



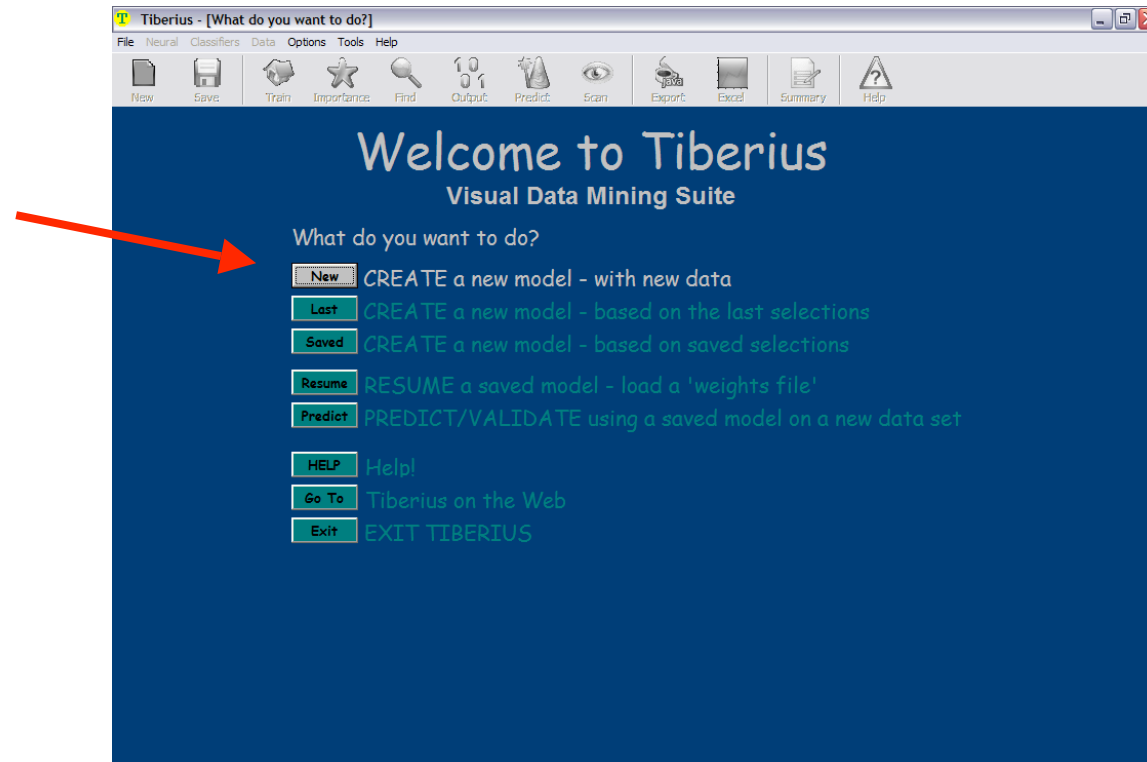
Neural Network Learning

Here we look at the practical considerations for constructing artificial neural networks:

- (1) All training data has to be numerical, even categorical data has to be mapped into numerical values (sub-symbolic learner)
- (2) It is customary to normalize the data into the interval $[0,1]$, or something close to it.
- (3) We need to pick a topology for the network
 - Do we need a hidden layer?
 - If so, how many nodes in the hidden layer?
- (4) We need to pick a learning rate.
- (5) We need to pick a convergence criterion that will tell us whether we learned the concept/training examples successfully.



