caffe study (1) - 数据结构(1)

 (2014-10-13 11:33:35)

[[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)转载▼](javascript:;)

|  |  |
| --- | --- |
|  |  |

以下主要是对于Caffe主页文档的总结  
  
1. 结构的生成：caffe的基本结构是采用google的proto库自动生成的，基本流程就是定义一个配置文件，扩展名为proto，调用proto库的编译器编译这个文件可以生成相应的类的c++的代码。具体的可以参见proto库的介绍。  
  
下面来介绍caffe中用到的一些结构  
  
2.Blob结构：  
  
a. blob是一个标准的Array，主要负责caffe中数据的存储(stores)、关联(communicates)、以及数据的操作(manipulates)。数据在网络结构中要经过正向以及反向转播的过程，在这个过程中要对于数据进行存储、数据之间的通讯、以及数据的操作，blob就是负责这个工作的。  
  
在具体的形式上blob是一个4-D结构的array，是按照（Num,Channels,Height and Width）的顺序存储的，这里的Num相当于minibatch SGD中的batch这样一个概念。此处据作者的解释是应为legacy reason。  
  
b. blob中数据的dimentions为Num N\*channel K \* Height H \* Width W.内存是行优先的(row-major)。访问数据的时候按照如下的规则来访问index(n,k,h,w) 在物理上位于index((n\*K + k) \*H + h)\*W + w. 这里要注意，index(n,k,h,w)实际上访问的是内存中(n+1,k+1,h+1,w+1)位置的数据，这是因为索引是从0开始的。  
  
N是batch size of data，在ImageNet中这个值是256，Channls是feature dimention对于RGB图像来说K = 3. blob的维度是根据layer的type和configuration不同计算不同的，对于一个有96个filter，11\*11的spatialdimension，3个input的blob的conv layer来说，其dimention为96\*3\*11\*11，对于一个有1000个ouput channel以及1024个input channnels的 fully-connected layer来说blob的维度为1\*1\*1000\*4096。

（@[Leo\_Shaw](http://blog.sina.com.cn/u/2187639384)指出此处的Num应该是filter的数目，这是我看文档不严谨的地方，非常感谢@[Leo\_Shaw](http://blog.sina.com.cn/u/2187639384)指出这个问题。上面的解释不完全错，但是只说明了一个方面，此处做一个小的修正：

在caffe文档中指出 blob的dimensions vary according to the type and configuration of the layers，N代表的含义在不同的层中是不一样的，在输入的Data层中，N代表了mini batch size，也就是一次可以处理的图像的数目，在conv层中，N确实代表了filter的数目，这是代码结构定义方面的问题，和算法本身没有关系。）

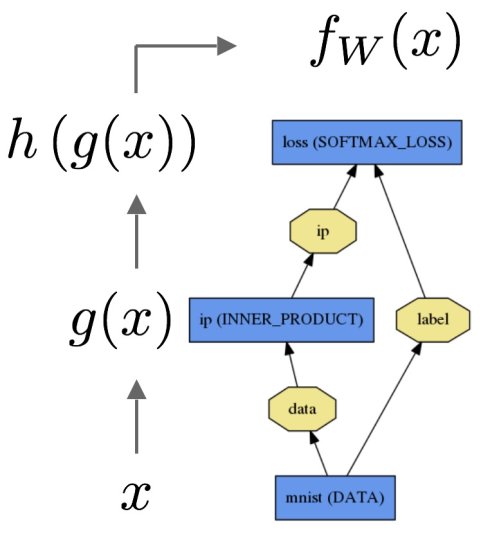
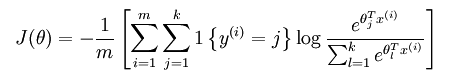
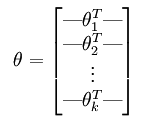
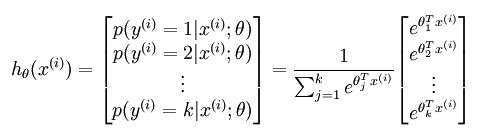
c.blob中的数据访问方法：const方式以及mutable方式，前者访问不能改变数据，后者访问可以改变数据。这种设计主要为了cpu和gpu之间数据的同步，具体的还要深入研究一下。这个有GPU和CPU两个版本。  
  
d.blob中数据分为两个chunks一个是data一个是diff，前者是正常的的传递的数据，后者是网络计算的gradient。  
  
3. Layer结构：  
  
a. Layer是一个核心的结构，它主要定义了基本的计算单元，也就是每一层的具体形式。这个层包括convolve filter、pools、inner products、loss等不同定义  
  
b. 每一个Layer定义了三个核心的计算：  
     
    1）Setup：初始化layer和its connections  
    2）Forward：根据从bottom来的输入计算输出送到top  
    3）Backward：根据top output的gradient计算input的gradient，然后输送到bottom。同时还会计算相对于parameters的gradient，并在内部存储  
  
  
4. Net结构：  
  
a.Net是一个DAG/directed acyclic graph，其节点就是一个个的Layer结构，net从data layer开始，以loss layer结束。  
  
b.通过调用Net::Init()来初始化net，初始化做了两个工作，一个是创建blobs和layers，调用layer::setup()，一个是做一些bookkeeping的事情，比如验证网络结构的有效性，输出一些log等。在net创建后，则根据Caffe::mode()使用cpu或者gpu进行计算。  
  
