Lukas Steindl

All the scripts and notes i took before github got cool.

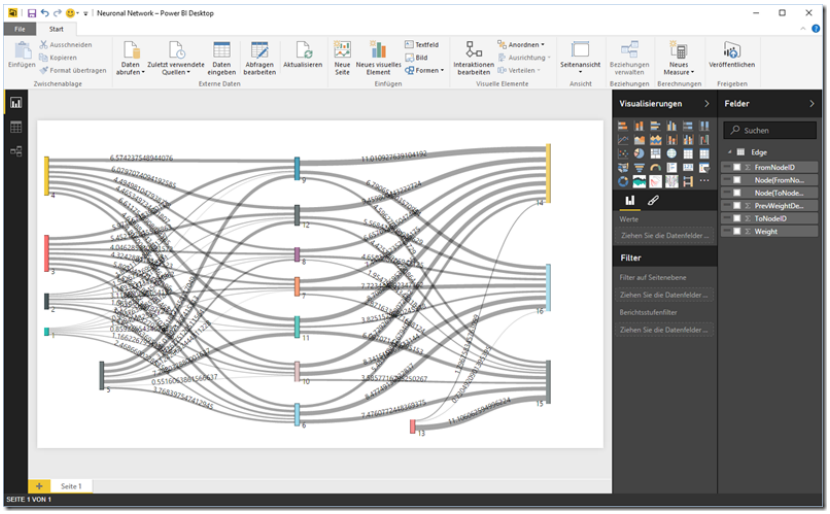
Uncategorized

Neural Networks in T-SQL

Posted On February 28, 2016

it might not be the fastest option to train a neural network but it seems to be a quite elegant approach to use T-SQL to implement the backpropagation algorithm. In order to learn and better understand the algorithm I decided to port the C# neural network example from Dr. James McCaffrey(MSR) to Sql Server. The goal of the network is to classify three different types of flowers based on four input features (sepal length, height etc.).

This is how the trained model looks like in the end:



(Its a nice sideeffect that you can easily visualize the network using Power BI when the

Recent Posts

Recent Comments

data is stored in SQL Server.)

The high level steps to train a neural network model are described briefly first:

1. Normalize the input data
2. Define the network structure (Number of Input Nodes, Number of Output Nodes are given by the problem, Number of Hidden nodes are up to you.)
3. Take a random training example and set the input and desired output state
4. Calculate the error for the training example
5. Change the weights so that the error decreases
6. Repeat Steps 3 to 5 until the error on the training data goes below a threshold or stop after a finite number of training interations

You will find the complete executable script at the end of this post. Here are some of the highlights I wanted to point out.

Doing Step 1 in T-sql is just a single line of code:

```
Insert Into NormalizedIrisData
Select (x1-mue_x1)/std_x1,(x2-mue_x2)/std_x2,(x3-mue_x3)/std_x3, (x4-mue_x4)/std_x4, class
from IrisData i cross join (
Select AVG(x1) as mue_x1, STDEVP(x1) as std_x1,AVG(x2) as mue_x2, STDEVP(x2) as std_x2,
AVG(x3) as mue_x3, STDEVP(x3) as std_x3,AVG(x4) as mue_x4,STDEVP(x4) as std_x4
from IrisData) d
```

Darron on SQL Server 2016 (CTP3) R Integration

Fausto on SQL Server 2016 (CTP3) R Integration

Sylvester on SQL Server 2016 (CTP3) R Integration

Haley on SQL Server 2016 (CTP3) R Integration

Archives

October 2019

June 2018

May 2018

April 2018

March 2018

June 2017

June 2016

May 2016

April 2016

February 2016

December 2015

November 2015

October 2015

Step 2 is simple using two tables you can encode your network structure:

```
Create Table Node (ID int primary key identity, NodeType char(2),
WeightedInputSum float, ActivationOutput float, ExpectedOutput float, Delta
float, Gradient float)
Create Table Edge (FromNodeID int references Node, ToNodeID int references
Node, [Weight] float, PrevWeightDelta float)

--3 input Nodes
Insert Into Node VALUES ('I',null,null,null,null,null),('I',null,null,null,null,null),
('I',null,null,null,null,null),('I',null,null,null,null,null),('BI',null,1,null,null,null)
--7 hidden Nodes
Insert Into Node VALUES ('H',null,null,null,null,null),
('H',null,null,null,null,null),('H',null,null,null,null,null),
('H',null,null,null,null,null),('H',null,null,null,null,null),
('H',null,null,null,null,null),('H',null,null,null,null,null),('BH',null,1,null,null,null)
--3 output nodes
Insert Into Node VALUES ('O',null,null,null,null,null),
('O',null,null,null,null,null),('O',null,null,null,null,null)
```