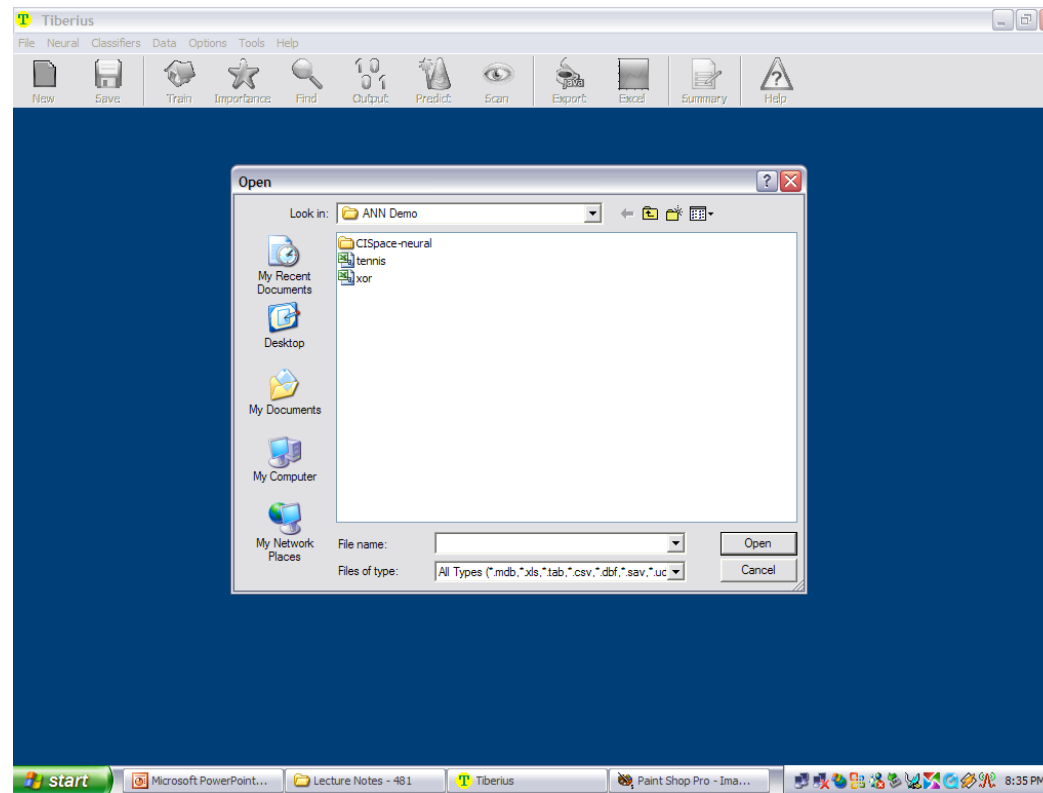
Tiberius Data Mining Suite



Create a new neural network.



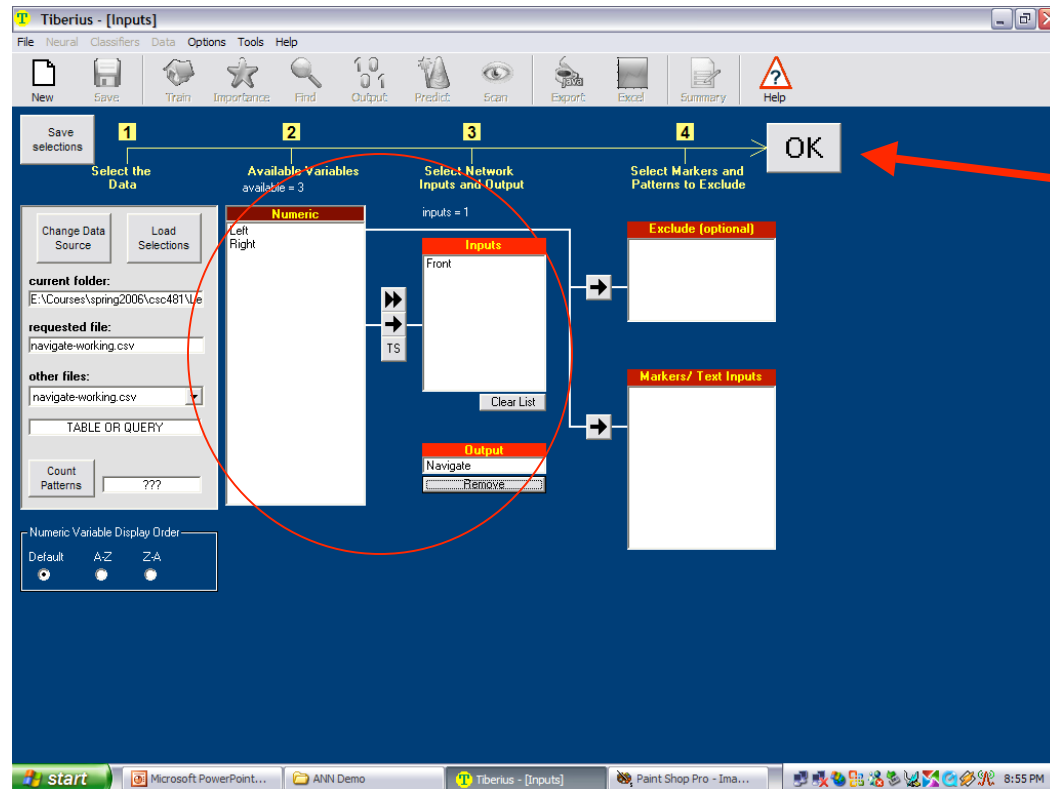
Tiberius Data Mining Suite



Select a CSV file for training. NOTE: ignore the tip dialog box.



