反向傳播神經網絡

Tejpal Singh Chhabra， 2006年3月29日

★★★★★ 4.76（38票）　率：

一個實現反向傳播算法神經網絡的C ++類，支持任意數量的層/神經元。

下載演示項目 - 4.64 Kb

介紹

該類CBackProp封裝了一個前饋神經網絡和一個反向傳播算法來訓練它。本文適用於那些已經對神經 網絡和反向傳播算法有所了解的人。如果你不熟悉這些，我建議首先閱讀一些材料。

背景

這是我在大學期間最後一學期期間研究的一個學術項目的一部分，為此我需要為不同數據集找到隱藏層的最佳數量和大小以及學習參數。要確定神經網絡的數據結構並使反向傳播算法正常工作並不容易。這篇文章的動機是拯救別人同樣的努力。

這裡有一點小小的免責聲明......本文描述了該算法的簡單實現，並沒有充分闡述該算法。改進包含的代碼有很大的空間（比如添加異常處理:-)，並且對於很多步驟，需要比我包含的更多的推理，例如，我為參數選擇的值，以及數量層/每層中的神經元用於演示使用情況並且可能不是最佳的。要了解更多這些，我建議去：

- 神經網絡常見問題

使用代碼

通常，使用涉及以下步驟：

- 使用創建網絡`CBackProp::CBackProp(int nl,int *sz,double b,double a)`
- 應用反向傳播算法 - 通過將輸入和期望輸出傳遞到`void CBackProp::bpgt(double *in,double *tgt)`一個循環來訓練網絡，直到通過得到的*均方誤差*`CBackProp::double mse(double *tgt)`減小到可接受的值。
- 使用訓練後的網絡通過使用*前饋*輸入數據來進行預測`void CBackProp::ffwd(double *in)`。

以下是我所包含的示例程序的說明。

一步一步來...

設定目標：

我們會盡力教我們的淨破解二元一***XOR***乙***XOR Ç***。異或是一個明顯的選擇，它不是線性可分的，因此需要隱藏層，不能通過單一的知覺來學習。

訓練數據集由多個記錄組成，其中每個記錄包含輸入到網絡的字段，其後是包含期望輸出的字段。在這個例子中，它是三個輸入+一個期望的輸出。

隱藏 複製代碼

```
// prepare XOR training data
double data[][4]={//    I  XOR  I  XOR  I   =   O
                  //--------------------------------
                        0,      0,      0,      0,
                        0,      0,      1,      1,
                        0,      1,      0,      1,
                        0,      1,      1,      0,
                        1,      0,      0,      1,
                        1,      0,      1,      0,
                        1,      1,      0,      0,
                        1,      1,      1,      1 };
```

組態：

接下來，我們需要為我們的神經網絡指定*一個合適的*結構，即它應該具有的隱藏層數以及每層中的神經元數量。然後，我們為其他參數指定*合適的*值：**學習速率** - ，我們可能還想指定**動量** - （這個是可選的）和**閾值** - （目標*均方差*，一旦達到其他訓練停止，訓練停止數次）。`betaalphathreshnum_iter`

我們定義一個網絡，分別有3,3,3和1個神經元。由於第一層是輸入層，即只是輸入參數的佔位符，它必須與輸入參數的數量相同，最後一層是輸出層必須與輸出數量相同- 在我們的例子中，它們是3和1.中間的那些其他層被稱為*隱藏層*。

隱藏 複製代碼

```
int numLayers = 4, lSz[4] = {3,3,3,1};
double beta = 0.2, alpha = 0.1, thresh = 0.00001;
long num_iter = 500000;
```

創建網絡：

隱藏 複製代碼

```
CBackProp *bp = new CBackProp(numLayers, lSz, beta, alpha);
```

訓練：

隱藏 複製代碼

```
for (long i=0; i < num_iter ; i++)
{
    bp->bpgt(data[i%8], &data[i%8][3]);

    if( bp->mse(&data[i%8][3]) < thresh)
        break; // mse < threshold - we are done training!!!
}
```

讓我們來測試它的智慧：

我們準備*測試數據*，這裡的*數據*與*培訓數據*減去所需的輸出*數據*相同。

隱藏 複製代碼

```
double testData[][3]={ //  I  XOR  I  XOR  I  =  ?
                        //----------------------
                            0,      0,      0,
                            0,      0,      1,
                            0,      1,      0,
                            0,      1,      1,
                            1,      0,      0,
                            1,      0,      1,
                            1,      1,      0,
                            1,      1,      1};
```

現在，使用訓練好的網絡對我們的測試數據進行預測....

隱藏 複製代碼

```
for ( i = 0 ; i < 8 ; i++ )
{
    bp->ffwd(testData[i]);
    cout << testData[i][0]<< "  "
         << testData[i][1]<< "  "
         << testData[i][2]<< "  "
         << bp->Out(0) << endl;
}
```

現在看一看：

存儲神經網絡

我認為下面的代碼有充足的意見，並且不言自明...

隱藏 縮小 ▲ 複製代碼

```
class CBackProp{

//      output of each neuron
        double **out;

//      delta error value for each neuron
        double **delta;

//      3-D array to store weights for each neuron
        double ***weight;

//      no of layers in net including input layer
        int numl;

//      array of numl elements to store size of each layer
        int *lsize;

//      learning rate
        double beta;

//      momentum
        double alpha;

//      storage for weight-change made in previous epoch
        double ***prevDwt;

//      sigmoid function
        double sigmoid(double in);
public:

        ~CBackProp();

//      initializes and allocates memory
        CBackProp(int nl,int *sz,double b,double a);

//      backpropogates error for one set of input
        void bpgt(double *in,double *tgt);

//      feed forwards activations for one set of inputs
        void ffwd(double *in);

//      returns mean square error of the net
        double mse(double *tgt);

//      returns i'th output of the net
        double Out(int i) const;
};
```

一些替代實現為層/神經元/連接定義了單獨的類，然後將它們組合在一起形成神經網絡。雖然它絕對是一種更清潔的方法，但我決定通過分配所需的確切內存量來使用double ***和double **存儲權重和輸出等，原因如下：

- 它同時位於（i-1）之間的連接實現學習算法，例如，體重為緩解次層的殲個神經元和神經我個層為其k個神經元，我個人比較喜歡w[i][k][j]（不是像net.layer[i].neuron[k].getWeight(j)）。thj th層的第i個神經元的輸出為out[i][j]，依此類推。
- 我感受到的另一個優點是可以靈活選擇任意數量和大小的圖層。

隱藏 縮小 ▲ 複製代碼

```
// initializes and allocates memory
CBackProp::CBackProp(int nl,int *sz,double b,double a):beta(b),alpha(a)
{

// Note that the following are unused,
//
// delta[0]
// weight[0]
// prevDwt[0]

//  I did this intentionally to maintain
//  consistency in numbering the layers.
//  Since for a net having n layers,
//  input layer is referred to as 0th layer,
//  first hidden layer as 1st layer
//  and the nth layer as output layer. And
//  first (0th) layer just stores the inputs
//  hence there is no delta or weight
//  values associated to it.


    //    set no of layers and their sizes
    numl=nl;
    lsize=new int[numl];

    for(int i=0;i<numl;i++){
        lsize[i]=sz[i];
    }

    //    allocate memory for output of each neuron
    out = new double*[numl];

    for( i=0;i<numl;i++){
        out[i]=new double[lsize[i]];
    }

    //    allocate memory for delta
    delta = new double*[numl];

    for(i=1;i<numl;i++){
        delta[i]=new double[lsize[i]];
    }
    //    allocate memory for weights
    weight = new double**[numl];

    for(i=1;i<numl;i++){
        weight[i]=new double*[lsize[i]];
    }
    for(i=1;i<numl;i++){
        for(int j=0;j<lsize[i];j++){
            weight[i][j]=new double[lsize[i-1]+1];
        }
    }

    //    allocate memory for previous weights
    prevDwt = new double**[numl];

    for(i=1;i<numl;i++){
        prevDwt[i]=new double*[lsize[i]];

    }
    for(i=1;i<numl;i++){
        for(int j=0;j<lsize[i];j++){
            prevDwt[i][j]=new double[lsize[i-1]+1];
        }
    }

    //    seed and assign random weights
    srand((unsigned)(time(NULL)));
    for(i=1;i<numl;i++)
        for(int j=0;j<lsize[i];j++)
            for(int k=0;k<lsize[i-1]+1;k++)
                weight[i][j][k]=(double)(rand())/(RAND_MAX/2) - 1;

    //    initialize previous weights to 0 for first iteration
    for(i=1;i<numl;i++)
        for(int j=0;j<lsize[i];j++)
            for(int k=0;k<lsize[i-1]+1;k++)
                prevDwt[i][j][k]=(double)0.0;
}
```

前饋

該功能更新每個神經元的輸出值。首先從第一個隱藏層開始，它將輸入傳遞給每個神經元，並o首先計算輸入的加權和，然後將Sigmoid函數應用於輸入（ ），並將其傳遞到下一層，直到輸出層為更新：

$$o = \sigma(\vec{w}\vec{x})$$

哪裡：

$$\sigma(y) = \frac{1}{1+e^{-y}}$$

隱藏 縮小 ▲ 複製代碼

```
// feed forward one set of input
void CBackProp::ffwd(double *in)
{
        double sum;

 // assign content to input layer

        for(int i=0;i < lsize[0];i++)
                out[0][i]=in[i];


// assign output(activation) value
// to each neuron usng sigmoid func

        // For each layer
        for(i=1;i < numl;i++){
                // For each neuron in current layer
                for(int j=0;j < lsize[i];j++){
                        sum=0.0;
                        // For input from each neuron in preceding layer
                        for(int k=0;k < lsize[i-1];k++){
                                // Apply weight to inputs and add to sum
                                sum+= out[i-1][k]*weight[i][j][k];
                        }
                        // Apply bias
                        sum+=weight[i][j][lsize[i-1]];
                        // Apply sigmoid function
                        out[i][j]=sigmoid(sum);
                }
        }
}
```

向後傳播的...

該算法在函數中實現`void CBackProp::bpgt(double *in,double *tgt)`。以下是涉及在輸出層中向後傳播錯誤直到第一個隱藏層的各個步驟。

隱藏 複製代碼

```
void CBackProp::bpgt(double *in,double *tgt)
{
    double sum;
```

首先，我們打電話`void CBackProp::ffwd(double *in)`更新每個神經元的輸出值。該函數將輸入傳遞給網絡並查找每個神經元的輸出：

$$o = \sigma(\vec{w}\vec{x})$$

哪裡：

$$\sigma(y) = \frac{1}{1+e^{-y}}$$

隱藏 複製代碼

```
ffwd(in);
```

下一步是找出輸出層的增量：

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

隱藏 複製代碼

```
for(int i=0;i < lsize[numl-1];i++){
   delta[numl-1][i]=out[numl-1][i]*
     (1-out[numl-1][i])*(tgt[i]-out[numl-1][i]);
}
```

然後找到隱藏層的增量...

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{kh}\delta_k$$

隱藏 複製代碼

```
for(i=numl-2;i>0;i--){
    for(int j=0;j < lsize[i];j++){
        sum=0.0;
        for(int k=0;k < lsize[i+1];k++){
                sum+=delta[i+1][k]*weight[i+1][k][j];
        }
        delta[i][j]=out[i][j]*(1-out[i][j])*sum;
    }
}
```

運用動力（如果alpha = 0，則不做任何事情）：

隱藏 複製代碼

```
for(i=1;i < numl;i++){
    for(int j=0;j < lsize[i];j++){
        for(int k=0;k < lsize[i-1];k++){
                weight[i][j][k]+=alpha*prevDwt[i][j][k];
        }
        weight[i][j][lsize[i-1]]+=alpha*prevDwt[i][j][lsize[i-1]];
    }
}
```

最後，通過找到對重量的修正來調整重量。

$$\Delta w_{ji} = \eta \delta_j x_{ji}$$

然後應用更正：

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

隱藏 複製代碼

```
for(i=1;i < numl;i++){
    for(int j=0;j < lsize[i];j++){
        for(int k=0;k < lsize[i-1];k++){
                prevDwt[i][j][k]=beta*delta[i][j]*out[i-1][k];
                weight[i][j][k]+=prevDwt[i][j][k];
        }
        prevDwt[i][j][lsize[i-1]]=beta*delta[i][j];
        weight[i][j][lsize[i-1]]+=prevDwt[i][j][lsize[i-1]];
    }
}
```

如何學習網絡？

*均方誤差*被用來衡量神經網絡學習的程度。

$$E(\vec{w}) = \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2$$

正如示例XOR程序中所示，我們應用上述步驟直到達到滿意的低錯誤級別。`CBackProp::double mse(double *tgt)`只是回報。

歷史

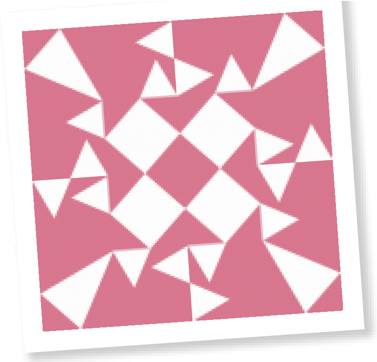
- 創建日期：2006年3月25日。

執照

分享

推特 FACEBOOK的

關於作者



Tejpal Singh Chhabra

軟件開發人員
澳大利亞

跟隨
這位會員

對軟件開發各個方面感興趣的技術愛好者。我主要在Unix / Linux上使用C，C ++，Java和Go編碼。

您也可能對。。。有興趣...

C＃上的神經網絡

開始使用英特爾®Parallel Studio XE對您的應用程序進行Turbo-Charging

神經網絡Pt 4神經網絡測試儀

使用英特爾®實感™攝像頭，ROS和SAWR構建自主移動機器人

使用IOT和移動邊緣計算優化緊急車輛的流量

SAPrefs - 類似Netscape的首選項對話框

評論和討論

添加評論或問題 Search Comments

首頁 上一頁 下一頁

利用matlab的反向傳播控制算法得到三相參考電流
Member 12111737 27-Nov-15 22:20

plz幫助得到這個

回复 · 電子郵件 · 查看主題

精美的一段代碼
SoothingMist 7-Jan-15 16:56

發現這項工作是相當不錯的。它評論很好，易於遵循。

對未來的建議是在可執行行的組件之間使用拼寫有意義的變量名稱和空白區域。

手頭應用程序的修改很簡單。

由於代碼似乎是使用Visual Studio和C ++的較舊版本開發的，因此有必要創建一個全新的項目，並對Visual Studio Express v2013及其對C ++ v11的使用進行一些修改。將查看是否可以將新代碼通過電子郵件發送給原作者。歡迎他發布。

回复 · 電子郵件 · 查看主題

乙狀結構衍生物

Raphael Gruaz 2-Oct-12 18:12

Thanks a lot for this article. It really helped me to understand how to implement a neural network. However, there is something I still cannot understand from your code: I thought the derivative function of Sigmoid was something like DS(x) = S(x)(1-S(x)), but I cannot see it anywhere in your back-propagation function...

Did I miss something? Or did you use a simplified form?
Thanks again,

Raphael

Reply · Email · View Thread

Re: Sigmoid derivative

Annisa Kartikasari 7-Jul-15 13:35

same question with me, it just written in sigmoid, and there is no definition what is sigmoid in the code. Have you solve this by yourself?

Reply · Email · View Thread

Overflow

Miguel Tomas 6-Aug-12 1:06

on the traning code, the data array has only 8 elements how could be on a iteration with 200.000 steps
here's the code:

Hide Copy Code

```
for (long i=0; i < num_iter ; i++)
{
    bp->bpgt(data[i%8], &data[i%8][3]);

    if( bp->mse(&data[i%8][3]) < thresh)
        break; // mse < threshold - we are done training!!!
}
```

and also the second argument of bp->bpgt has to be an array and on the above code it isn't.

Can anyone help me with this?

regards

Reply · Email · View Thread

How to save the network?

tooym 21-Jan-10 11:05

Firstly,thanks to the author for providing me a very useful backpropagation neural network in C++. Regarding to the neural network as shown above, how can i save the network 'bp' permanently for the further testing with others input?
Thanks.

Reply · Email · View Thread

How to minimize Total Network Error

AjayIndian 1-Jan-10 17:21

Dear Sir,
I have developed the code for back-propagation algo.,but I m not able to minimize the Total Network Error(Global Error),How it can be done.
Actually after learning,when I m presenting the test pattern to the network,the network is giving the result as the last result given by the network for last training pattern.

looking for ur kind cooperation.

ajay

Reply · Email · View Thread

Understanding the basis for code

locuaz 17-Oct-09 16:34

Hi dude, your article is interesting and didactic, I'm focused on understanding your backpropagation algorithm, so I need you can tell me which books or sources you used to implement it into C++, please. I'm having serious problems to adopt/understand the equation tp update weigths, 'cause there are some books and papers using different representations for it, including you. For example, someone is using this formula:
weight(new) = weight(old) + learning rate * output error * output(neurons i) * output(neurons i+1) * (1 - output(neurons i+1))
And my doubt is, wtf he gets:
output(neurons i) * output(neurons i+1) * (1 - output(neurons i+1))
It doesnot appear in my books 😕

Reference: Backpropagation
Thanks in advance.

I'm in love with Delphi 😎

Billiardo

Reply · Email · View Thread

how can I get the source code

xumin1988 17-Oct-09 0:03

dear Sir,
I am a new user!I can find the demo file,but where is the source?Please help me.Thank you !

Reply · Email · View Thread

Bias value?

emrecaglar 13-Oct-09 3:30

I was wondering where the bias weights are set to 1 or -1? I think they are initialized to random values in constructor.Am i right? Many thanks

Reply · Email · View Thread

Out(0)

tangsu 1-Sep-09 20:10

Thank you for uploading your code for implementation of backpropagation neural networks.

I have a question about statement: ">> bp->Out(0) >> endl;"

Function "Out" has no reference anywhere in the code. Could you please clarify on this?

Tang Su

Reply · Email · View Thread

telecom billing

Sofritom 13-Jul-09 19:32

Hi Tejpal,

I wish to know about the telecom billing part - did you worked with FTS? what does is has to do with Neural network?

Reply · Email · View Thread

Error in the code [modified]

gpwr9k95 12-Jun-09 7:08

The code computes MSE for just one training value. Check these lines in main():

Hide Copy Code

```
for (i=0; i<num_iter ; i++)
{
    bp->bpgt(data[i%8], &data[i%8][3]);
    if( bp->mse(&data[i%8][3]) < Thresh) {
```

The learning iterations are arranged such that only one training value is checked at each iteration steps. As a result, for a net with one output, like in the article example, MSE simply becomes the squared error between the net output and the training value, i.e. (Net_Out - Target)^2. This is wrong because the iterations stop when MSE is less than Threshold for just one training value.
Instead, iterations should be arranged in epochs where one epoch contains the complete scan over all training values and MSE is calculated by summing squared errors between the net outputs and the corresponding training values for each epoch as in the example below (ntr - number of training samples):

Hide Copy Code

```
double MSE;
int ep;
for (ep=0;ep<nep;ep++)
{
    MSE=0.0;
    for(int i=0;i<ntr;i++)
    {
        bp->bpgt(data[i], &data[i][lSz[0]]);
        MSE+=bp->mse(&data[i][lSz[0]]);
    }
    MSE/=ntr;
    if(MSE<minErr)
    {
        cout << "Network trained in " << ep << " epochs. MSE: " << MSE << endl;
        break;
    }
```

Otherwise, the code is fairly simple and easy to understand.
Thanks.

modified on Friday, June 12, 2009 12:07 AM

Reply · Email · View Thread 5.00/5 (1 vote)

is this a bug?

Member 429450 19-Apr-09 9:18

```
for(i=1;i < numl;i++){
for(int j=0;j < lsize[i];j++){
for(int k=0;k < lsize[i-1];k++){
prevDwt[i][j][k]=beta*delta[i][j]*out[i-1][k];
weight[i][j][k]+=prevDwt[i][j][k];
}
prevDwt[i][j][lsize[i-1]]=beta*delta[i][j]; <----------------------here
weight[i][j][lsize[i-1]]+=prevDwt[i][j][lsize[i-1]];
}
}
```

i think should be..

```
for(i=1;i < numl;i++){
for(int j=0;j < lsize[i];j++){
for(int k=0;k <= lsize[i-1];k++) { <---------------------------here
prevDwt[i][j][k]=beta*delta[i][j]*out[i-1][k];
weight[i][j][k]+=prevDwt[i][j][k];
}
}
}
```

I made similar modifications to the momentum code as well.

Another issue that i noticed is that iteration stops whenever the error for _one_ of the training data falls below the threshold.

Other than it works good.
Thanks

Reply · Email · View Thread

How about bias

KadirErturk 9-Mar-09 0:36

great code. Now I am trying to convert your code to complex number domain. I mean the element of net has complex number (i.e a+ib)

Would you correct me if am I wrong;

in your code

```
// Apply bias
sum+=weight[i][j][lsize[i-1]];
// Apply sigmoid function
....
```

so, is it mean that bias is constant for any layer. I think each neuron should have its own bias.

regards
Kadir

Reply · Email · View Thread

Normalized values

picand 6-May-08 20:09

Hello every one !
Does this algorithm works with values that are not between 0 and 1, or do I have to normalize them to implement them ?
Thank you.
PA

Reply · Email · View Thread

Feedforward Backpropagation Neural Network

raceng0585 7-Dec-07 23:41

In MatLab,

1st. we use purelin function p(x)=x where x is the summation of total weight multiple with bias that pass thought the output neuron in feedforward, but what is the dpurelin equation/formula that used in backpropagation?

2nd. we use logsig function logsig(x)=1/(1+e^(-x)) where x is the summation of total weight multiple with bias that pass thought the hidden neuron in feedforward method, but wat is the dlogsig equation/formula that used in backpropagation?

3rd. how to write the traingd or trainda function code in C style?

Reply · Email · View Thread

Trouble converting to C

pradeep swamy 19-Nov-07 16:37

This NN code is very fast. I'm converting this code to C, but having trouble while converting "***weight" variable to C. How to convert multiple pointer variable into C? Do you have a C equivalent code of this NN?

Reply · Email · View Thread

Re: Trouble converting to C

robiii 16-Jul-09 23:31

in c you cant use the new operator.
just use malloc

Reply · Email · View Thread

Scaling Input

ravenspoint 16-Oct-07 22:44

Although CBackProp will handle any range of input (unlike the outputs which must be in the range 0,1 - see my earlier post) I have found that if I also scale the inputs to the range 0,1 the results are improved, sometimes greatly.

The example below shows the results of experimenting with a time series generated from the function y = 0.5 sine(0.5 x)

Hide Expand ▼ Copy Code

```
Input        Actual       Pred        Error       %
  0.000000   0.479462   0.366899   0.112563   23.476872
  1.000000   0.488765   0.357821   0.130944   26.790794
  2.000000   0.378401   0.338155   0.040246   10.635803
  3.000000   0.175392   0.293070  -0.117679   67.094771
  4.000000  -0.070560   0.182579  -0.253139 -358.756813
  5.000000  -0.299236  -0.070652  -0.228584 -76.389083
  6.000000  -0.454649  -0.376426  -0.078223 -17.205125
  7.000000  -0.498747  -0.470585  -0.028163  -5.646709
  8.000000  -0.420735  -0.230091  -0.190645 -45.312249
  9.000000  -0.239713   0.271760  -0.511473 -213.368952
 10.000000   0.000000   0.328039  -0.328039   1.#INF00
 11.000000   0.239713   0.294158  -0.054445  22.712536
 12.000000   0.420735   0.164060   0.256676  61.006471
 13.000000   0.498747   0.069891   0.428856  85.986604
 14.000000   0.454649   0.018435   0.436214  95.945287
 15.000000   0.299236  -0.007738   0.306975 102.586070
 16.000000   0.070560  -0.020798   0.091358 129.475865
 17.000000  -0.175392  -0.027278  -0.148113 -84.447263
 18.000000  -0.378401  -0.030488  -0.347914 -91.943055
 19.000000  -0.488765  -0.032076  -0.456689 -93.437334
Maximum Absolute Error 0.511473


Input        Actual       Pred        Error       %
  0.000000   0.479462   0.469578   0.009884    2.061550
  0.052632   0.488765   0.447425   0.041340    8.458115
  0.105263   0.378401   0.384571  -0.006170    1.630557
  0.157895   0.175392   0.211022  -0.035630   20.314779
```

Reply · Email · View Thread 5.00/5 (1 vote)

MSE Calculation

ravenspoint 16-Oct-07 22:04

The calculation of mean square error looks strange to me.

Should we not divide by the number of outputs (instead of just 2 in every case) ?

I think the code in double CBackProp::mse(double *tgt) should be:

Hide Copy Code

```
double mse=0;
for(int i=0;i<lsize[numl-1];i++){
    mse+=(tgt[i]-out[numl-1][i])*(tgt[i]-out[numl-1][i]);
}
return mse/lsize[numl-1];
```

Reply · Email · View Thread

Re: MSE Calculation

brutjbro 23-Oct-07 21:25

I agree with you, "return mse/2;" should be replaced by "return mse/lsize[numl-1];".
File: BackProp.cpp, line 130.

btw It is a really good, simple and well written article.

Reply · Email · View Thread

Re: MSE Calculation

ravenspoint 23-Oct-07 22:48

Thank you for double checking my observation. It is always good to get somebody else to look at something like this, since I might well have been missing something. This is especially the case here, where the original author seems no longer to be around.

Yes, the article, and code, are a good, simple start to working with neural networks. However, the class CBackProp is really too simple to be useful by itself in serious work.

1. The undocumented scaling issues are a problem

2. The interface, which requires data to be passed in through pointers to elements of two dimensional arrays, is quite unfriendly.

3. The learning procedure in the sample application is extremely limited, and will not work with different input data.

4. There is no support for my own particular interest, time series analysis.

I am developing a wrapper class for CBackProp which addresses the above issues.

James

Reply · Email · View Thread

Re: MSE Calculation

brutjbro 25-Oct-07 3:57

Sorry, my previous statement is incorrect. MSE here is CORRECT.

I just realized, that this is not a standard MSE.
It was modified in Neural Networks theory to make it more simple to derive MSE function.
- simple example: (1/2 * (x)^2)' = 2 * 1/2 * x' = x'

Reply · Email · View Thread

Re: MSE Calculation

ravenspoint 29-Oct-07 3:16

我不明白。

意思是平方誤差如何除了平方誤差之和除以總數？

你簡單的例子中x是什麼意思？

詹姆士

回复 · 電子郵件 · 查看主題 5.00 / 5（1票）

刷新 1 2 3 下一頁»

一般 新聞 建議 問題 錯誤 答案 笑話 讚美 Rant Admin

固定鏈接 | 廣告 | 隱私 | 使用條款 | 移動
Web03-2016 | 2.8.180302.2 | 最後更新2006年3月28日

請選取語言 ▼
佈局：固定 | 流體