射您的反馈!

返回 了解详情

我们已记录您对此推广内容的反馈,以便改 善您今后的浏览体验。





访问: 13652次 积分: 267 等级: 8L00 2 排名: 千里之外

原创: 12篇 转载: 1篇 译文: 0篇 评论: 9条

文章搜索

文章分类

机器学习 (8)

kaggle (2)

编程语言 (2)

tools (2)

文章存档

2016年07月 (1)

2016年06月

2016年05月 (4)

2016年04月 (6)

2016年03月 (1)

阅读排行

ubuntu 16.04 无GPU版caffe... (3861)机器学习之神经网络bp算法推... (2862) kaggle机器学习教程(Python... (2424)初识Kaggle: 手写体数字识别 (1506)机器学习之EM算法解析 (520)将latex公式转换成图片 (498)机器学习之高斯混合模型 (449)机器学习数学基础(1)

捋一捋AdaBoost (1):算法...

登录 | 注册

: 目录视图 ₩ 摘要视图 RSS 订阅

ES6、虚拟现实、物联网(评论送书)

异步赠书:9月重磅新书升级,本本经典 SDCC 2017之区块链技术实战线上峰会

机器学习之神经网络bp算法推导

标签: 机器学习 神经网络

2016-05-05 22:30

2868人阅读

≔ 分类:

机器学习(7) -

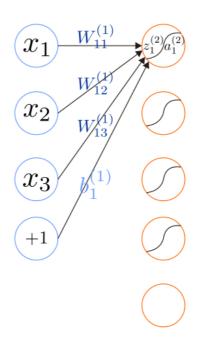
■版权声明:本文为博主原创文章,未经博主允许不得转载。

目录(?) [+]

这是一篇学习UFLDL反向传导算法的笔记,按自己的思路捋了一遍,有不对的地方请大家指点。

首先说明一下神经网络的符号:

- $1. n_l$ 表示神经网络的层数。
- $2. s_l$ 表示第 l 层神经元个数 , 不包含偏置单元。
- 3. $z_i^{(l)}$ 表示第 l 层第 i 个神经元的输入; $a_i^{(l)}$ 表示第 l 层第 i 个神经元的输出。
- 4. $W_{ij}^{\,(l)}$ 表示第 l 层第 j 个神经元连接到第 l+1 层第 i 个神经元的权重,因此权值矩阵 W 的维 数为 s_{l+1} x s_l



第二层各神经元的计算方法如下:

(258)

返回 了解详情

我们已记录您对此推广内容的反馈,以便改 **基您今后的浏览体验**



* SDCC 2017之区块链技术实战线上峰会 * 快速集成一个视频直播功能

最新评论

机器学习之神经网络bp算法推导

hjl240 : 上文确实有个小问题, 求和符合的 上限应该改为S_nl-1

机器学习之神经网络bp算法推导

Amandayyt:绝对写的超级棒,终于弄明

ubuntu 16.04 无GPU版caffe安装简记

Imkeep:安装anaconda的话,还用执行楼 主贴的第一个部分的代码吗?我看里面的安 装Python之类的ana...

机器学习之神经网络bp算法推导

Emma_8:多谢,有些地方跟我想的一 样,比如前面的参数含义。其他不懂的地 方,也得到了更加深刻的理解,多谢博主

机器学习之神经网络bp算法推导

Starry5cm : 总之 谢谢博主的原创文章

机器学习之神经网络bp算法推导

Starry5cm : 定位然后计算倒数第二层即第 nl-1 层第 i 个神经元的残差: 倒数第二个等 式第二个求和符合的上限S...

机器学习之神经网络bp算法推导

Starry5cm : 然后计算倒数第二层即第 nl-1 层第 i 个神经元的残差:倒数第二个等式没

ubuntu 16.04 无GPU版caffe安装简记

明明3x:感谢分享,解决了anaconda下运 行make runtest "libhdf5.so.10" 的问题

ubuntu 16.04 无GPU版caffe安装简记

Saerdna_pp : 感谢分享,解决了anaconda 下运行make runtest "libhdf5.so.10" 的 问题..