Step 3 just loads a random training example into the input nodes and sets the desired output at the output nodes

```
Update InputNodes set ActivationOutput = (Select x1 from TrainingData where i
= @SampleID) where InputNodes.ID = 1
Update InputNodes set ActivationOutput = (Select x2 from TrainingData where i
= @SampleID) where InputNodes.ID = 2
Update InputNodes set ActivationOutput = (Select x3 from TrainingData where i
= @SampleID) where InputNodes.ID = 3
Update InputNodes set ActivationOutput = (Select x4 from TrainingData where i
= @SampleID) where InputNodes.ID = 4
Update OutputNodes set ExpectedOutput = (Select class1 from TrainingData
where i = @SampleID) where OutputNodes.ID = 14
Update OutputNodes set ExpectedOutput = (Select class2 from TrainingData
where i = @SampleID) where OutputNodes.ID = 15
Update OutputNodes set ExpectedOutput = (Select class3 from TrainingData
where i = @SampleID) where OutputNodes.ID = 16
```

Step 4 passes the inputs through the network and calculates the error

```
Update h set h.WeightedInputSum = x.WeightedInputSum
from HiddenNodes h
join (Select ToNodeID, SUM(i.ActivationOutput*e.Weight) as
```

June 2015
July 2014
February 2014
November 2013
June 2013
April 2013

Categories

Uncategorized

Meta

Log in
Entries feed
Comments feed
WordPress.org

```
'WeightedInputSum' from InputNodes i join Edge e on i.ID = e.FromNodeID
group by ToNodeID) x on h.ID = x.ToNodeID

--calculate activation of the hidden nodes using the hyperbolic tangent activation
function:
Update HiddenNodes set ActivationOutput = dbo.tanh(WeightedInputSum)
where NodeType = 'H'

-- calculate the weighted input sum of the output nodes
Update o set o.WeightedInputSum = x.WeightedInputSum
from OutputNodes o
join (Select ToNodeID, SUM(h.ActivationOutput*e.Weight) as
'WeightedInputSum' from HiddenNodes h join Edge e on h.ID = e.FromNodeID
group by ToNodeID) x on o.ID = x.ToNodeID

--calculate activation of the output nodes using Soft Max activation function:
declare @MaxOutput float
declare @Scale float
Select @MaxOutput = Max(WeightedInputSum) from OutputNodes
Select @Scale = SUM(EXP(WeightedInputSum - (@MaxOutput))) from
OutputNodes
Update OutputNodes set ActivationOutput = EXP(WeightedInputSum-
@MaxOutput)/@Scale
--Calculate the error for this one training example:
Update OutputNodes set Delta = ExpectedOutput - ActivationOutput
```

Step 5 calculates the change of the error for the change of a weight and changes the weight so that the error gets smaller. this is the core of the backpropagation algorithm.

```
--Calculate OutputLayer Gradient = (SoftMax Derivative * Delta)
Update OutputNodes set Gradient = (1-ActivationOutput)*ActivationOutput *
Delta

--Calculate HiddenLayer Deltas (= Sum of the weighted Output Layer gradients)
Update h set Delta = x.Delta
from HiddenNodes h join (
Select e.FromNodeID, SUM(o.Gradient*e.Weight) as 'Delta'
from OutputNodes o join Edge e on o.ID = e.ToNodeID group by
e.FromNodeID) x
on h.ID = x.FromNodeID
```

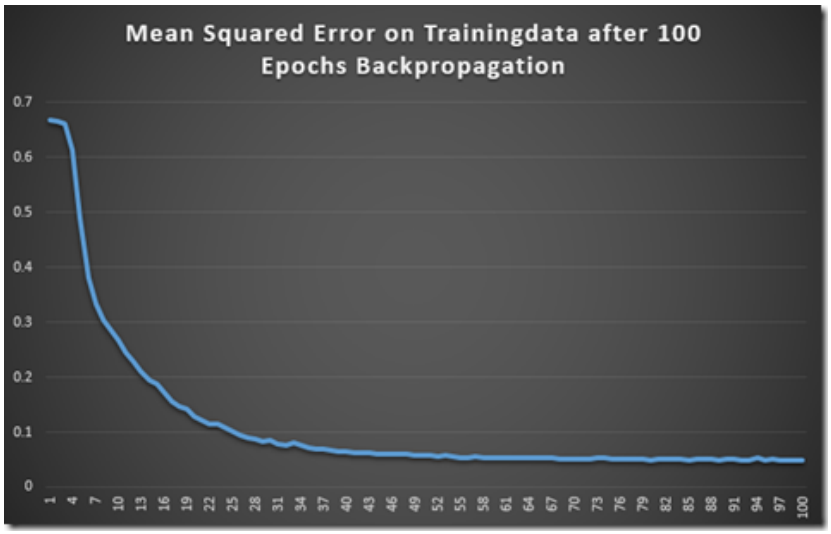
```
--calculate HiddenLayer Gradient = (tanh Derivative * Delta)
Update HiddenNodes set Gradient = (1-ActivationOutput)*
(1+ActivationOutput) * Delta

-- Update weights between input and hidden layer
Update e set PrevWeightDelta = @learningrate * h.Gradient*
i.ActivationOutput,
 [Weight] = [Weight] * 0.9999 + @learningrate * h.Gradient* i.ActivationOutput
+ @momentum * PrevWeightDelta
from HiddenNodes h join Edge e on e.ToNodeID = h.ID join InputNodes i on
e.FromNodeID = i.ID

--Update weights between hidden and output layer ->
Update e set PrevWeightDelta = @learningrate * o.Gradient*
h.ActivationOutput,
[Weight] = [Weight]* 0.9999 + @learningrate * o.Gradient* h.ActivationOutput
+ @momentum * PrevWeightDelta
from HiddenNodes h join Edge e on e.FromNodeID = h.ID join OutputNodes o
on e.ToNodeID = o.ID
```