5. 总结  
  
在caffe里面这几个关键的结构中，blob负责的是数据的存储，Layer负责了数据计算的传输，而net则将前两者又包了一层，net将会扔到slover(关于slover后边再分析)中进行最终的学习。

## caffe study(2) 关于forward和backward - forward

http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif (2014-10-13 11:35:10)

[[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)转载▼](javascript:;)

|  |  |
| --- | --- |
| 标签： [股票](http://search.sina.com.cn/?c=blog&q=%B9%C9%C6%B1&by=tag) |  |

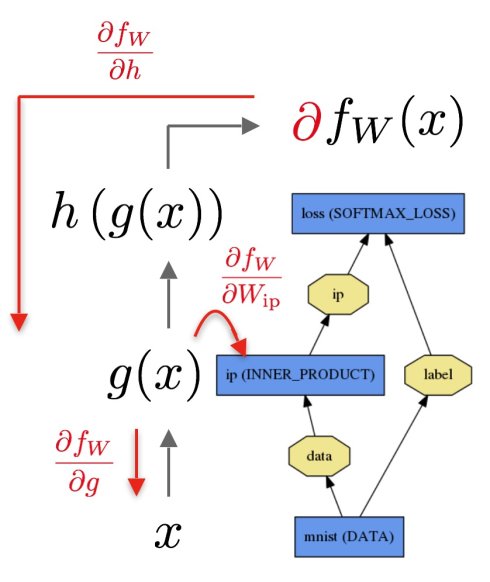
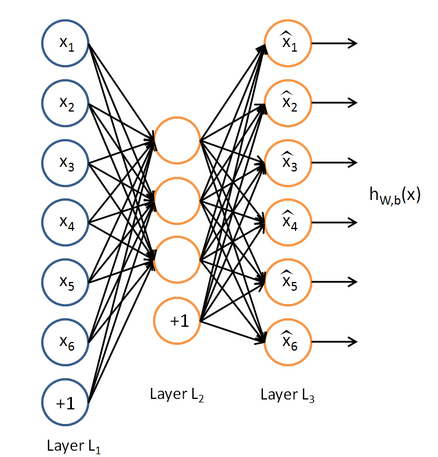
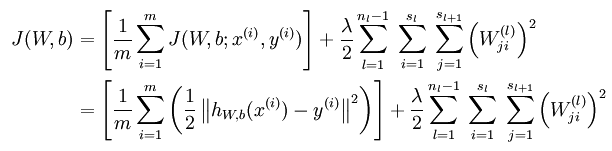
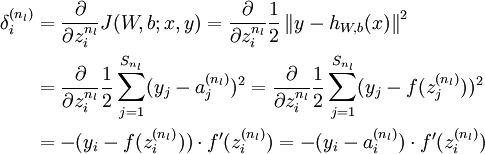
在caffe中这两个过程是net中本质的计算过程，这个数据是指的目标函数的生成、导数的推导等。下面首先来看一下forward所处理的工作。  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v2or&url=http://album.sina.com.cn/pic/004j58Jbzy6MMeiFLY965)  
  
  
  
  
  
  
  
  
  
  
forward过程主要是根据layer的input来生成layer的output。下面以一个三层的网络为例来说明这个过程。此处采用的loss函数为softmax。  
  
基本的hypothesis采用的是Logistic函数，此时输出y的取值为{0，1}  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v2or&url=http://album.sina.com.cn/pic/004j58Jbzy6MMemIJRidc)(1)  
在多项分布中hypothesis和参数theta可以记为，注意这时候y的取值为{1,…,k}  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v2or&url=http://album.sina.com.cn/pic/004j58Jbzy6MMemJr3Z9b)(2)  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v2or&url=http://album.sina.com.cn/pic/004j58Jbzy6MMemwzpQ6e)(3)  
log后可以得到如下的loss  
[caffe <wbr>study(2) <wbr>关于forward和backward <wbr>- <wbr>forward](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v2or&url=http://album.sina.com.cn/pic/004j58Jbzy6MMemOTCl35)(4)  
这个loss的含义相当于ouput是input的Logistic函数的归一化的和，这对于每一个输出节点，如果此时label输出为1，则在这个节点上的logistic函数的输出权重为1，其他节点输出权重为0.这个函数就是softmax loss的具体形式。  
  
这里面g(x)=theta'  \*  x, h就是(2)式，f就是最后的公式(4)，forward的整个过程就是根据输入x，得到输出loss J(theta)的一个过程。  
  
在非loss层，输出到h为止。

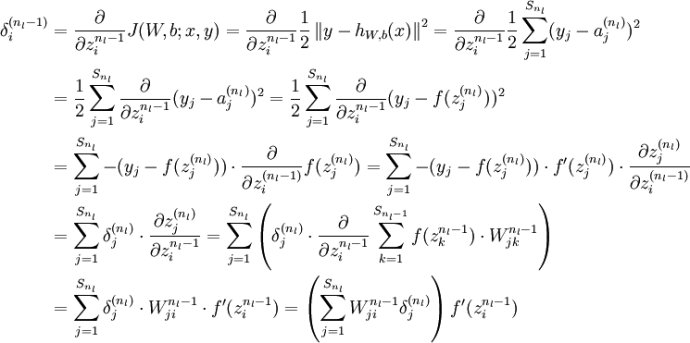
## caffe study(2) 关于forward和backward - backward

http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif(2014-10-13 11:41:07)