机器学习之神经网络bp算法推导 - bigPo's Blog - CSDN博客

$$\begin{split} a_1^{(2)} &= f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \\ a_2^{(2)} &= f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \\ a_3^{(2)} &= f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \\ a_4^{(2)} &= f(W_{41}^{(1)}x_1 + W_{42}^{(1)}x_2 + W_{43}^{(1)}x_3 + b_4^{(1)}) \end{split}$$

我们可以将其向量化表示:

$$oldsymbol{z}^{(2)} = W^{(1)} oldsymbol{x} + oldsymbol{b}^{(1)} \ oldsymbol{a}^{(2)} = f(oldsymbol{z}^{(2)})$$

这里的矩阵W的具体形式为:

$$W_{4 imes 3} = egin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \ W_{31}^{(1)} & W_{32}^{(1)} & W_{33}^{(1)} \ W_{41}^{(1)} & W_{42}^{(1)} & W_{43}^{(1)} \end{bmatrix}$$

第2层的神经元个数为4,第1层神经元的个数为3,因此为 4×3 维的矩阵。

代价函数

对于单个样本我们将神经网络的代价函数定义为:

$$J(W,b;x,y) = rac{1}{2} \left\lVert h_{W,b}(x) - y
ight
Vert^2$$

对所有 K 个样本,神经网络的总的代价函数(这也是批量的由来)为:

$$egin{aligned} J(W,b) &= \left[rac{1}{K}\sum_{k=1}^{K}J(W,b;x^{(k)},y^{(k)})
ight] + rac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}
ight)^2 \ &= \left[rac{1}{K}\sum_{k=1}^{K}\left(rac{1}{2}\left\|h_{W,b}(x^{(k)})-y^{(k)}
ight\|^2
ight)
ight] + rac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}
ight)^2 \end{aligned}$$

使用批量梯度下降算法寻求神经网络的最优参数

我们使用批量梯度下降算法寻求神经网络的最优参数 $W^{(l)},b^l$ 。

我们先来看对于 第 l+1 层第 i 个神经元来说,第 l 层第 j 个神经元的权值可按如下方式迭代更 新:

$$egin{aligned} W_{ij}^{(l)} &= W_{ij}^{(l)} - lpha \, rac{\partial}{\partial W_{ij}^{(l)}} \, J(W,b) \ &= W_{ij}^{(l)} - lpha \left[\left(rac{1}{K} \sum_{k=1}^K rac{\partial}{\partial W_{ij}^{(l)}} \, J(W,b;x^{(k)},y^{(k)})
ight) + \lambda W_{ij}^{(l)}
ight] \end{aligned}$$

类似的,对于 第l+1 层第i 个神经元来说,第l 层的偏置单元的权值可按如下方式迭代更新:

返回 了解详情

我们已记录您对此推广内容的反馈,以便改善您今后的浏览体验。





$$b_i^{(l)} = b_i^{(l)} - lpha \, rac{\partial}{\partial b_i^{(l)}} \, J(W,b)$$

$$=b_i^{(l)}-\alpha\left[\frac{1}{K}\sum_{k=1}^K\frac{\partial}{\partial b_i^{(l)}}J(W,b;x^{(k)},y^{(k)})\right]$$

我们现在的目的是求出以下两个式子就可以对参数进行迭代了:

$$egin{aligned} rac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x^{(k)},y^{(k)}) \ rac{\partial}{\partial b_i^{(l)}} J(W,b;x^{(k)},y^{(k)}) \end{aligned}$$

又我们知道第 l+1 层第 i 个神经元的输入 $z_i^{(l+1)}$ 可以由以下式子计算:

$$z_i^{(l+1)} = \sum_{j=1}^{s_l} W_{ij}^{(l)} a_j^{(l)} + b_i^{(l)}$$

再进一步的对上面的式子进行变形:

$$\begin{split} & \frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x^{(k)},y^{(k)}) \\ & = \frac{\partial J(W,b;x^{(k)},y^{(k)})}{\partial z_i^{(l+1)}} \cdot \frac{\partial z_i^{(l+1)}}{\partial W_{ij}^{(l)}} \\ & = \frac{\partial J(W,b;x^{(k)},y^{(k)})}{\partial z_i^{(l+1)}} \cdot a_j^{(l)} \end{split}$$