Tiberius Data Mining Suite



Hit OK when you are done.

Select the neural network inputs and output; the inputs are given by your independent attributes, the output is your target attribute.



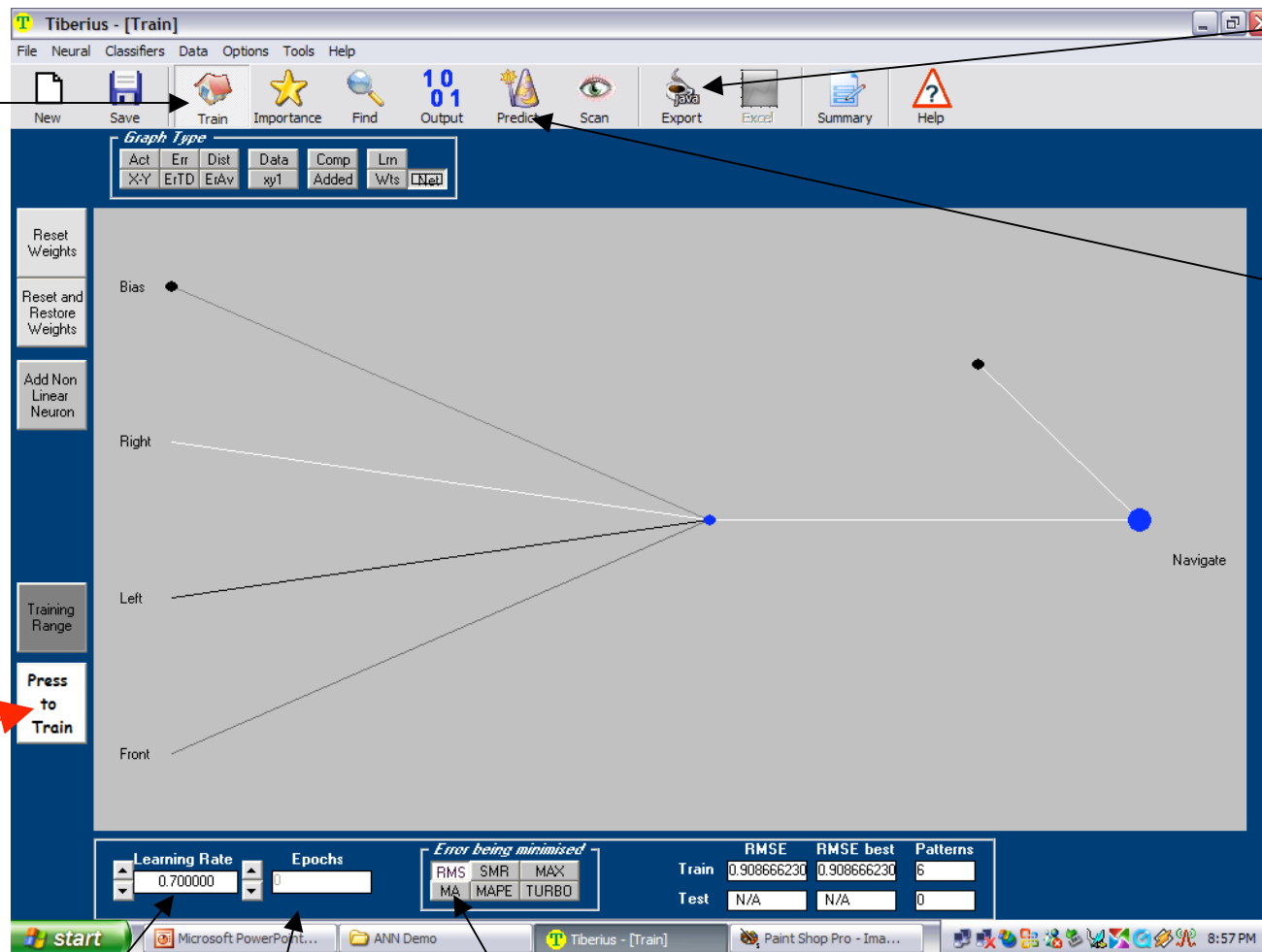
Tiberius Data Mining Suite

Training Window

Generate Code Window

Check Results Window

Start Training



Learning Rate

Training Iterations

Error

Training Status



Tiberius Data Mining Suite

The screenshot displays the Tiberius Data Mining Suite interface. The main window is titled "Tiberius - [Model Results and New Predictions]". It features a menu bar (File, Neural, Classifiers, Data, Options, Tools, Help) and a toolbar with icons for New, Save, Train, Importance, Find, Output, Predict, Scan, Export, Excel, Summary, and Help. The central area is divided into two main sections: the "Results Table" and the "Query Table".

The "Results Table" is a table with columns: Pattern No., Input 1, Input 2, Input 3, Actual, Model, Error, and Marker. It contains 6 rows of data. A red circle highlights the "Actual" and "Model" columns for the first row.

Pattern No.	Input 1	Input 2	Input 3	Actual	Model	Error	Marker
1	0	0	0	2	2.0003	0.0003	
2	1	0	0	0	-0.0002	-0.0002	
3	1	1	0	1	1.0001	0.0001	
4	1	0	1	0	0.0002	0.0002	
5	0	1	1	2	2.0000	0.0000	
6	0	0	1	2	1.9997	-0.0003	

The "Query Table" is a table with columns: Front, Left, Right, and Navigate. It contains 4 rows of data. The "current val" row is highlighted in red.

	Front	Left	Right	Navigate
max val	1	1	1	2
min val	0	0	0	0
current val	0	0	0	2.0003

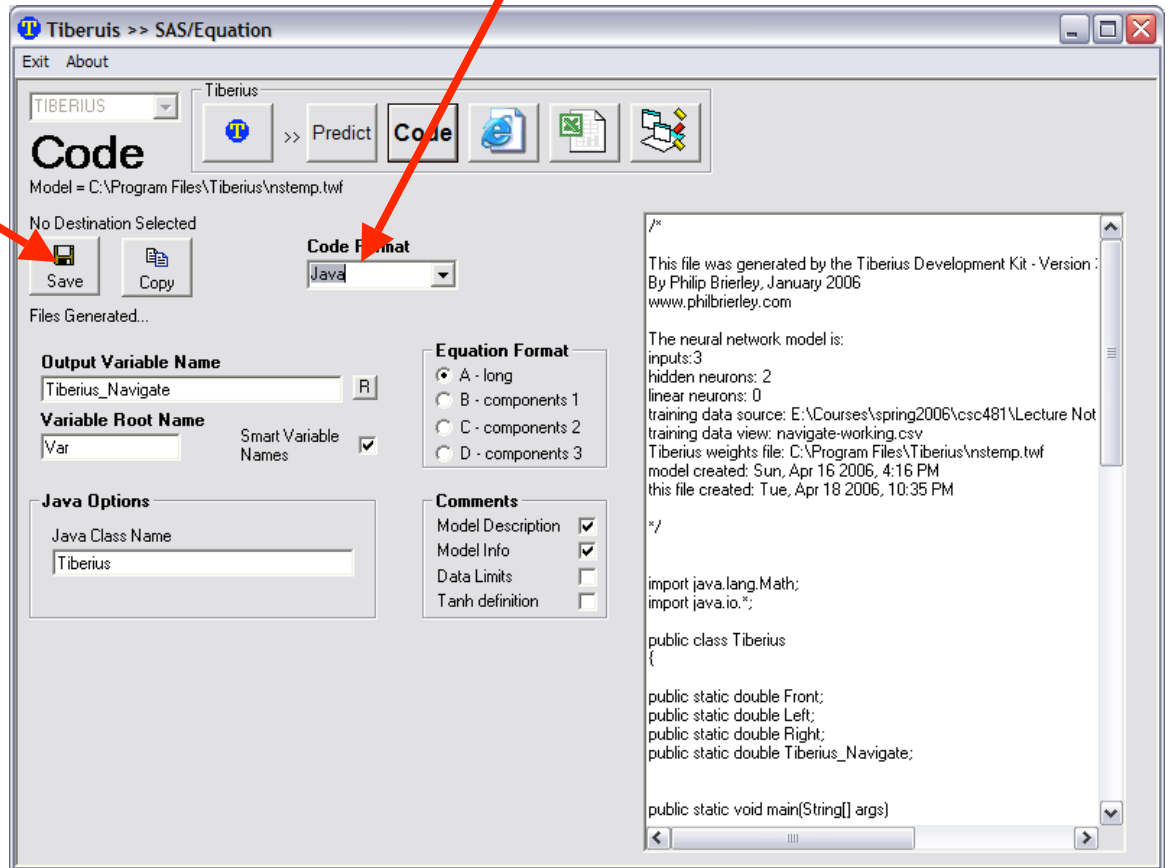
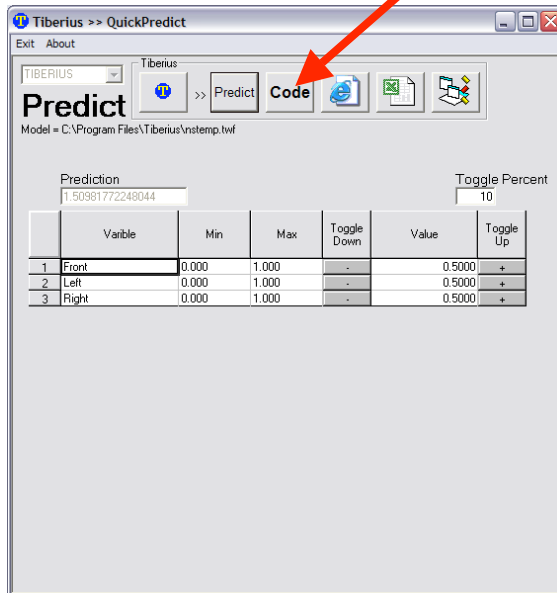
Below the Query Table, there is a "Modelled Navigate" section with a value of 2.0003 displayed in a green box.

On the right side of the interface, there is a legend for pattern types: Training Pattern (white square), Testing Pattern (blue square), Excluded Pattern (green square), and Missing Data (orange square). Below the legend are two radio buttons: "Include/Exclude" (selected) and "To Query Table".

Validating the model: Comparing the target values computed by the neural network with the target attribute values given in the training data.



Tiberius Data Mining Suite





Tiberius Data Mining Suite

Done!



The Generated Java Code

```
public class Tiberius
{

    public static double Left;
    public static double Right;
    public static double Tiberius_Navigate;

    public static void calcNet()
    {
        Tiberius_Navigate =
        (((+0.253747737130866 *
        (((Front * (-3.34737363256178)) + 1.67368681628089)
        + ((Left * (-0.23827360982432)) + 0.11913680491216)
        + ((Right * (-1.81516094934551E-02)) + 9.07580474672754E-03)
        - 0.134088362248703)
        +0.646454845854852 *
        (myTanh(((Front * (-2.84977887744496)) + 1.42488943872248)
        + ((Left * (2.37752505197836)) - 1.18876252598918)
        + ((Right * (3.39382279056324E-02)) - 1.69691139528162E-02)
        + 1.2472225227062
        )))
        -4.03005224703632E-03)/2) + 0.5) * 2);
    }

    public static double myTanh(double x)
    {
        if (x > 20)
            return 1;
        else if (x < -20)
            return -1;
        else
        {
            double a = Math.exp(x);
            double b = Math.exp(-x);
            return (a-b)/(a+b);
        }
    }
}
```



The NN Adapter

// Tiberius Neural Network Adapter for QLearner

```
class DecisionModule
```

```
{
```

```
    // attribute values
```

```
    // values for Left
```

```
    public static final int lclear = 0;
```

```
    public static final int lblocked = 1;
```

```
    // values for Right
```

```
    public static final int rclear = 0;
```

```
    public static final int rblocked = 1;
```

```
    // values for Front
```

```
    public static final int fclear = 0;
```

```
    public static final int fblocked = 1;
```

```
    // values for Navigate
```

```
    public static final int left = 0;
```

```
    public static final int right = 1;
```

```
    public static final int walk = 2;
```

```
    private static final double epsilon = 0.001;
```

// decision function

```
public static int decide(int Left,int Right,int Front) {
```

```
    Tiberius.Left = Left;
```

```
    Tiberius.Right = Right;
```

```
    Tiberius.Front = Front;
```

```
    Tiberius.calcNet();
```

```
    // NOTE: the following code is highly dependent on the
```

```
    // value assignment for Navigate
```

```
    double retVal = Tiberius.Tiberius_Navigate;
```

```
    if (retVal > 2 - epsilon)
```

```
        return walk;
```

```
    else if (retVal > 1 - epsilon)
```

```
        return right;
```

```
    else if (retVal > 0 - epsilon)
```

```
        return left;
```

```
    else
```

```
        return -1;
```

```
    }
```

```
}
```