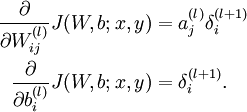
[[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)转载▼](javascript:;)

|  |  |
| --- | --- |
| 标签： [股票](http://search.sina.com.cn/?c=blog&q=%B9%C9%C6%B1&by=tag) |  |

1. 在本质上backward解决的是导数计算的问题，用的就是BP算法，所谓的BP算法就是使用当前层权重对于当前误差加权然后传递到下层节点的一个流程。下图是caffe的doc中示意的，这里面有两部分，一个部分就是向下传递的对于g的偏导数，这个数值就是反向传播的输出;另外一部分就是在当前层中对于当前权重的求导。  
  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v2os&url=http://album.sina.com.cn/pic/004j58Jbzy6MMeJ4H7h5f)  
  
  
2. 下面将通过一个简单的三层网络来说明一下caffe的backward过程所传递的内容的具体形式。  
  
  
a. 如下图所示，这是一个简单的NN的结构，其输出层就一个节点。其中x代表输入数据，a代表每一层的激活值，对应h。在下面的描述中采用z来对应g。  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v2os&url=http://album.sina.com.cn/pic/004j58Jbzy6MMeUCaSR01)  
  
  
在加入正则项之后，loss形式为  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v2os&url=http://album.sina.com.cn/pic/004j58Jbzy6MMeISDJP05)  
  
b.  对于输入层net的求导  
  
对于第n\_{l} 输出层 的每个输出单元i ，我们根据以下公式计算对于z的导数，把z看成一个局部net的话，这个就是对于net进行的求导。  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v2os&url=http://album.sina.com.cn/pic/004j58Jbzy6MMeIHT4Yb8)  
其中s\_{nt}为当前输出层的神经元的个数，a为第j个神经元的激活值，这个值是通过激活函数f计算得到（如的logistic函数），f的输入就是z。因为当  i != j  的时候，导数为0，因此得到了第三行的求导结果。  
  
c. 对于非输入层net的求导

  
根据基本的链式法则可以推导到倒数第二行，里面的导数部分就变成了当前层的每一个神经元z\_{j}对于下一层的z\_{i}求导的一个过程，将z\_{j}做一个展开，就是z\_{i}层激活的一个加权和，如b所示，得到了最终的对于非输入层的导数。其中W\_{ji}指的的是输出层和隐藏层之间的权重链接。

这个导数分为两部分，一部分就是括号内的，是上层导数在当前层节点上的加权和，这个定义了误差专递的方式，也就是反向传播的一个核心；第二项就是当前层激活函数对于net的导数。这样就得到了一个反向转播的递推公式。caffe中backward的就是在传递每一个节点的导数的计算值。  
  
d. 在传递完基本的残差后，在本层还需要计算的就是对于当前w\_{ij}的一个求导。只需要对于上述结果后边再加入一个net对于w\_{ij}的求导就可以了，令l = n - 1， n-2，…

[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v2os&url=http://album.sina.com.cn/pic/004j58Jbzy6MMeIlLdM6a)

## caffe study(3) 关于激活函数以及loss function

http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif (2014-11-03 14:50:55)

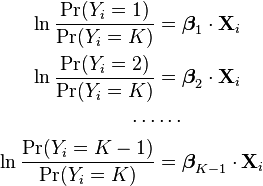
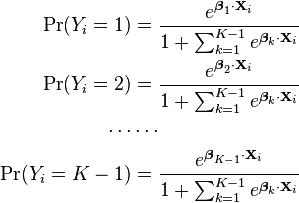
[[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)转载▼](javascript:;)

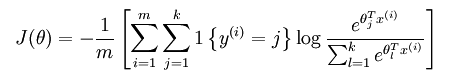
|  |  |
| --- | --- |
| 标签： [股票](http://search.sina.com.cn/?c=blog&q=%B9%C9%C6%B1&by=tag) |  |

loss 是估计值和真实值之映射到某一空间的误差，而loss function就是这种误差的描述形式，loss function反映出了对于问题的定义。在caffe中，包含了常用的loss function，主要有以下几种：  
  
MULTINOMIAL\_LOGISTIC\_LOSS  
SIGMOID\_CROSS\_ENTROPY\_LOSS  
SOFTMAX\_LOSS  
EUCLIDEAN\_LOSS  
HINGE\_LOSS  
INFOGAIN\_LOSS  
  
下面来看一下他们的具体形式（公式和世界的代码有些出入，下面材料的绝大部分来自于wiki）  
  
**1. EUCLIDEAN\_LOSS**  
  
这个loss的具体含义就是所有样本估计值和预测值的欧式距离平方的均值，也就是均方根误差(MSE)。  
  
[caffe <wbr>study(3) <wbr>关于激活函数以及loss <wbr>function](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjXF5YgG42)  
  
假设模型结果与测量值 误差满足，均值为0的高斯分布，即正态分布。这个假设是靠谱的，符合一般客观统计规律。  
数据x与y的条件概率：  
  
[caffe <wbr>study(3) <wbr>关于激活函数以及loss <wbr>function](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjXFqUu2c6)  
若使 模型与测量数据最接近，那么其概率积就最大。概率积，就是概率密度函数的连续积，这样，就形成了一个最大似然函数估计。对最大似然函数估计进行推导，就得出了求导后结果： 平方和最小公式  
  