同样的 , 对于 $b_i^{(l)}$ 的偏导数 :

$$\begin{split} & \frac{\partial}{\partial b_{i}^{(l)}} J(W, b; x^{(k)}, y^{(k)}) \\ & = \frac{\partial J(W, b; x^{(k)}, y^{(k)})}{\partial z_{i}^{(l+1)}} \cdot \frac{\partial z_{i}^{(l+1)}}{\partial b_{i}^{(l)}} \\ & = \frac{\partial J(W, b; x^{(k)}, y^{(k)})}{\partial z_{i}^{(l+1)}} \end{split}$$

残差的定义

接下来我们定义:

$$\delta_i^{(l)} = rac{\partial}{\partial z^{(l)}} J(W,b;x^{(k)},y^{(k)})$$

为第k个样本在第l层第i个神经元上产生的残差。再次回顾我们的参数更新公式: 对于 $W_{ij}^{(l)}$ 我们有:

了解详情

我们已记录您对此推广内容的反馈,以便改 善您今后的浏览体验。





$$\begin{split} W_{ij}^{(l)} &= W_{ij}^{(l)} - \alpha \, \frac{\partial}{\partial W_{ij}^{(l)}} \, J(W,b) \\ &= W_{ij}^{(l)} - \alpha \, \left[\left(\frac{1}{K} \sum_{k=1}^K \frac{\partial}{\partial W_{ij}^{(l)}} \, J(W,b;x^{(k)},y^{(k)}) \right) + \lambda W_{ij}^{(l)} \right] \\ &= W_{ij}^{(l)} - \alpha \, \left[\left(\frac{1}{K} \sum_{k=1}^K \frac{\partial J(W,b;x^{(k)},y^{(k)})}{\partial z_i^{(l+1)}} \cdot a_j^{(l)} \right) + \lambda W_{ij}^{(l)} \right] \\ &= W_{ij}^{(l)} - \alpha \, \left[\left(\frac{1}{K} \sum_{k=1}^K \delta_i^{(l+1)} \cdot a_j^{(l)} \right) + \lambda W_{ij}^{(l)} \right] \end{split}$$

类似的 , 对于 $b_i^{(l)}$ 我们有 :

$$\begin{split} b_i^{(l)} &= b_i^{(l)} - \alpha \, \frac{\partial}{\partial b_i^{(l)}} \, J(W,b) \\ &= b_i^{(l)} - \alpha \, \frac{1}{K} \sum_{k=1}^K \frac{\partial}{\partial b_i^{(l)}} \, J(W,b;x^{(k)},y^{(k)}) \\ &= b_i^{(l)} - \alpha \, \frac{1}{K} \sum_{k=1}^K \frac{\partial J(W,b;x^{(k)},y^{(k)})}{\partial z_i^{(l+1)}} \\ &= b_i^{(l)} - \alpha \, \frac{1}{K} \sum_{k=1}^K \delta_i^{(l+1)} \end{split}$$

现在的核心问题只剩下一个了,这个残差该如何求?

我们先计算最后一层第i个神经元上的残差,这里为了简单起见,不再指定为第k个样本。

$$\begin{split} \delta_i^{(n_l)} &= \frac{\partial}{\partial z_i^{(n_l)}} J(W, b; x, y) \\ &= \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \left\| h_{W,b}(x) - y \right\|^2 \\ &= \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \sum_{j=1}^{s_{n_l}} (y_j - a_j^{(n_l)})^2 \\ &= \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \sum_{j=1}^{s_{n_l}} (y_j - f(z_j^{(n_l)}))^2 \\ &= -(y_i - f(z_i^{(n_l)})) f'(z_i^{(n_l)}) \end{split}$$

然后计算倒数第二层即第 n_l-1 层第 i 个神经元的残差:

感谢您的反馈! 返回 了解详情

我们已记录您对此推广内容的反馈,以便改善您今后的浏览体验。

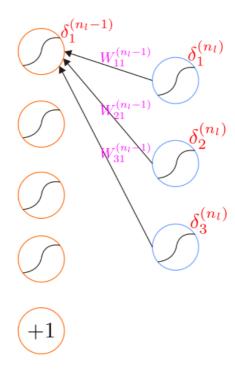




机器字习之种经网络DD界法推导 - DIGPOS BIOG - CSDN 图 4

$$\begin{split} \delta_i^{(nl-1)} &= \frac{\partial}{\partial z_i^{(nl-1)}} \, J(W,b;x,y) \\ &= \frac{\partial}{\partial z_i^{(nl-1)}} \, \frac{1}{2} \sum_{j=1}^{s_{nl}} (y_j - a_j^{(nl)})^2 \\ &= \frac{1}{2} \sum_{j=1}^{s_{nl}} \frac{\partial}{\partial z_i^{(nl-1)}} \, (y_j - f(z_j^{(nl)}))^2 \\ &= \sum_{j=1}^{s_{nl}} -(y_j - f(z_j^{(nl)})) \, \frac{\partial}{\partial z_i^{(nl-1)}} \, f(z_j^{(nl)}) \\ &= \sum_{j=1}^{s_{nl}} -(y_j - f(z_j^{(nl)})) f'(z_j^{(nl)}) \, \frac{\partial z_j^{(nl)}}{\partial z_i^{(nl-1)}} \\ &= \sum_{j=1}^{s_{nl}} \delta_j^{(nl)} \, \frac{\partial}{\partial z_i^{(nl-1)}} \sum_{q=1}^{s_{nl}} W_{jq}^{(nl-1)} f(z_q^{(nl-1)}) \\ &= \sum_{j=1}^{s_{nl}} W_{ji}^{(nl-1)} \delta_j^{(nl)} f'(z_i^{(nl-1)}) \end{split}$$

下面是残差传播的示意图:



从这里可以看出紧挨着的两层神经元之间的残差是有关系的,这也是反向传播的由来。更一般的,可以将上述关系表述为:

$$\delta_i^{(l)} = \sum_{i=1}^{sl+1} W_{ji}^{(l)} \delta_j^{(l+1)} f'(z_i^{(l)})$$

再再次回顾我们的参数更新公式:

$$\begin{split} W_{ij}^{(l)} &= W_{ij}^{(l)} - \alpha \left[\left(\frac{1}{K} \sum_{k=1}^K \delta_i^{(l+1)} \cdot a_j^{(l)} \right) + \lambda W_{ij}^{(l)} \right] \\ b_i^{(l)} &= b_i^{(l)} - \alpha \frac{1}{K} \sum_{k=1}^K \delta_i^{(l+1)} \end{split}$$

我们需要先计算输出层神经元的残差,然后一级一级的计算前一层的神经元的残差,利用这些残差就可以更新神经网络参数了。

返回 了解详情

我们已记录您对此推广内容的反馈,以便改善您今后的浏览体验。





向量化表示

这里我们尝试将上述结果表示成向量或矩阵的形式,比如我们希望能一次性更新某一层神经元的权值和偏置,而不是一个一个的更新。

 $\delta_i^{(l+1)}$ 表示的是第 l+1 层第 i 个神经元的残差,那么整个第 l+1 层神经元的偏差是多少呢?

$$\delta_i^{(l+1)} = \sum_{i=1}^{sl+2} W_{ji}^{(l+1)} \delta_j^{(l+2)} f'(z_i^{(l+1)})$$

从而得到:

$$\delta^{(l+1)} = (W^{(l+1)})^T \delta^{(l+2)} ullet f'(z^{(l+1)})$$

注:这里的 ullet 是指点乘,即对应元素相乘, $\delta^{(l+1)}$ 是一个 $s_{l+1} imes 1$ 维的列向量。

 $a_j^{(l)}$ 表示第 l 层第 j 个神经元的输出,因此整个第 l 层的神经元的输出可用 $a^{(l)}$ 表示,是一个 $s_l \times 1$ 维的列向量。

因此对于矩阵 $W^{(l)}$ 来说,我们记:

$$abla_{W^{(l)}}J(W,b;x,y)=\delta^{(l+1)}(a^{(l)})^T$$

我们将 $\Delta W^{(l)}$ 初始化为 0 ,然后对所有 K 个样本将它们的 $\nabla_{W^{(l)}}J(W,b;x,y)$ 累加到 $\Delta W^{(l)}$ 中去:

$$\Delta W^{(l)} := \Delta W^{(l)} +
abla_{W^{(l)}} J(W,b;x,y)$$

然后更新一次 $W^{(l)}$:

$$W^{\left(l
ight)}=W^{\left(l
ight)}-lpha\left[\left(rac{1}{K}\Delta W^{\left(l
ight)}
ight)+\lambda W^{\left(l
ight)}
ight]$$