Thats it!

Visualizing the Training Error shows a smooth learning curve:



Here is the entire Script for your convenience:

```
use master
```

```
GO
Alter Database NeuralNetwork set single_user with rollback immediate
go
Drop Database NeuralNetwork
GO
Create Database NeuralNetwork
go
use NeuralNetwork
GO



Create Table IrisData (x1 float, x2 float, x3 float, x4 float, class varchar(255))
GO

Insert Into IrisData Values (5.1,3.5,1.4,0.2,'Iris-setosa')
Insert Into IrisData Values (4.9,3.0,1.4,0.2,'Iris-setosa')
Insert Into IrisData Values (4.7,3.2,1.3,0.2,'Iris-setosa')
Insert Into IrisData Values (4.6,3.1,1.5,0.2,'Iris-setosa')
Insert Into IrisData Values (5.0,3.6,1.4,0.2,'Iris-setosa')
Insert Into IrisData Values (5.4,3.9,1.7,0.4,'Iris-setosa')
Insert Into IrisData Values (4.6,3.4,1.4,0.3,'Iris-setosa')
Insert Into IrisData Values (5.0,3.4,1.5,0.2,'Iris-setosa')
Insert Into IrisData Values (4.4,2.9,1.4,0.2,'Iris-setosa')
Insert Into IrisData Values (4.9,3.1,1.5,0.1,'Iris-setosa')
Insert Into IrisData Values (5.4,3.7,1.5,0.2,'Iris-setosa')
Insert Into IrisData Values (4.8,3.4,1.6,0.2,'Iris-setosa')
Insert Into IrisData Values (4.8,3.0,1.4,0.1,'Iris-setosa')
Insert Into IrisData Values (4.3,3.0,1.1,0.1,'Iris-setosa')
Insert Into IrisData Values (5.8,4.0,1.2,0.2,'Iris-setosa')
Insert Into IrisData Values (5.7,4.4,1.5,0.4,'Iris-setosa')
Insert Into IrisData Values (5.4,3.9,1.3,0.4,'Iris-setosa')
Insert Into IrisData Values (5.1,3.5,1.4,0.3,'Iris-setosa')
Insert Into IrisData Values (5.7,3.8,1.7,0.3,'Iris-setosa')
Insert Into IrisData Values (5.1,3.8,1.5,0.3,'Iris-setosa')
Insert Into IrisData Values (5.4,3.4,1.7,0.2,'Iris-setosa')
Insert Into IrisData Values (5.1,3.7,1.5,0.4,'Iris-setosa')
Insert Into IrisData Values (4.6,3.6,1.0,0.2,'Iris-setosa')
Insert Into IrisData Values (5.1,3.3,1.7,0.5,'Iris-setosa')
Insert Into IrisData Values (4.8,3.4,1.9,0.2,'Iris-setosa')
Insert Into IrisData Values (5.0,3.0,1.6,0.2,'Iris-setosa')
Insert Into IrisData Values (5.0,3.4,1.6,0.4,'Iris-setosa')
Insert Into IrisData Values (5.2,3.5,1.5,0.2,'Iris-setosa')
Insert Into IrisData Values (5.2,3.4,1.4,0.2,'Iris-setosa')
Insert Into IrisData Values (4.7,3.2,1.6,0.2,'Iris-setosa')
Insert Into IrisData Values (4.8,3.1,1.6,0.2,'Iris-setosa')
Insert Into IrisData Values (5.4,3.4,1.5,0.4,'Iris-setosa')
Insert Into IrisData Values (5.2,4.1,1.5,0.1,'Iris-setosa')
Insert Into IrisData Values (5.5,4.2,1.4,0.2,'Iris-setosa')
```

```
Insert Into IrisData Values (4.9,3.1,1.5,0.2,'Iris-setosa')
Insert Into IrisData Values (5.0,3.2,1.2,0.2,'Iris-setosa')
Insert Into IrisData Values (5.5,3.5,1.3,0.2,'Iris-setosa')
Insert Into IrisData Values (4.9,3.6,1.4,0.1,'Iris-setosa')
Insert Into IrisData Values (4.4,3.0,1.3,0.2,'Iris-setosa')
Insert Into IrisData Values (5.1,3.4,1.5,0.2,'Iris-setosa')
Insert Into IrisData Values (5.0,3.5,1.3,0.3,'Iris-setosa')
Insert Into IrisData Values (4.5,2.3,1.3,0.3,'Iris-setosa')
Insert Into IrisData Values (4.4,3.2,1.3,0.2,'Iris-setosa')
Insert Into IrisData Values (5.0,3.5,1.6,0.6,'Iris-setosa')
Insert Into IrisData Values (5.1,3.8,1.9,0.4,'Iris-setosa')
Insert Into IrisData Values (4.8,3.0,1.4,0.3,'Iris-setosa')
Insert Into IrisData Values (5.1,3.8,1.6,0.2,'Iris-setosa')
Insert Into IrisData Values (4.6,3.2,1.4,0.2,'Iris-setosa')
Insert Into IrisData Values (5.3,3.7,1.5,0.2,'Iris-setosa')
Insert Into IrisData Values (5.0,3.3,1.4,0.2,'Iris-setosa')
Insert Into IrisData Values (7.0,3.2,4.7,1.4,'Iris-versicolor')
Insert Into IrisData Values (6.4,3.2,4.5,1.5,'Iris-versicolor')
Insert Into IrisData Values (6.9,3.1,4.9,1.5,'Iris-versicolor')
