个非线性函数，那么整个神经网络的…

深度学习（一）：虚拟机Linux系统搭建CPU TensorFlow 467

最近重装了一遍，在此捋一捋：1.在win7下进行Linux虚拟机搭建参考链接：在win7下进行Linux虚拟机搭建对于Linux系统。最易于理解的版本就是著名…

Tensorflow梯度下降常用的常用优化方法 516

Tensorflow梯度下降常用的常用优化方法1.tf.train.exponential_decay() 指数衰减学习率：#tf.train.exponent...

tensorflow优化细节（哪些变量要优化，优化比例）的手动控制方法 60

tensorflow中，从建好loss的graph到输入run()函数的op间还有一个slim.learning.create_train_op的步骤。了解这个步骤的内容就能打通整个tensorflo...

深度学习——sgd等优化方法比较 2261

1、adagrad相比于sgd和momentum更加稳定，即不需要怎么调参。 2、精调的sgd和momentum系列方法无论是收敛速度还是precision都比adagrad要好…

深度学习中常见的优化算法比较 1123

SGD Basic SGD the baisc sgd is: # Vanilla update x += - learning_rate * ...

66 10 24 10

等级： 博客 4 访问： 9万+

积分： 1261 排名： 5万+

勋章：

最新评论

Tensorboard “No s...
Lelouc_CC：[reply]u010899985[/reply] 谢谢

Tensorboard “No s...
u010899985：[reply]Lelouc_CC[/reply] 可以在pycharm 中指定安装版本号，或者卸载...

Tensorboard “No s...
Lelouc_CC：我是一个初学者，麻烦楼主，tensorflow版本怎么转换?

如何从宿主机拖动复制文件到虚拟机V...
zhuzongzhi856：专业

如何从宿主机拖动复制文件到虚拟机V...
azureeeeeeeeee：解决了！十分感谢哇！！