这里再强调一下:上式中的 $\Delta W^{(l)}$ 是所有 K 个样本的 $\delta^{(l+1)}(a^{(l)})^T$ **累加**和,如果希望做随机梯度下降了或者是mini-batch,这里就不用把所有样本的残差加起来了。

类似的,令:

$$abla_{b^{(l)}}J(W,b;x,y)=\delta^{(l+1)}$$

我们将 $\Delta b^{(l)}$ 初始化为 0 , 然后对所有 K 个样本将它们的 $\nabla_{b^{(l)}}J(W,b;x,y)$ 累加到 $\Delta b^{(l)}$ 中去

$$\Delta b^{(l)} := \Delta b^{(l)} + \nabla_{b^{(l)}} J(W,b;x,y)$$

于是有:

返回 了解详情

我们已记录您对此推广内容的反馈,以便改 善您今后的浏览体验。





$b^{(l)} = b^{(l)} - lpha \left\lceil rac{1}{K} \, \Delta b^{(l)} ight ceil$

同样的 , 上式中的 $\Delta b^{(l)}$ 是所有 K 个样本的 $\delta^{(l+1)}$ **累加**和。

小结

上面的推导过程尝试把所有的步骤都写出来了,个人感觉比UFLDL上的教程更为详尽,只要你耐心 看总能看得懂的。当然这篇文章有些细节并未作说明,比如惩罚因子的作用,为什么没有对偏置进 行规则化,激活函数的选择等,这些都可以在UFLDL中找到答案,对应的链接在下证 出。

- [1] 神经网络
- [2] 反向传导算法

顶 踩

- 上一篇 将latex公式转换成图片
- 下一篇 捋一捋AdaBoost (1):算法实现

相关文章推荐

- 从神经网络到BP算法 (纯理论推导)
- 大数据技术实战线上峰会--董西成
- 深度学习BP算法的推导(附加RNN,LSTM的推导...
- 30天系统掌握机器学习--唐宇迪
- 深度学习BP算法的推导(附加RNN,LSTM的推导...
- ORACLE数据库学习资料集锦
- BP算法与公式推导
- PHP从零开始实战篇

- BP神经网络算法推导
- 玩转微信小程序第一篇
- BP神经网络:误差反向传播公式的简单推导
- 深度学习案例分享——人脸检测
- BP神经网络推导过程详解
- BP神经网络公式推导及实现(MNIST)
- matlab编写的BP神经网络算法,适合对计算公式...
- 捋一捋AdaBoost (1): 算法实现





天然植物萃取 无激素专利





查看评论



- 引用"u011017860"的评论: -

[Ctrl + F] 定位

然后计算倒数第二层即第 nl-1 层第 i 个神经元的残差:

6楼 前天 17:03发表

返回 了解详情

我们已记录您对此推广内容的反馈,以便改 善您今后的浏览体验。

Baide音度



上文确实有个小问题,求和符合的上限应该改为S_nl-1

Amandayyt

绝对写的超级棒,终于弄明白了

倒数第二个...

5楼 2017-08-10 16:17发表

Emma__8

多谢,有些地方跟我想的一样,比如前面的参数含义。

其他不懂的地方,也得到了更加深刻的理解,多谢博主

Starry5cm

3楼 2017-02-17 2-..-~~~

2楼 2017-02-17 22:24发表

1楼 2017-02-17 22:18发表

4楼 2017-05-08 21:22发表

总之 谢谢博主的原创文章

Starry5cm

[Ctrl + F] 定位

然后计算倒数第二层即第 nl-1 层第 i 个神经元的残差: 倒数第二个等式第二个求和符合的上限S_nl-1

Starry5cm

然后计算倒数第二层即第 nl-1 层第 i 个神经元的残差:

倒数第二个等式没有加上b

您还没有登录,请[登录]或[注册]

*以上用户言论只代表其个人观点,不代表CSDN网站的观点或立场

公司简介 | 招贤纳士 | 广告服务 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 | 网站客服 杂志客服 微博客服 webmaster@csdn.net

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved

