

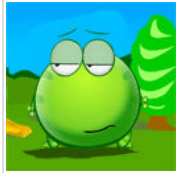
网络资源是无限的

目录视图

摘要视图

RSS 订阅

个人资料



fengbingchun



访问：2252590次

积分：25003

等级：BLOG > ?

排名：第202名

原创：341篇 转载：144篇

译文：0篇 评论：1434条

文章分类

Android (9)
ActiveX (18)
Bar Code (16)
Caffe (20)
C# (5)
Cimg (4)
Contour Detection (9)
CxlImage (6)
Code::Blocks (3)
Cloud Computing (1)
C/C++ (82)
CUDA (10)
CMake (3)
Design Patterns (25)
Database/Dataset (4)
Deep Learning (9)
Eclipse (3)
Emgu CV (1)
Eigen (1)
FFmpeg (1)
Feature Extraction (1)
FreeType (1)
Face (8)
GPU (3)
Git (3)
GCC (1)
GDAL (5)

CSDN学院招募微信小程序讲师啦

程序员简历优化指南！

【观点】移动原生App开发 PK HTML 5开发

云端应用征文大赛，秀

绝招，赢无人机！

tiny-cnn执行过程分析(MNIST)

2016-01-31 17:42

3565人阅读

评论(2)

收藏

举报

分类： Caffe (19) Deep Learning (8) Neural Network (12)

版权声明：本文为博主原创文章，未经博主允许不得转载。

在<http://blog.csdn.net/fengbingchun/article/details/50573841>中以MNIST为例对tiny-cnn的使用进行了介绍，下面对其执行过程进行分析：

支持两种损失函数：(1)、mean squared error(均方差)；(2)、cross entropy(交叉熵)。在MNIST中使用的是mean squared error，代码段：

```
[cpp]
01. // mean-squared-error loss function for regression
02. class mse {
03. public:
04.     static float_t f(float_t y, float_t t) {
05.         return (y - t) * (y - t) / 2;
06.     }
07.
08.     static float_t df(float_t y, float_t t) {
09.         return y - t;
10.     }
11. };
```

支持六种激活函数：(1)、tanh；(2)、sigmoid；(3)、softmax；(4)、rectifiedlinear(relu)；(5)、leaky relu；(6)、identity。MNIST中使用的是tanh，代码段：

```
[cpp]
01. class tan_h : public function {
02. public:
03.     float_t f(const vec_t& v, size_t i) const override {
04.         const float_t ep = std::exp(v[i]);
05.         const float_t em = std::exp(-v[i]);
06.         return (ep - em) / (ep + em);
07.     }
08.
09.     // fast approximation of tanh (improve 2-3% speed in LeNet-5)
10.     /**float_t f(float_t x) const {
11.         const float_t x2 = x * x;
12.         x *= 1.0 + x2 * (0.1653 + x2 * 0.0097);
13.         return x / std::sqrt(1.0 + x * x); // invsqrt(static_cast<float>(1.0 + x * x));
14.     }*/
15.
16.     float_t df(float_t y) const override { return 1.0 - y * y; }
17.     std::pair<float_t, float_t> scale() const override { return std::make_pair(-0.8, 0.8); }
```

设计CNN结构，用于MNIST，与LeNet-5结构相似，去除了F6层：

输入层Input：图像大小32*32，神经元数量32*32=1024，代码段：

- HTML (3)
- Image Recognition (8)
- Image Processing (18)
- Image Registration (13)
- ImageMagick (3)
- Java (5)
- Linux (20)
- Log (2)
- Makefile (2)
- Mathematical Knowledge (6)
- Multi-thread (4)
- Matlab (33)
- MFC (8)
- MinGW (3)
- Mac (1)
- Neural Network (13)
- OCR (9)
- Office (2)
- OpenCL (2)
- OpenSSL (7)
- OpenCV (86)
- OpenGL (2)
- OpenGL ES (3)
- OpenMP (3)
- Photoshop (1)
- Python (4)
- Qt (1)
- SIMD (14)
- Software Development (4)
- System architecture (2)
- Skia (1)
- SVN (1)
- Software Testing (4)
- Shell (2)
- Socket (3)
- Target Detection (2)
- Target Tracking (2)
- VC6 (6)
- VS2008 (16)
- VS2010 (4)
- VS2013 (3)
- vigra (2)
- VLC (5)
- VLFeat (1)
- wxWidgets (1)
- Watermark (4)
- Windows7 (6)
- Windows Core Programming (9)
- XML (2)

- Free Codes
- pubd
- freecode
- Peter's Functions
- CodeProject
- SourceCodeOnline
- Computer Vision Source Code
- Codesoso
- Digital Watermarking
- SourceForge
- HackChina
- oschina

```
[cpp]
01. const int width = header.num_cols + 2 * x_padding;
02. const int height = header.num_rows + 2 * y_padding;
03.
04. std::vector<uint8_t> image_vec(header.num_rows * header.num_cols);
05.
06. ifs.read((char*) &image_vec[0], header.num_rows * header.num_cols);
07.
08. dst.resize(width * height, scale_min);
09.
10. for (size_t y = 0; y < header.num_rows; y++)
11.     for (size_t x = 0; x < header.num_cols; x++)
12.         dst[width * (y + y_padding) + x + x_padding]
13.             = (image_vec[y * header.num_cols + x] / 255.0) * (scale_max - scale_min) + scale_min;
```

C1层：卷积窗大小5*5，输出特征图数量6，卷积窗种类6，输出特征图大小28*28，可训练参数5*5*6+6=156，神经元数量28*28*6=4704；

S2层：卷积窗大小2*2，输出下采样图数量6，卷积窗种类6，输出下采样图大小14*14，可训练参数1*6+6=12，神经元数量14*14*6=1176；

C3层：卷积窗大小5*5，输出特征图数量16，卷积窗种类16，输出特征图大小10*10，可训练参6*16*5*5+16=2416，神经元数量10*10*16=1600；

S4层：卷积窗大小2*2，输出下采样图数量16，卷积窗种类16，输出下采样图大小5*5，可训练参数1*16+16=32，神经元数量5*5*16=400；

C5层：卷积窗大小5*5，输出特征图数量120，卷积窗种类120，输出特征图大小1*1，可训练参数5*5*16*120+120=48120，神经元数量1*120=120；

输出层Output：输出特征图数量10，卷积窗种类10，输出特征图大小1*1，可训练参数120*10+10=1210，神经元数量1*10=10。

原有MNIST图像大小为28*28，此处为32*32，上下左右各填补2个像素，填补的像素取值为-1，其它像素取值范围为[-1,1]。

权值和阈值(偏置)初始化：权值采用均匀随机数产生，阈值均赋0。

C1层权值，初始化范围[sqrt(6.0/(25+150)), sqrt(6.0/(25+150))];

S2层权值，初始化范围[sqrt(6.0/(4+1)), -sqrt(6.0/(4+1))];

C3层权值，初始化范围[sqrt(6.0/(150+400)), -sqrt(6.0/(150+400))];

S4层权值，初始化范围[sqrt(6.0/(4+1)), -sqrt(6.0/(4+1))];

C5层权值，初始化范围[sqrt(6.0/(400+3000)), -sqrt(6.0/(400+3000))];

输出层权值，初始化范围[sqrt(6.0/(120+10)), -sqrt(6.0/(120+10))].

前向传播：

C1层代码段：

```
[cpp]
01. vec_t &a = a_[worker_index]; // w*x
02. vec_t &out = output_[worker_index]; // output
03. const vec_t &in = *(prev_out_padded_[worker_index]); // input
04.
05. std::fill(a.begin(), a.end(), (float_t)0.0);
06.
07. for_i(parallelize_, out_.depth_, [&](int o) {
08.     for (layer_size_t inc = 0; inc < in_.depth_; inc++) {
09.         if (!tbl_.is_connected(o, inc)) continue;
10.
11.         const float_t *pw = &this->w_[weight_.get_index(0, 0, in_.depth_ * o + inc)];
12.         const float_t *pi = &in[in_padded_.get_index(0, 0, inc)];
13.         float_t *pa = &a[out_.get_index(0, 0, o)];
14.
15.         for (layer_size_t y = 0; y < out_.height_; y++) {
```

关闭

- libsvm
- joys99
- CodeForge
- cvchina
- tesseract-ocr
- sift
- TiRG
- imgSeek
- OpenSURF

Friendly Link

- OpenCL
- Python
- poesia-filter
- TortoiseSVN
- imgSeek
- Notepad
- Beyond Compare
- CMake
- VIGRA
- CodeGuru
- vcchome
- aforgenet
- CVLAB
- Doxygen
- Coursera
- OpenMP

Technical Forum

- Matlab China
- OpenCV China
- The CImg Library
- Open Computer Vision Library
- CxImage
- ImageMagick
- ImageMagick China
- OpenCV_China
- Subversion China

```
16.         for (layer_size_t x = 0; x < out_.width_; x++) {
17.             const float_t * ppw = pw;
18.             const float_t * ppi = pi + (y * h_stride_) * in_padded_.width_ + x * w_stride_
19.             float_t sum = (float_t)0.0;
20.
21.             // should be optimized for small kernel(3x3,5x5)
22.             for (layer_size_t wy = 0; wy < weight_.height_; wy++) {
23.                 for (layer_size_t wx = 0; wx < weight_.width_; wx++) {
24.                     sum += *ppw++ * ppi[wy * in_padded_.width_ + wx];
25.                 }
26.             }
27.             pa[y * out_.width_ + x] += sum;
28.         }
29.     }
30. }
31.
32. if (!this->b_.empty()) {
33.     float_t *pa = &a[out_.get_index(0, 0, 0)];
34.     float_t b = this->b_[0];
35.     std::for_each(pa, pa + out_.width_ * out_.height_, [&](float_t& f) { f += b; });
36. }
37. });
38.
39. for_i(parallelize_, out_size_, [&](int i) {
40.     out[i] = h_.f(a, i);
41. });
```

S2层代码段：

```
[cpp]
01. vec_t& a = a_[index];
02.
03. for_i(parallelize_, out_size_, [&](int i) {
04.     const wi_connections& connections = out2wi_[i];
05.
06.     a[i] = 0.0;
07.
08.     for (auto connection : connections) // 13.1%
09.         a[i] += W_[connection.first] * in[connection.second]; // 3.2%
10.
11.     a[i] *= scale_factor_;
12.     a[i] += b_[out2bias_[i]];
13. });
14.
15. for_i(parallelize_, out_size_, [&](int i) {
16.     output_[index][i] = h_.f(a, i);
17. });
```

C3层、C5层代码段与C1层相同。

S4层代码段与S2层相同。

输出层代码段：

```
[cpp]
01. vec_t &a = a_[index];
02. vec_t &out = output_[index];
03.
04. for_i(parallelize_, out_size_, [&](int i) {
05.     a[i] = 0.0;
06.     for (layer_size_t c = 0; c < in_size_; c++) {
07.         a[i] += W_[c*out_size_ + i] * in[c];
08.     }
09.
10.     if (has_bias_)
11.         a[i] += b_[i];
12. });
13.
14. for_i(parallelize_, out_size_, [&](int i) {
15.     out[i] = h_.f(a, i);
16. });
```

反向传播：

关闭

Technical Blog

邹宇华

深之JohnChen

HUNNISH

周伟明

superdant

carson2005

OpenHero

Netman(Linux)

wqvbjhc

yang_xian521

gnuipc

gnuipc

千里8848

CVART

tornadomeet

gotosuc

onezeros

hellogv

abcjennifer

czy_sparrow

评论排行

Windows7 32位机上, O (120)

tiny-cnn开源库的使用(MI (93)

Ubuntu 14.04 64位机上 (89)

tesseract-ocr3.02字符识 (63)

Windows7上使用VS201 (47)

tesseract-ocr (42)

图像配准算法 (41)

Windows 7 64位机上Op (36)

OpenCV中resize函数五 (34)

小波矩特征提取matlab代 (30)

最新评论

Tesseract-OCR 3.04在Windows
fengbingchun: @iliked: 没有密码, 那个commit只是提示是从哪个commit fork过来的, 无需管那个

Tesseract-OCR 3.04在Windows
iliked: 问一下, 你第一句中的commit的那个密码, 怎么用啊
卷积神经网络(CNN)的简单实现(fengbingchun: @hugl950123: 是需要opencv的支持, 你在本地opencv的环境配好了吗, 配好了就应该没...

卷积神经网络(CNN)的简单实现(hugl950123: @fengbingchun: 博主请问一下, test_CNN_predict()函数是不是需要open...

卷积神经网络(CNN)的简单实现(hugl950123: @fengbingchun: 博主请问一下, test_CNN_predict()函数是不是需要open...

卷积神经网络(CNN)的简单实现(hugl950123: @fengbingchun: 谢谢, 能够成功运行了现在

卷积神经网络(CNN)的简单实现(fengbingchun: @hugl950123: NN中一共有四个工程, 它们之间没有任何关系, 都是独立的, 如果要运行这篇文章的...

输出层代码段:

```
[cpp]
01. vec_t delta(out_dim());
02. const activation::function& h = layers_.tail()->activation_function();
03.
04. if (is_canonical_link(h)) {
05.     for_i(out_dim(), [&](int i){ delta[i] = out[i] - t[i]; });
06. } else {
07.     vec_t dE_dy = gradient<E>(out, t);
08.
09.     // delta = dE/da = (dE/dy) * (dy/da)
10.     for (size_t i = 0; i < out_dim(); i++) {
11.         vec_t dy_da = h.df(out, i);
12.         delta[i] = vectorize::dot(&dE_dy[0], &dy_da[0], out_dim());
13.     }
14. }
```

C5层代码段:

```
[cpp]
01. const vec_t& prev_out = prev_->output(index);
02. const activation::function& prev_h = prev_->activation_function();
03. vec_t& prev_delta = prev_delta_[index];
04. vec_t& dw = dw_[index];
05. vec_t& db = db_[index];
06.
07. for (layer_size_t c = 0; c < this->in_size_; c++) {
08.     // propagate delta to previous layer
09.     // prev_delta[c] += current_delta[r] * W[c * out_size_ + r]
10.     prev_delta[c] = vectorize::dot(&curr_delta[0], &W[c*out_size_], out_size_);
11.     prev_delta[c] *= prev_h.df(prev_out[c]);
12. }
13.
14. for_(parallelize_, 0, (size_t)out_size_, [&](const blocked_range& r) {
15.     // accumulate weight-step using delta
16.     // dw[c * out_size_ + i] += current_delta[i] * prev_out[c]
17.     for (layer_size_t c = 0; c < in_size_; c++)
18.         vectorize::muladd(&curr_delta[r.begin()], prev_out[c], r.end() - r.begin(), &dw[c*
19.
20.     if (has_bias_) {
21.         for (int i = r.begin(); i < r.end(); i++)
22.             db[i] += curr_delta[i];
23.     }
24. });
```

S4层代码段:

```
[cpp]
01. const vec_t& prev_out = *(prev_out_padded_[index]);
02. const activation::function& prev_h = prev_->activation_function();
03. vec_t* prev_delta = (pad_type_ == padding::same) ? &prev_delta_padded_[index] : &prev_delta;
04. vec_t& dw = dw_[index];
05. vec_t& db = db_[index];
06.
07. std::fill(prev_delta->begin(), prev_delta->end(), (float_t)0.0);
08.
09. // propagate delta to previous layer
10. for_i(in_.depth_, [&](int inc) {
11.     for (layer_size_t outc = 0; outc < out_.depth_; outc++) {
12.         if (!tbl_.is_connected(outc, inc)) continue;
13.
14.         const float_t *pw = &this->W_[weight_.get_index(0, 0, in_.depth_ * outc + inc)];
15.         const float_t *pdelta_src = &curr_delta[out_.get_index(0, 0, outc)];
16.         float_t *pdelta_dst = &(prev_delta)[in_padded_.get_index(0, 0, inc)];
17.
18.         for (layer_size_t y = 0; y < out_.height_; y++) {
19.             for (layer_size_t x = 0; x < out_.width_; x++) {
20.                 const float_t *ppw = pw;
21.                 const float_t ppdelta_src = pdelta_src[y * out_.width_ + x];
22.                 float_t *ppdelta_dst = pdelta_dst + y * h_stride_ * in_padded_.width_ +
23.
24.                 for (layer_size_t wy = 0; wy < weight_.height_; wy++) {
25.                     for (layer_size_t wx = 0; wx < weight_.width_; wx++) {
26.                         ppdelta_dst[wy * in_padded_.width_ + wx] += *ppw++ * ppdelta_src;
```

关闭

卷积神经网络(CNN)的简单实现(
hugl950123: @fengbingchun:下
的是新的,我在CNN.cpp文件中
每个函数都设置了断点,还是没
有变化=...

卷积神经网络(CNN)的简单实现(
fengbingchun: @hugl950123:你
用的是GitHub上最新的吗?既然
能编译过,在Debug下设断点,
应该很快...

卷积神经网络(CNN)的简单实现(
hugl950123: 博主,请问我按照
您的代码成功编译后执行结果窗
口一闪而过,并且里面什么内容
也没有,应该如何解决,能不
能...

阅读排行

- C#中OpenFileDialog的假
tesseract-ocr3.02字符识 (47141)
举例说明使用MATLAB C (34575)
OpenCV中resize函数五 (25987)
利用cvMinAreaRect2求 (24317)
Windows 7 64位机上搭建 (24277)
opencv 检测直线、线段、 (22586)
OpenCV运动检测跟踪(b (20776)
图像配准算法 (20475)
有效的rtsp流媒体测试地: (19237)
(19143)

文章存档

- 2017年01月 (18)
2016年12月 (11)
2016年11月 (8)
2016年10月 (7)
2016年09月 (16)

展开



碧桂园森林城市



```

27.         }
28.     }
29. }
30. }
31. }
32. });
33.
34. for_i(parallelize_, in_padded_.size(), [&](int i) {
35.     (*prev_delta)[i] *= prev_h.df(prev_out[i]);
36. });
37.
38. // accumulate dw
39. for_i(in_.depth_, [&](int inc) {
40.     for (layer_size_t outc = 0; outc < out_.depth_; outc++) {
41.
42.         if (!tbl_.is_connected(outc, inc)) continue;
43.
44.         for (layer_size_t wy = 0; wy < weight_.height_; wy++) {
45.             for (layer_size_t wx = 0; wx < weight_.width_; wx++) {
46.                 float_t dst = 0.0;
47.                 const float_t * prevo = &prev_out[in_padded_.get_index(wx, wy, inc)];
48.                 const float_t * delta = &curr_delta[out_.get_index(0, 0, outc)];
49.
50.                 for (layer_size_t y = 0; y < out_.height_; y++) {
51.                     dst += vectorize::dot(prevo + y * in_padded_.width_, delta + y * out_
52.                 }
53.                 dw[weight_.get_index(wx, wy, in_.depth_ * outc + inc)] += dst;
54.             }
55.         }
56.     }
57. });
58.
59. // accumulate db
60. if (!db.empty()) {
61.     for (layer_size_t outc = 0; outc < out_.depth_; outc++) {
62.         const float_t *delta = &curr_delta[out_.get_index(0, 0, outc)];
63.         db[outc] += std::accumulate(delta, delta + out_.width_ * out_.height_, (float_t)0
64.     }
65. }

```

C3层代码段 :

```

[cpp]
01. const vec_t& prev_out = prev_->output(index);
02. const activation::function& prev_h = prev_->activation_function();
03. vec_t& prev_delta = prev_delta_[index];
04.
05. for_(parallelize_, 0, (size_t)in_size_, [&](const blocked_range& r) {
06.     for (int i = r.begin(); i != r.end(); i++) {
07.         const wo_connections& connections = in2wo_[i];
08.         float_t delta = 0.0;
09.
10.         for (auto connection : connections)
11.             delta += W_[connection.first] * current_delta[connection.second]; // 40.6%
12.
13.         prev_delta[i] = delta * scale_factor_ * prev_h.df(prev_out[i]); // 2.1%
14.     }
15. });
16.
17. for_(parallelize_, 0, weight2io_.size(), [&](const blocked_range& r) {
18.     for (int i = r.begin(); i < r.end(); i++) {
19.         const io_connections& connections = weight2io_[i];
20.         float_t diff = 0.0;
21.
22.         for (auto connection : connections) // 11.9%
23.             diff += prev_out[connection.first] * current_delta[connection.second];
24.
25.         dw_[index][i] += diff * scale_factor_;
26.     }
27. });
28.
29. for (size_t i = 0; i < bias2out_.size(); i++) {
30.     const std::vector<layer_size_t>& outs = bias2out_[i];
31.     float_t diff = 0.0;
32.
33.     for (auto o : outs)
34.         diff += current_delta[o];
35. }

```

关闭

```
36.         db_[index][i] += diff;
37.     }
```

S2层、输入层代码段与S4层相同。

C1层代码段与C3层相同。

权值和偏置更新代码段：

[cpp] C P

```
01. void update(const vec_t& dw, const vec_t& /*Hessian*/, vec_t &W) {
02.     vec_t& g = get<0>(W);
03.
04.     for_i(W.size(), [&](int i) {
05.         g[i] += dw[i] * dw[i];
06.         W[i] -= alpha * dw[i] / (std::sqrt(g[i]) + eps);
07.     });
08. }
```

对MNIST中的60000个训练样本，依次执行上面的操作，并更新权值和偏置。

每此循环执行完60000个训练样本，会对10000个测试样本，进行测试，获得识别率。

共迭代30次，然后将最终的权值、偏置等相关参数保持到指定的文件中。

顶

0

踩

0

上一篇 [tiny-cnn开源库的使用\(MNIST\)](#)

下一篇 [VLFeat开源库介绍及在VS2013中的编译](#)

我的同类文章

Caffe (19)	Deep Learning (8)	Neural Network (12)
<ul style="list-style-type: none">Caffe中Layer注册机制 2017-01-10 阅读 87cifar数据集介绍及到图像转... 2016-12-10 阅读 245卷积神经网络(CNN)代码实... 2016-12-03 阅读 2188Windows7 64bit VS2013 Ca... 2016-03-26 阅读 1550tiny-cnn开源库的使用(MNIST) 2016-01-24 阅读 9464		<ul style="list-style-type: none">windows7下解决caffe check... 2017-01-09 阅读 121深度学习开源库tiny-dnn的使... 2016-12-04 阅读 465一步一步指引你在Windows... 2016-03-26 阅读 1381卷积神经网络(CNN)的简单... 2016-03-06 阅读 7518卷积神经网络(CNN)基础介绍 2016-01-16 阅读 11361
更多文章		

竞争策略

书法高考

什么是学习综述

代码解析

程序员培训机构

layer

脉冲神经网络

高老书法

猜你在找

Visual Studio 2015开发C++程序的基本使用

Swift与Objective-C\C\C++混合编程

C++语言基础

C++标准模板库从入门到精通

卷积神经网络CNN的简单实现MNIST

TensorFlow教程06MNIST的CNN实现源码和运行结果

tensorflow学习笔记五mnist实例--卷积神经网络CNN

卷积神经网络CNN的简单实现MNIST

Android入门必备之C++快速入门

MNIST手写数字的识别CNN篇

jQuery MiniUI , Ajax Web开发

了解详情

查看评论

1楼 张嘉晖2 2016-05-02 20:58发表

迭代次数怎么改？

Re: fengbingchun 2016-05-03 08:29发表

回复u013139259：迭代次数是由上层控制的。可参考<http://blog.csdn.net/fengbingchun/article/details/50573841>，由int num_epochs = 30;指定迭代次数

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

核心技术类目

全部主题

Hadoop

AWS

移动游戏

Java

Android

iOS

Swift

智能硬件

Docker

Op

VPN

Spark

ERP

IE10

Eclipse

CRM

JavaScript

数据库

Ubuntu

NFC

WAP

jQuery

BI

HTML5

Spring

Apache

.NET

API

HTML

SDK

IIS

Fedora

XML

LBS

Unity

Splashtop

UML

components

Windows Mobile

Rails

QEMU

KDE

Cassandra

CloudStack

FTC

coremail

OPhone

CouchBase

云计算

iOS6

Rackspace

Web App

SpringSide

Maemo

Compuware

大数据

aptech

Perl

Tornado

Ruby

Hibernate

ThinkPHP

HBase

Pure

Solr

Angular

Cloud Foundry

Redis

Scala

Django

Bootstrap

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

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-600-2320 | 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 |

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2016, CSDN.NET, All Rights Reserved

关闭