Insert Into IrisData Values (5.5,2.3,4.0,1.3,'Iris-versicolor')
Insert Into IrisData Values (6.5,2.8,4.6,1.5,'Iris-versicolor')
Insert Into IrisData Values (5.7,2.8,4.5,1.3,'Iris-versicolor')
Insert Into IrisData Values (6.3,3.3,4.7,1.6,'Iris-versicolor')
Insert Into IrisData Values (4.9,2.4,3.3,1.0,'Iris-versicolor')
Insert Into IrisData Values (6.6,2.9,4.6,1.3,'Iris-versicolor')
Insert Into IrisData Values (5.2,2.7,3.9,1.4,'Iris-versicolor')
Insert Into IrisData Values (5.0,2.0,3.5,1.0,'Iris-versicolor')
Insert Into IrisData Values (5.9,3.0,4.2,1.5,'Iris-versicolor')
Insert Into IrisData Values (6.0,2.2,4.0,1.0,'Iris-versicolor')
Insert Into IrisData Values (6.1,2.9,4.7,1.4,'Iris-versicolor')
Insert Into IrisData Values (5.6,2.9,3.6,1.3,'Iris-versicolor')
Insert Into IrisData Values (6.7,3.1,4.4,1.4,'Iris-versicolor')
Insert Into IrisData Values (5.6,3.0,4.5,1.5,'Iris-versicolor')
Insert Into IrisData Values (5.8,2.7,4.1,1.0,'Iris-versicolor')
Insert Into IrisData Values (6.2,2.2,4.5,1.5,'Iris-versicolor')
Insert Into IrisData Values (5.6,2.5,3.9,1.1,'Iris-versicolor')
Insert Into IrisData Values (5.9,3.2,4.8,1.8,'Iris-versicolor')
Insert Into IrisData Values (6.1,2.8,4.0,1.3,'Iris-versicolor')
Insert Into IrisData Values (6.3,2.5,4.9,1.5,'Iris-versicolor')
Insert Into IrisData Values (6.1,2.8,4.7,1.2,'Iris-versicolor')
Insert Into IrisData Values (6.4,2.9,4.3,1.3,'Iris-versicolor')
Insert Into IrisData Values (6.6,3.0,4.4,1.4,'Iris-versicolor')
Insert Into IrisData Values (6.8,2.8,4.8,1.4,'Iris-versicolor')
Insert Into IrisData Values (6.7,3.0,5.0,1.7,'Iris-versicolor')
Insert Into IrisData Values (6.0,2.9,4.5,1.5,'Iris-versicolor')
Insert Into IrisData Values (5.7,2.6,3.5,1.0,'Iris-versicolor')
Insert Into IrisData Values (5.5,2.4,3.8,1.1,'Iris-versicolor')
Insert Into IrisData Values (5.5,2.4,3.7,1.0,'Iris-versicolor')
Insert Into IrisData Values (5.8,2.7,3.9,1.2,'Iris-versicolor')
```

```
Insert Into IrisData Values (6.0,2.7,5.1,1.6,'Iris-versicolor')
Insert Into IrisData Values (5.4,3.0,4.5,1.5,'Iris-versicolor')
Insert Into IrisData Values (6.0,3.4,4.5,1.6,'Iris-versicolor')
Insert Into IrisData Values (6.7,3.1,4.7,1.5,'Iris-versicolor')
Insert Into IrisData Values (6.3,2.3,4.4,1.3,'Iris-versicolor')
Insert Into IrisData Values (5.6,3.0,4.1,1.3,'Iris-versicolor')
Insert Into IrisData Values (5.5,2.5,4.0,1.3,'Iris-versicolor')
Insert Into IrisData Values (5.5,2.6,4.4,1.2,'Iris-versicolor')
Insert Into IrisData Values (6.1,3.0,4.6,1.4,'Iris-versicolor')
Insert Into IrisData Values (5.8,2.6,4.0,1.2,'Iris-versicolor')
Insert Into IrisData Values (5.0,2.3,3.3,1.0,'Iris-versicolor')
Insert Into IrisData Values (5.6,2.7,4.2,1.3,'Iris-versicolor')
Insert Into IrisData Values (5.7,3.0,4.2,1.2,'Iris-versicolor')
Insert Into IrisData Values (5.7,2.9,4.2,1.3,'Iris-versicolor')
Insert Into IrisData Values (6.2,2.9,4.3,1.3,'Iris-versicolor')
Insert Into IrisData Values (5.1,2.5,3.0,1.1,'Iris-versicolor')
Insert Into IrisData Values (5.7,2.8,4.1,1.3,'Iris-versicolor')
Insert Into IrisData Values (6.3,3.3,6.0,2.5,'Iris-virginica')
Insert Into IrisData Values (5.8,2.7,5.1,1.9,'Iris-virginica')
Insert Into IrisData Values (7.1,3.0,5.9,2.1,'Iris-virginica')
Insert Into IrisData Values (6.3,2.9,5.6,1.8,'Iris-virginica')
Insert Into IrisData Values (6.5,3.0,5.8,2.2,'Iris-virginica')
Insert Into IrisData Values (7.6,3.0,6.6,2.1,'Iris-virginica')
Insert Into IrisData Values (4.9,2.5,4.5,1.7,'Iris-virginica')
Insert Into IrisData Values (7.3,2.9,6.3,1.8,'Iris-virginica')
Insert Into IrisData Values (6.7,2.5,5.8,1.8,'Iris-virginica')
Insert Into IrisData Values (7.2,3.6,6.1,2.5,'Iris-virginica')
Insert Into IrisData Values (6.5,3.2,5.1,2.0,'Iris-virginica')
Insert Into IrisData Values (6.4,2.7,5.3,1.9,'Iris-virginica')
Insert Into IrisData Values (6.8,3.0,5.5,2.1,'Iris-virginica')
Insert Into IrisData Values (5.7,2.5,5.0,2.0,'Iris-virginica')
Insert Into IrisData Values (5.8,2.8,5.1,2.4,'Iris-virginica')
Insert Into IrisData Values (6.4,3.2,5.3,2.3,'Iris-virginica')
Insert Into IrisData Values (6.5,3.0,5.5,1.8,'Iris-virginica')
Insert Into IrisData Values (7.7,3.8,6.7,2.2,'Iris-virginica')
Insert Into IrisData Values (7.7,2.6,6.9,2.3,'Iris-virginica')
Insert Into IrisData Values (6.0,2.2,5.0,1.5,'Iris-virginica')
Insert Into IrisData Values (6.9,3.2,5.7,2.3,'Iris-virginica')
Insert Into IrisData Values (5.6,2.8,4.9,2.0,'Iris-virginica')
Insert Into IrisData Values (7.7,2.8,6.7,2.0,'Iris-virginica')
Insert Into IrisData Values (6.3,2.7,4.9,1.8,'Iris-virginica')
Insert Into IrisData Values (6.7,3.3,5.7,2.1,'Iris-virginica')
Insert Into IrisData Values (7.2,3.2,6.0,1.8,'Iris-virginica')
Insert Into IrisData Values (6.2,2.8,4.8,1.8,'Iris-virginica')
Insert Into IrisData Values (6.1,3.0,4.9,1.8,'Iris-virginica')
Insert Into IrisData Values (6.4,2.8,5.6,2.1,'Iris-virginica')
Insert Into IrisData Values (7.2,3.0,5.8,1.6,'Iris-virginica')
Insert Into IrisData Values (7.4,2.8,6.1,1.9,'Iris-virginica')
Insert Into IrisData Values (7.9,3.8,6.4,2.0,'Iris-virginica')
```

```
Insert Into IrisData Values (6.4,2.8,5.6,2.2,'Iris-virginica')
Insert Into IrisData Values (6.3,2.8,5.1,1.5,'Iris-virginica')
Insert Into IrisData Values (6.1,2.6,5.6,1.4,'Iris-virginica')
Insert Into IrisData Values (7.7,3.0,6.1,2.3,'Iris-virginica')
Insert Into IrisData Values (6.3,3.4,5.6,2.4,'Iris-virginica')
Insert Into IrisData Values (6.4,3.1,5.5,1.8,'Iris-virginica')
Insert Into IrisData Values (6.0,3.0,4.8,1.8,'Iris-virginica')
Insert Into IrisData Values (6.9,3.1,5.4,2.1,'Iris-virginica')
Insert Into IrisData Values (6.7,3.1,5.6,2.4,'Iris-virginica')
Insert Into IrisData Values (6.9,3.1,5.1,2.3,'Iris-virginica')
Insert Into IrisData Values (5.8,2.7,5.1,1.9,'Iris-virginica')
Insert Into IrisData Values (6.8,3.2,5.9,2.3,'Iris-virginica')
Insert Into IrisData Values (6.7,3.3,5.7,2.5,'Iris-virginica')
Insert Into IrisData Values (6.7,3.0,5.2,2.3,'Iris-virginica')
Insert Into IrisData Values (6.3,2.5,5.0,1.9,'Iris-virginica')
Insert Into IrisData Values (6.5,3.0,5.2,2.0,'Iris-virginica')
Insert Into IrisData Values (6.2,3.4,5.4,2.3,'Iris-virginica')
Insert Into IrisData Values (5.9,3.0,5.1,1.8,'Iris-virginica')

Create Table NormalizedIrisData (x1 float, x2 float, x3 float, x4 float, class
varchar(255))

Insert Into NormalizedIrisData
Select (x1–mue_x1)/std_x1,(x2–mue_x2)/std_x2,(x3–mue_x3)/std_x3, (x4–
mue_x4)/std_x4, class
from IrisData i cross join (
Select AVG(x1) as mue_x1, STDEVP(x1) as std_x1,AVG(x2) as mue_x2,
STDEVP(x2) as std_x2,
AVG(x3) as mue_x3, STDEVP(x3) as std_x3,AVG(x4) as mue_x4,STDEVP(x4)
as std_x4
from IrisData) d

–Select * from NormalizedIrisData


Create Table TrainingData (i int primary key identity, x1 float, x2 float, x3 float,
x4 float, class1 bit,class2 bit,class3 bit)
Create Table ValidationData (i int primary key identity, x1 float, x2 float, x3
float, x4 float, class1 bit,class2 bit,class3 bit)


Truncate Table TrainingData
Truncate Table ValidationData

Insert into TrainingData
Select x1,x2,x3,x4,CASE class WHEN ('Iris-virginica') THEN 1 ELSE 0
END,CASE class WHEN ('Iris-versicolor') THEN 1 ELSE 0 END,CASE class
WHEN ('Iris-setosa') THEN 1 ELSE 0 END
from (
```

```
Select ROW_NUMBER() over (order by class)%10 as 'Bucket' ,* from
NormalizedIrisData)x where Bucket < 7


Insert into ValidationData
Select x1,x2,x3,x4,CASE class WHEN ('Iris-virginica') THEN 1 ELSE 0
END,CASE class WHEN ('Iris-versicolor') THEN 1 ELSE 0 END,CASE class
WHEN ('Iris-setosa') THEN 1 ELSE 0 END
from (
Select ROW_NUMBER() over (order by class)%10 as 'Bucket' ,* from
NormalizedIrisData)x where Bucket >= 7

use NeuralNetwork
go

–helperfunction as sql doesn`t offer a hyperbolic tangent function
Create Function tanh (@x float) Returns float
as
begin
    — Declare the return variable here
    declare @Result float
    — Add the T-SQL statements to compute the return value here
    set @Result =  (exp(@x) – exp(-@x)) / (exp(@x) + exp(-@x))
    — Return the result of the function
    return @Result
end
go
Create Table Error (MSE float)
–structure the neural network:
Create Table Node (ID int primary key identity, NodeType char(2),
WeightedInputSum float, ActivationOutput float, ExpectedOutput float, Delta
float, Gradient float)
Create Table Edge (FromNodeID int references Node, ToNodeID int references
Node, [Weight] float, PrevWeightDelta float)

–3 input Nodes
Insert Into Node VALUES ('I',null,null,null,null,null),
('I',null,null,null,null,null),('I',null,null,null,null,null),
('I',null,null,null,null,null),('BI',null,1,null,null,null)
–7 hidden Nodes
Insert Into Node VALUES ('H',null,null,null,null,null),
('H',null,null,null,null,null),('H',null,null,null,null,null),
('H',null,null,null,null,null),('H',null,null,null,null,null),
('H',null,null,null,null,null),('H',null,null,null,null,null),
('BH',null,1,null,null,null)
–3 output nodes
Insert Into Node VALUES ('O',null,null,null,null,null),
('O',null,null,null,null,null),('O',null,null,null,null,null)
```

```
go
Create View HiddenNodes as Select * from Node where NodeType = 'H' or
NodeType = 'BH'
go
Create View InputNodes as Select * from Node where NodeType = 'I'  or
NodeType = 'BI'
go
Create View OutputNodes as Select * from Node where NodeType = 'O'
go

–Initialize Weights and Biases
Insert Into Edge
Select i.ID,h.ID,0.02*RAND(CHECKSUM(NEWID()))-0.01,0 from InputNodes
i cross join HiddenNodes h
where h.NodeType != 'BH' — no input weight into the hidden bias fake neuron

Insert Into Edge
Select h.ID,o.ID,0.02*RAND(CHECKSUM(NEWID()))-0.01,0 from
HiddenNodes h cross join OutputNodes o


–calculate forward pass

go
Create Procedure CalculateErrorForSample (@SampleID int)
as
begin
    — take a training example and save it as activation of the input neurons and
the labels as expected output of the output neurones
    Update InputNodes set ActivationOutput = (Select x1 from TrainingData
where i = @SampleID) where InputNodes.ID = 1
    Update InputNodes set ActivationOutput = (Select x2 from TrainingData
where i = @SampleID) where InputNodes.ID = 2
    Update InputNodes set ActivationOutput = (Select x3 from TrainingData
where i = @SampleID) where InputNodes.ID = 3
    Update InputNodes set ActivationOutput = (Select x4 from TrainingData
where i = @SampleID) where InputNodes.ID = 4
    Update OutputNodes set ExpectedOutput = (Select class1 from
TrainingData where i = @SampleID) where OutputNodes.ID = 14
    Update OutputNodes set ExpectedOutput = (Select class2 from
TrainingData where i = @SampleID) where OutputNodes.ID = 15
    Update OutputNodes set ExpectedOutput = (Select class3 from
TrainingData where i = @SampleID) where OutputNodes.ID = 16

    — calculate the weighted input sum of the hidden nodes
Update h set h.WeightedInputSum = x.WeightedInputSum
from HiddenNodes h
join (Select ToNodeID, SUM(i.ActivationOutput*e.Weight) as
'WeightedInputSum' from InputNodes i join Edge e on i.ID = e.FromNodeID
```

```
group by ToNodeID) x on h.ID = x.ToNodeID

–calculate activation of the hidden nodes using the hyperbolic tangent activation
function:
Update HiddenNodes set ActivationOutput = dbo.tanh(WeightedInputSum)
where NodeType = 'H'

— calculate the weighted input sum of the output nodes
Update o set o.WeightedInputSum = x.WeightedInputSum
from OutputNodes o
join (Select ToNodeID, SUM(h.ActivationOutput*e.Weight) as
'WeightedInputSum' from HiddenNodes h join Edge e on h.ID = e.FromNodeID
group by ToNodeID) x on o.ID = x.ToNodeID

–calculate activation of the output nodes using Soft Max activation function:
declare @MaxOutput float
declare @Scale float
Select @MaxOutput = Max(WeightedInputSum) from OutputNodes
Select @Scale = SUM(EXP(WeightedInputSum – (@MaxOutput))) from
OutputNodes
Update OutputNodes set ActivationOutput = EXP(WeightedInputSum–
@MaxOutput)/@Scale
–Calculate the error for this one training example:
Update OutputNodes set Delta = ExpectedOutput – ActivationOutput
end


go
Create Procedure UpdateWeights (@learningrate float, @momentum float)
as
begin
–Calculate OutputLayer Gradient = (SoftMax Derivative * Delta)
Update OutputNodes set Gradient = (1–ActivationOutput)*ActivationOutput *
Delta

–Calculate HiddenLayer Deltas (= Sum of the weighted Output Layer gradients)
Update h set Delta = x.Delta
from HiddenNodes h join (
Select e.FromNodeID, SUM(o.Gradient*e.Weight) as 'Delta'
from OutputNodes o join Edge e on o.ID = e.ToNodeID group by
e.FromNodeID) x
on h.ID = x.FromNodeID

–calculate HiddenLayer Gradient = (tanh Derivative * Delta)
Update HiddenNodes set Gradient = (1–ActivationOutput)*
(1+ActivationOutput) * Delta

— Update weights between input and hidden layer
Update e set PrevWeightDelta = @learningrate * h.Gradient*
```

```
i.ActivationOutput,
 [Weight] = [Weight] * 0.9999 + @learningrate * h.Gradient* i.ActivationOutput
+ @momentum * PrevWeightDelta
from HiddenNodes h join Edge e on e.ToNodeID = h.ID join InputNodes i on
e.FromNodeID = i.ID

–Update weights between hidden and output layer ->
Update e set PrevWeightDelta = @learningrate * o.Gradient*
h.ActivationOutput,
[Weight] = [Weight]* 0.9999 + @learningrate * o.Gradient* h.ActivationOutput
+ @momentum * PrevWeightDelta
from HiddenNodes h join Edge e on e.FromNodeID = h.ID join OutputNodes o
on e.ToNodeID = o.ID
end

go

–Calculate Training Set Error
Create Procedure CalculateMSEforTrainingData
as
begin
    — cannot be a function as it changes a table 🙁 not nice i know!
    set nocount on
    declare @id int = 1
    declare @squarederror decimal(32,5) = 0
    declare @numberofobservations int
    Select @numberofobservations = COUNT(*) from TrainingData
    while @id <= @numberofobservations
    begin
        exec CalculateErrorForSample @id
        Select @squarederror = @squarederror + SUM(delta*delta) from
OutputNodes
        set @id += 1
    end
    Select @squarederror/@numberofobservations
end
go

–stochastic gradient descent
Create Procedure Train (@learningrate float, @momentum float)
as
begin
set nocount on
declare @numberofsamples int = 105
declare @x int = 0
while @x < @numberofsamples
begin
    declare @trainingsample int
    Select @trainingsample = CAST(RAND()*105 as int)+1
```

```
        –print @trainingsample
        exec CalculateErrorForSample @trainingsample
        exec UpdateWeights @learningrate, @momentum
        set @x += 1
    end
    end
    go

    Create Procedure Learn
    as
    begin
    set nocount on
        declare @error float
        declare @epoch int = 0
            while @epoch < 20
            begin
                Insert Into Error
                exec CalculateMSEforTrainingData
                Select  @error = MSE from Error
                –Training while MSE > Threshold
                if  (@error < 0.02)
                begin
                  break
                end
                exec Train 0.05, 0.01
                print @error
                –calculate the new error
                Truncate Table Error
                Insert Into Error
                exec CalculateMSEforTrainingData
                Select  @error = MSE from Error
                set @epoch +=1
            end
    end
    go


    –this will start the backpropagation algorithm until the errorfunction converges.
    –you can also run a single step of back propagation calling the next two lines!
    exec Learn

    –exec CalculateMSEforTrainingData
    –exec Train 0.05, 0.01


    –Select * from InputNodes
    –Select * from HiddenNodes
    –Select * from OutputNodes
    –Select * from Edge
```

```
Use NeuralNetwork
–VALIDATION:

GO
Create Procedure CalculateErrorForValidationSample (@SampleID int)
as
begin
    Update InputNodes set ActivationOutput = (Select x1 from ValidationData
where i = @SampleID) where InputNodes.ID = 1
    Update InputNodes set ActivationOutput = (Select x2 from ValidationData
where i = @SampleID) where InputNodes.ID = 2
    Update InputNodes set ActivationOutput = (Select x3 from ValidationData
where i = @SampleID) where InputNodes.ID = 3
    Update InputNodes set ActivationOutput = (Select x4 from ValidationData
where i = @SampleID) where InputNodes.ID = 4
    Update OutputNodes set ExpectedOutput = (Select class1 from
ValidationData where i = @SampleID) where OutputNodes.ID = 14
    Update OutputNodes set ExpectedOutput = (Select class2 from
ValidationData where i = @SampleID) where OutputNodes.ID = 15
    Update OutputNodes set ExpectedOutput = (Select class3 from
ValidationData where i = @SampleID) where OutputNodes.ID = 16

    Update h set h.WeightedInputSum = x.WeightedInputSum
    from HiddenNodes h
    join (Select ToNodeID, SUM(i.ActivationOutput*e.Weight) as
'WeightedInputSum' from InputNodes i join Edge e on i.ID = e.FromNodeID
    group by ToNodeID) x on h.ID = x.ToNodeID

    Update HiddenNodes set ActivationOutput =
dbo.TANH(WeightedInputSum) where NodeType = 'H'

    Update o set o.WeightedInputSum = x.WeightedInputSum
    from OutputNodes o
    join (Select ToNodeID, SUM(h.ActivationOutput*e.Weight) as
'WeightedInputSum' from HiddenNodes h join Edge e on h.ID = e.FromNodeID
    group by ToNodeID) x on o.ID = x.ToNodeID

    declare @MaxOutput float
    declare @Scale float
    Select @MaxOutput = Max(WeightedInputSum) from OutputNodes
    Select @Scale = SUM(EXP(WeightedInputSum – (@MaxOutput))) from
OutputNodes
    Update OutputNodes set ActivationOutput = EXP(WeightedInputSum–
@MaxOutput)/@Scale
    Update OutputNodes set Delta = ExpectedOutput – ActivationOutput
END
```

```
GO



CREATE Procedure CalculateMSEforValidationData
AS
BEGIN
	-- Declare the return variable here
DECLARE @Result float
set nocount on
declare @id int = 1
declare @squarederror decimal(32,5) = 0
declare @numberofobservations int
Select @numberofobservations = COUNT(*) from ValidationData
while @id <= @numberofobservations
begin
	exec CalculateErrorForValidationSample @id
	Select @squarederror = @squarederror + SUM(delta*delta) from
OutputNodes
	set @id += 1
end

Select @squarederror/@numberofobservations
END
GO



exec CalculateMSEforValidationData
```

Add a Comment

Comment

Name

Email

Website

Add Comment