  
MSE误差的优点是执行简单，较容易理解，缺点就是强制预测和标注要exactly的匹配，也就是一个非0即1的概念，本节点和其他节点是独立的，这样带来的问题是会导致最后的train出来的model有可能并不是那么准确。  
  
**2. MULTINOMIAL\_LOGISTIC\_LOSS**  
  
如果一个事件发生的概率为：

[caffe <wbr>study(3) <wbr>关于激活函数以及loss <wbr>function](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjXFjZc7bb)

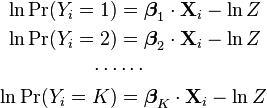
那么事件几率的概念是指的发生的概率与不发生的概率的比值，取对数则称为对数几率(log odds)。  
the logit

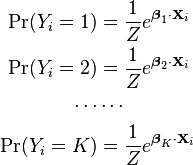
[caffe <wbr>study(3) <wbr>关于激活函数以及loss <wbr>function](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjY1sA5Ada)  
  
对于对元的分布，采用base-line logit，如下所示。(从某种意义上二元的logit是多元的一个特例)  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjY1uQkg98)  
  
通过概率和为1这样一个约束，可以得到Pr(Y=K)的表示，从而推导出K-1个分布的公式，如下所示：  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjY1LZGv97)

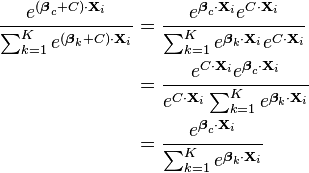
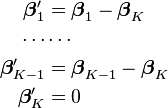
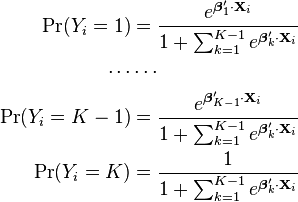
所有样本的概率和就是最后使用到的MULTINOMIAL\_LOGISTIC\_LOSS  
  
  
**3. SOFTMAX\_LOSS**  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjYdA0Be74)  
  
  
下面看一下基本的softmax的推导过程  
  
 在概率论中，一个归一化常数的的作用让所有的密度函数满足和为1这样一个条件

[caffe <wbr>study(3) <wbr>关于激活函数以及loss <wbr>function](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjYdGjP4ea)

此处采用log的形式描述，其中Z为归一化的常数

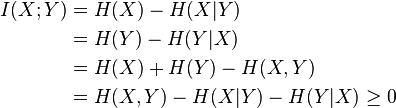
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjYdHHdY46)

[caffe <wbr>study(3) <wbr>关于激活函数以及loss <wbr>function](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjYdhTr775)  
  
得到  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjYdIIGy2e)  
根据和为1这样的约束可以得到

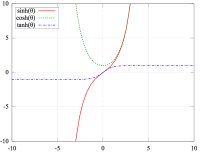
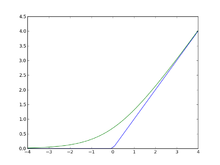
[caffe <wbr>study(3) <wbr>关于激活函数以及loss <wbr>function](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjYdttdW7c)  
  
softmax 则采用了连续函数来进行函数的逼近，最后采用概率的形式进行输出，这样就弱化了EUCLIDEAN\_LOSS中带来的问题。  
  
[caffe <wbr>study(3) <wbr>关于激活函数以及loss <wbr>function](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjZ4QU7peb)  
  
  
softmax中，各个节点的输出是一个归一化后的概率值，这个值随着每次迭代是动态变化的。  
  
**4. softmax和multinomial logistic的统一性**  
  
最后得到softmax的形式，在所有的beta参数中，只有K-1个是独立的，对于所有添加一个常量C不改变softmax的值，证明如下：  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjYevs5r69)  
  
如果这个把最后一个第K个样本的beta设置为常数，得到如下的变换  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjYewKvI06)  
  
  
  
带入原来的公式，可以得到：  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjYeGitQd2)  
  
这个输出就是multinomial logistic.  
  
**5. INFOGAIN\_LOSS**  
  
信息熵是对于信息的一种量化，是对于某种系统信息的一种量度，熵就是系统的平均信息量，也可以理解为某种信息出现的概率。

[caffe <wbr>study(3) <wbr>关于激活函数以及loss <wbr>function](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjYepIqped)

信息增益是在加入一个条件后，所得到的熵和原来的熵的差，具体形式如下：

[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjYeLqeN25)  
  
  
**6. SIGMOID\_CROSS\_ENTROPY\_LOSS**  
  
[caffe <wbr>study(3) <wbr>关于激活函数以及loss <wbr>function](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjZqtFiP8b)  
  
  
  
**7. HINGE\_LOSS**

[caffe <wbr>study(3) <wbr>关于激活函数以及loss <wbr>function](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjYex6BVdf)

求和后便成了Hinge loss  
     
这个loss就是SVM用到的loss。Hinge loss就是0-1 loss的改良版，这个改良主要在两个方面，一个是在t.y在【0 1】之间不再是采用hard的方式，而是一个soft的方式。另外一个就是在【-inf，0】之间不再采用固定的1来定义能量的损失，而是采用一个线性函数对于错误分类的情况进行惩罚。  
  
  
**下面是两个常用的激活函数**  
  
  
**8. TANH**  
  
  
[caffe <wbr>study(3) <wbr>关于激活函数以及loss <wbr>function](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjYfAMB349)  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjYfpUJSe4)  
  
  
**9. ReLU**  
  
relu是rectified linear unit的简写  
  
[caffe <wbr>study(3) <wbr>关于激活函数以及loss <wbr>function](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjYfsitb29)  
  
对于ReLU函数的近soft-plus函数  
  
[caffe <wbr>study(3) <wbr>关于激活函数以及loss <wbr>function](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjYfNn3v5d)  
  
他们的函数曲线如下图所示  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v3um&url=http://album.sina.com.cn/pic/004j58Jbzy6NjYfLRGu56)  
  
和其他的函数比较起来，ReLU函数更符合生物学的激活模型。

a.ReLU是单方向的  
b.在激活方面，这个函数强制阈值下的值为0，因此可以不使用l1正则就可以达到稀疏的目的。  
c.可以有效的缓解vanishing gradient problem，原来的vanishing gradient problem主要因为在backward过程中gradient随着深度的增加不断的连乘，如果每部分都小于1的话，随着层数的增加这个值会越来越小，从而导致vanishing gradient的问题。而ReLU函数由于没有上限，因此其导数在很大层度上缓解了这个问题。而sigmod以及tanh函数都没有这样的一个性质。  
d.计算速度快。  
  
因此ReLU是现在DNN模型中比较常用的激活函数。

## caffe study(4) - 优化算法基本原理

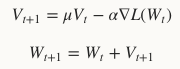
http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif (2014-11-07 15:21:45)

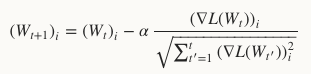
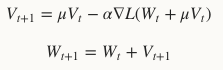
[[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)转载▼](javascript:;)

|  |  |
| --- | --- |
| 标签： [股票](http://search.sina.com.cn/?c=blog&q=%B9%C9%C6%B1&by=tag) |  |

机器学习就两个问题，一个是model的定义，一个就是如何优化，model是战略，优化则是执行。  
  
在Caffe现在的版本中，主要使用了三种优化算法SGD、AdaGrad、以及NAG，这三种算法快速、高效，缺点就是极容易收敛到局部极值点或者不容易收敛。  
  
a. 就个人理解来说，对于大数据的处理，算法的高效率要稍微优于精度。对于大规模的非线性优化来说，通过不断的调参可以有效的降低局部收敛或者不收敛的状况，从某种意义上说机器学习的本质就是调的一手好参：），是经验活也是体力活。  
b. 至于哪种算法好，可能要针对问题做选择。在几个月前，余凯老师在出门问问组织的一次学术活动中提到，目前工业界最为常用的算法就是mini-batch SGD，性价比最高，其他的一些算法更多的是学术上的目的。  
  
1. 关于stochastic approximation  
  
一般性的优化目标函数都由两部分组成，一个是loss term，一个是regularisation term，如下所示，前者就是根据model的形式定义的energy function，后者则对于参数的解空间进行了限制。

[caffe <wbr>study(4) <wbr>- <wbr>优化算法基本原理](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v41r&url=http://album.sina.com.cn/pic/004j58Jbzy6Nqm3r1Fj9d)  
  
  
其中D为数据集，最后的目标函数就是在数据集上的平均。当数据集小的时候，计算量不算很大，但是当数据量非常大的时候，这种策略就会使得计算效率下降，由此引出了stochastic approximation的策略，也就是使用随机子集对于原函数近似的思路。  
[caffe <wbr>study(4) <wbr>- <wbr>优化算法基本原理](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v41r&url=http://album.sina.com.cn/pic/004j58Jbzy6Nqm2Lgoo10)  
  
其中N<<D，这种策略成功的一个关键点就是大量的数据，主要原因有：  
  
a.使用子集拟合出的函数只代表子集的特征，因此单一的子集不可避免会带来误差。  
b.随着数据的增加，随机子集的数量越来越多，从概率上讲，其中接近真实函数的分布的的子集会占多数，因此总体上而言，拟合的函数会越来越接近全部数据拟合的函数。  
  
2. Stochastic gradient decent（SGD）算法  
  
优化算法就是看迭代公式，关键的就是步长和方向。

caffe中SGD权重迭代公式如下，W\_{t}为t时刻的参数，V\_{t}为t时刻的增量，a为learning rate，u为前一次更新的权重  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v41r&url=http://album.sina.com.cn/pic/004j58Jbzy6Nqm2Yx9m48)  
  
通过调节a和u可以有效的改变迭代速度以及迭代的方向。可以看出W\_{i}的更新是和上次增量以及梯度线性相关的的。  
  
在Caffe的文档中给出了一个基本的调参规则  
  
a. 初始的learning rate建议设置为0.01。  
b. 在迭代过程中当loss到达了一个明显的“plateau”的时候，则以10倍的速度drop learning rate，相当于稍微减少步长，  
c. u是一个调解参数，一般设置为0.9，这个参数起到一个平滑的作用，可以让SGD算法更加稳定和快速  
d. u和a是一个反相关的，增加u，则降低a，反之亦然。  
  
3. AdaGrad算法  
  
迭代规则为

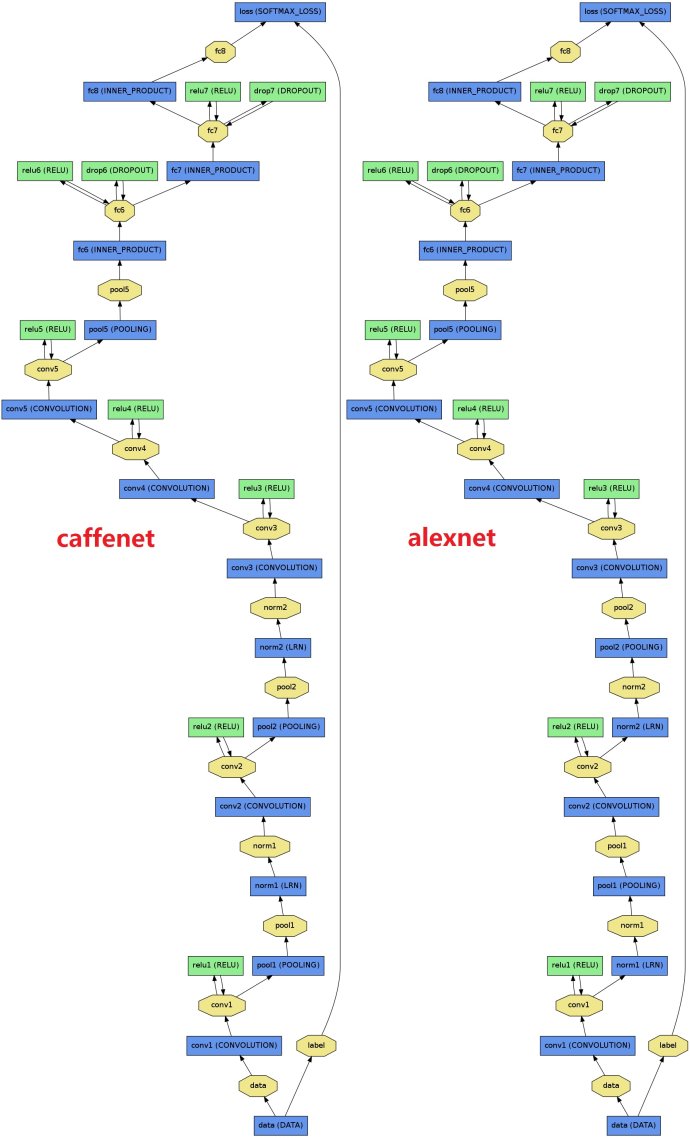
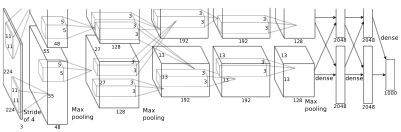
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v41r&url=http://album.sina.com.cn/pic/004j58Jbzy6Nqm2Tl8k00)  
  
其中i是指的W中的每一个维度。这个迭代规则的意思是采用前面的所有时间的梯度生成一个归一化的因子，对当前的梯度进行一次尺度变换。这个尺度的含义就是如果前面函数梯度大，那么当前增量的加速度就小一点，反之则大一点；整个趋势还是随着迭代的次数增加，增量放缓。  
  
4. Nesterov‘s accelerated Gradient （NAG）  
  
这个算法严格的说来是凸优化中的算法，具有O(1/t^2)的收敛率，收敛速度比较快。因为DNN是一个non-convex的model，所以NAG方法并不能达到这个收敛速度。caffe文档中指出，这个方法对于某些deeplearning 的 architecture是非常有效的。  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v41r&url=http://album.sina.com.cn/pic/004j58Jbzy6Nqm2yu6H67)  
  
这个公式和SGD方法很类似，不同之处在于其梯度项，不再是简单的当前位置的梯度，而是当前位置加上增量之后的新位置处的梯度。  
  
5. 一点思考  
  
5年前随着Sparse representation的热潮的兴起，做很多老的凸优化的算法在新的计算机视觉任务中显示出了巨大的能力，比如APG、NAG、ADMM等，人们发现了有趣的现象，效率高优化算法往往都是老算法，越老效率越高，主要原因是早些时候，计算能力非常有限，早期的数学家只好真正从算法中挤水分；随着时代的发展，计算能力强了，算法效率差不多就行了；现在到了大数据时代，效率终于又成为了瓶颈，所以老算法又焕发了第二春。另外一些二阶段方法 Newton，LBFGS等因为数据存储量以及计算量大，在大规模的数据中可能不如最简单的SGD类方法好用，当然选用什么算法，要视计算资源以及计算任务而定。

## caffe study(5) - AlexNet 之结构篇

http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif (2014-11-10 18:00:13)

[[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)转载▼](javascript:;)

|  |  |
| --- | --- |
| 标签： [股票](http://search.sina.com.cn/?c=blog&q=%B9%C9%C6%B1&by=tag) |  |

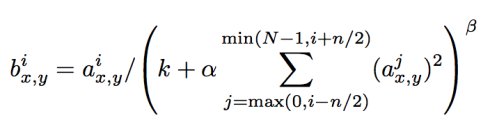
在2012年的时候，Geoffrey和他学生Alex为了回应质疑者，在ImageNet的竞赛中出手了，刷新了image classification的记录，一举奠定了deep learning 在计算机视觉中的地位。后边的故事大家都知道了，deep learning一统天下，所向披靡。  
  
在这次竞赛中Alex所用的结构被称为作为AlexNet。本部分首先介AlexNet的基本的architecture，在后边的学习中将分析其算法的基本原理以及参数细节。  
  
  
针对2012年的这组数据集caffe也定义了自己的结构，被称为caffeNet，文档中说在迭代30多w次的的情况下精度大概提高了0.2个百分点。下面是两个Net的结构图，它们的区别在于norm1，pool1，以及norm2，pool2互换了顺序。  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v47i&url=http://album.sina.com.cn/pic/004j58Jbzy6NvjDo5UI60)  
  
  
下面来看一下AlexNet的结构。在Alex的论文中，其基本结构为  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v47i&url=http://album.sina.com.cn/pic/004j58Jbzy6Nvk3SG142a)  
  
  
1. 基本结构  
  
a. 共有8层，其中前5层convolutional，后边3层full-connected ，最后的一个full-connected层的output是具有1000个输出的softmax，最后的优化目标是最大化平均的multinomial logistic regression  
  
b. 在第一层conv1和conv2之后直接跟的是Response-nomalization layer，也就是norm1，norm2层。  
  
c. 在每一个conv层以及full-connected层后紧跟的操作是ReLU操作。  
  
d. Max pooling操作是紧跟在第一个norm1，norm2，以及第5个conv层，也就是conv5  
  
e.Dropout操作是在最后两个full-connected层。  
  
2. 操作流程  
  
a. 在第一个conv层（conv1）中，AlexNet采用了96个11\*11\*3的kernel在stride为4的情况下对于224\*224\*3的图像进行了滤波。直白点就是采用了11\*11的卷积模板在三个通道上，间隔为4个像素的采样频率上对于图像进行了卷积操作。4个像素是receptive field中心在kernel map上的距离，这是个经验数据。  
  
最初的输入神经元的个数为224\*224\*3 = 150528个。对于每一个map来说，间隔为4，因此224/4 = 56，然后减去边缘的一个为55个，也就是本层的map大小为55\*55，然后神经元数目为55\*55\*96 = 290400（原文中是253440，这个不太理解，有知道的请告知）  
  
得到基本的conv数据后，则进行一次ReLU（relu1）以及Norm（norm1）的变换后，然后进行pooling（pool1），作为输出传递到下一层。  
  
本层map数目为96.  
  
b.第二个conv层(conv2)是第一个conv层(conv1)经过norm(norm1)以及pool(pool1)后，然后再apply 256个5\*5的卷积模板卷积后的结果。  
  
pool1后，map的size减半int（55/2）= 27，得到的本层的神经元数目为27\*27\*256 = 186642个。  
  
本层map数目为256个。  
  
c. 第三个conv层(conv3)的生成过程和第二层类似，不同的是这一层是apply 384个3\*3的卷积模板得到的。  
  
pool2后，map的size减半int(27/2) = 13，得到本层神经元数目为13\*13\*384 = 64896。  
  
本层map数目为384个。  
  
d. 第四个conv层(conv4)是第三个conv层(conv3)进行一次ReLU(relu3)后，然后直接apply 384个3\*3的卷积模板得到的。  
  
本层神经元的数目为13\*13\*384 = 64896。  
  
本层map数目为384个，size还是13\*13.  
  
e. 第五个conv层(conv5)与第四个生成类似，只是对上层进行了一个ReLU(relu4)后生成的，不同之处在于此处apply的卷积模板为256个3\*3的。  
  
本层神经元的数目为13\*13\*256 = 43264。  
  
本层map数目为256个，size还是13\*13。  
  
f. 第一个full-connected层(fc6)是上一个conv层(conv5)进行pooling(pool5)后的全连接。  
  
pool5后，map的size减半int(13/2) = 6，上层基本连接为6\*6\*256，然后全连接后到4096个节点，本层最终节点为4096个。  
  
g.第二个full-connected层(fc7)是上一个full-connected(fc6)进行ReLU(relu6)后，然后进行dropout(drop6)后再进行全连接的结果  
  
本层节点数目为4096个。  
  
h.最后一个full-connetcted层(fc8)则是上一个full-connected层(fc7)再次进行ReLU(relu7)以及dropout(drop7)后再进行全连接的结果。最后输出为融合label的softmax loss。  
  
本层节点数目为1000个，对应着1000类对象。  
  
在下面将对于AlexNet的原理进行以及算法的执行细节进行分析。

## caffe study(5) - AlexNet 之算法篇

http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif (2014-11-27 17:30:20)

[[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)转载▼](javascript:;)

|  |  |
| --- | --- |
| 标签： [股票](http://search.sina.com.cn/?c=blog&q=%B9%C9%C6%B1&by=tag) |  |

在机器学习中，我们通常要考虑的一个问题是如何的“以偏概全”，也就是以有限的样本或者结构去尽可能的逼近全局的分布。这就要在样本以及结构模型上下一些工夫。  
  
在一般的训练任务中，考虑的关键问题之一就是数据分布是否合理：首先是数据集的覆盖度，也就是数据集是否能够覆盖样本空间；其次还要尽可能的保证具有和真实数据一样的分布（注意数据分布是未知的，你只能根据一些先验来近似），这样的数据才是有效的。当然这些方式只是增大了得到正确解的概率，而并不能保证一定可以得到正确解。当你不知道你所取的训练集合是否和真实分布一致的时候，那么就要多取几次，每一个数据集都算算，对于分类器也是这样，单个分类器往往不能精确描述一个分界面，那么我们就组合一下，每个都算算。从方法论上讲，对于事物观察到的往往是局部，因此会犯以偏概全的错误，如果能够将所得到的“偏” ensambling 一下，那么就生成了相对的“全”，从而可以更大的概率逼近总体分布。这种思想在好多方面都体现出来，如交叉验证，经典的RANSAC，Random Tree（forest），Adaboost 等方法。  
  
下面将从数据和模型两个方面来学习一下AlexNet中的一些技巧，主要参考的是Alex 2012 年的 NIPS论文ImageNet classification with deep convolutional neural networks.  
   
1. 数据的处理：  
  
到目前为止，还没有人看到数据集的大小对deeplearning算法理论上限造成的影响，也就是说数据集合还没有达到临界点，所以增加数据集只有好处，没有坏处。  
  
在Alex的论文中，采用了两个方法对于图像进行了增强。  
  
a. 增大训练样本：通过对于图像的变换实现了对于数据集合的enlarge。首先对于输入的图像（size 256\*256）随机提取224\*224的图像集合，并对他们做一个horizontal reflections。变换后图像和原图像相差了32个像素，因此主体部分应该都包含在训练集合中，相当于在位置这个维度上丰富了训练数据。对horizontal reflections来说，相当于相机在主轴方向做了镜像，丰富了反方向的图像。数据集合增大了2048倍，直接结果就是降低了overfitting同时降低了网络结构设计的复杂层度。  
  
在测试阶段，取每一个测试样本四个角以及中间区域，一共5个patch然后再镜像后得到10个样本输入到网络中，最后将10个softmax输出平均后作为最后的输出。  
  
b.使用PCA对于训练数据进行增强：对于每一个RGB图像进行一个PCA的变换，完成去噪功能，同时为了保证图像的多样性，在eigenvalue上加了一个随机的尺度因子，每一轮重新生成一个尺度因子，这样保证了同一副图像中在显著特征上有一定范围的变换，降低了overfitting的概率。  
  
以上的策略是不是真的有必要，这个还是要打一个问号，因为对于a部分来说，样本少，可以在结构设计上下下功夫，可能达到相同的效果。对于b来说，deeplearning还需要对于图像加入增强处理吗？如果这样的话，自然也可以用一些传统人工特征先来一遍，再deeplearning了。我想关键的原因是deeplearning还没有真正的被证明的规则，所以你用什么策略都有点道理，但是谁敢保证不是“以偏概全”呢？  
  
2. 模型结构：  
  
在模型的设计上，AlexNet做了一个Local Response Normalization的处理，同时在节点的选择上采用了一个dropout策略。  
  
a. Local Response Normalization.  
  
公式如下，其中a是每一个神经元的激活，n是在同一个位置上临近的kernel map的数目，N是可kernel的总数目，k，alpha，beta都是预设的一些hyper-parameters，其中k=2，n=5，alpha = 1\*e-4，beta = 0.75。  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=eb3aea990102v5px&url=http://album.sina.com.cn/pic/004j58Jbzy6NWoEZO5sb2)  
  
从这个公式中可以看出，原来的激活a被加一个归一化权重（分母部分）生成了新的激活b，相当于在同一个位置（x，y），不同的map上的激活进行了平滑，但是至于为什么k，alpha，beta这样来设置，没有想太清楚。  
  
这个平滑大概可以将识别率提高1-2个百分点。  
  
b. Dropout策略  
  
使用多个model来共同进行预测是一个降低test errors的基本方法，但是单独的训练多个model组合会导致整个的训练成本增加，毕竟训练一个单一的网络需要很长的时间，即便计算资源足够，在不影响精度的情况下降低整个运算时间还是我们追求的目标。  
  
由此Hinton提出了dropout策略，这个策略很简单，对于每一个隐层的output，以50%的概率将他们设置为0，不再对于forward或者backward的过程起任何作用。对于每一个input来说，使用的不同的网络结构，但是权重是共享的。这样求得的参数能够适应不同的情况下的网络结构，也就是提高了系统的泛化能力。  
  
在AlexNet中最后的两个full-connected层中使用了这个策略。  
  
3. 优化算法的参数  
  
论文中使用SGD算法，基本参数设置在前面优化算法的总结中已经提到了。这里要说几个个人体会。  
  
 a. 原文中输入的batch数目是256，应该Alex经过调节后的结果，我实际用到的机器性能比较低，内存8G，显存4G，所以不得不就将batch数目往下调到64，以免产生out of memory的错误。这样就需要调节其他的参数来保证数据的收敛。原因是batch比较小，导致本文开篇提到的样本覆盖面过低，产生了非常多的局部极小点，在步长和方向的共同作用下，导致数据产生了震荡，导致了不收敛。  
  
b.在这种情况下，把learning rate调节到了0.02，相当于加大了步长，这样可以在一定程度上避免震荡，可以越过局部极小点往比较大的极值点行走。  
  
c. 对于每一层的bias从1设置为了0.1，在一定程度上限制了激活的大小，这样就限制了某一过大的误差的影响，这样可以避免迭代方向出现过大的变化。  
  
d. 经过b c后，系统终于收敛了，但带来的不良后果就是整个收敛速度变慢，因此还需要增加最大迭代次数，经过测试迭代次数成了从45w修改成了70w。  
  
e. 在整个运行过程中，出现了几次平稳点，20w以及40w左右的时候，因此迭代的learning rate应该随着迭代的接近平稳点的时候有意的减小一些，目前是以每10w次减小为1/10，调参数用了5天，最后运行时间为15天。  
  
f. 关于调参策略，上面只是按照一些简单的理解设置的，如果没有一个合理的解释，调参就变成了一个很low的工作。还好发现了好几篇关于调参的论文，主要是优化算法理论方面的，学习完再回来测试一下。