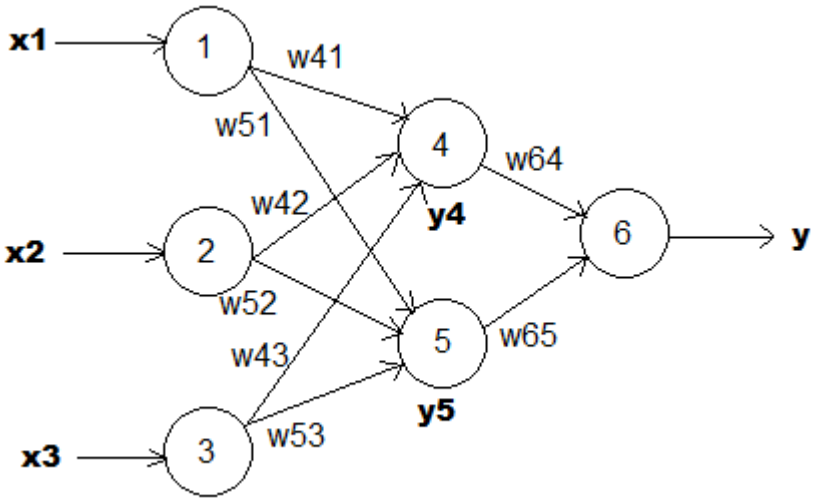
طريقة الانتشار الخلفي للخطأ

Error Back Propagation

بفرض لدينا في الشبكة العصبونية MLP المبينة بالشكل :



جميع العصبونات لها نفس تابع التفعيل **sigmoid** ،وتابع الخطأ المستخدم يُعطى بالعلاقة $E = \frac{1}{2}(d-y)^2$

حيث:

d : الخرج المرغوب (خرج المعلم).

y : الخرج الفعلي (خرج الشبكة).

نريد تدريب الشبكة العصبونية باستخدام طريقة الانتشار الخلفي للخطأ Error Back Propagation حيث قُمنا في البداية بإعطاء أوزان الشبكة القيم العشوائية التالية مع إهمال أوزان الانحياز:

$w_{41}=0.2, w_{51}=0.7,\ w_{42}=-0.1, w_{52}=-1.2,\ w_{43}=0.4, w_{53}=1.2,\ w_{64}=1.1, w_{65}=0.1$

ثم قمنا بتغذية الشبكة بعينة التدريب التالية : **[x=(10,30,20) ,d=1]**

المطلوب :

1- اكتب علاقة خرج كل عصبون من عصبونات الشبكة مع حساب قيمته العددية.

2- استنتج مقدار تغيُّر الوزن w_{65} واحسب قيمة هذا الوزن الجديدة ؛علماً أنَّ :

معدل التدريب (Learning Rate) $\eta=0.1$

3- استنتج العلاقة التي تعطي مقدار تغيُّر الوزن w_{42}

الحل :

الطلب الأول :

تتكون طريقة الانتشار الخلفي للخطأ من الخطوات التالية :

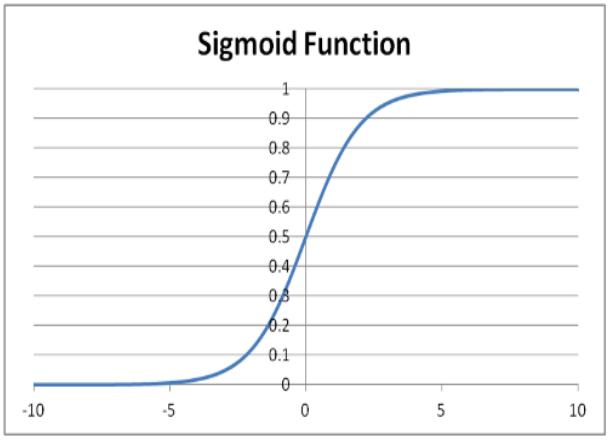
1- إعطاء أوزان عشوائية للوصلات بين عصبونات الشبكة .

2- تغذية الشبكة بإحدى المعطيات المعدّة للتدريب .

3- تطبيق عملية الانتشار الأمامي لتحديد خرج الشبكة العصبونية .

4- مقارنة الخرج الفعلي مع الخرج المطلوب ،وتحديد قيمة الخطأ .

5- التراجع بالخطأ عبر الشبكة وتصحيح الأوزان في الاتجاه الذي يضمن تصغير قيمة الخطأ ومن هنا جاءت تسميتها بالانتشار الخلفي للخطأ .



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

$$y_4 = \frac{1}{1 + e^{-(x_1.w_{41} + x_2.w_{42} + x_3.w_{43})}}$$

$$= \frac{1}{1 + e^{-(10\times0.2 - 30\times0.1 + 20\times0.4)}} = \frac{1}{1 + e^{-7}} = 0.999$$

وبنفس الطريقة :

$$y_5 = \frac{1}{1 + e^{-(x_1 .w_{51} + x_2 .w_{52} + x_3 .w_{53})}}$$

$$= \frac{1}{1 + e^{-(10\times0.7 - 30\times1.2 + 20\times1.2)}} = \frac{1}{1 + e^{+5}} = 0.0066$$

$$y = \frac{1}{1 + e^{-(y_4 .w_{64} + y_5 .w_{65})}}$$

$$= \frac{1}{1 + e^{-(0.999\times1.1 + 0.0066\times0.1)}} = \frac{1}{1 + e^{-1.0996}} = 0.750$$

الطلب الثاني :

الآن نقوم بتعديل أوزان طبقة الخرج وبحسب خوارزمية الانحدار التدريجي " gradient descent "

$$\Delta w_{65} = -\eta \frac{\partial E}{\partial w_{65}}$$ يكون :

$$\frac{\partial E}{\partial w_{65}} = \frac{\partial \frac{1}{2}(d - y)^2}{\partial y} \times \frac{\partial y}{\partial w_{65}} = -(d - y) \times \frac{\partial y}{\partial w_{65}}$$

أي لإيجاد مشتق تابع الخطأ بالنسبة للأوزان فإنّنا بحاجة لإيجاد مشتق تابع الخرج y بالنسبة للأوزان ،وبالتالي فإنَّ تابع الخرج يجب أن يكون قابلاً للاشتقاق مثل تابع ال sigmoid.

Chain Rule $\Longrightarrow$ $\frac{\partial y}{\partial w_{65}} = \frac{\partial y}{\partial s} \times \frac{\partial s}{\partial w_{65}}$

حيث :

$$s = \sum_{i=4}^{5} w_{6i} \cdot y_i$$

$$\frac{\partial y}{\partial s} = y(1-y)$$

$$\frac{\partial s}{\partial w_{65}} = \frac{\partial}{\partial w_{65}} \sum_{i=4}^{5} w_{6i} \cdot y_i = \frac{\partial}{\partial w_{65}} (w_{64} \cdot y_4 + w_{65} \cdot y_5) = y_5$$

وبالتعويض نجد :

$$\frac{\partial E}{\partial w_{65}} = -(d-y).y(1-y).y_5$$

$$\frac{\partial E}{\partial w_{65}} = -\delta . y_5 \quad \Leftarrow \quad (d-y).y(1-y) = \delta \text{ حيث :}$$

$$\Delta w_{65} = -\eta \frac{\partial E}{\partial w_{65}} = \eta . \delta . y_5$$

$$\Delta w_{65} = \eta . \delta . y_5 = 0.1 \times \delta \times 0.0066$$

لكن :

$$\delta = (d-y).y(1-y) = (1-0.750) \times (1-0.750) \times 0.750 = 0.0468$$

$$\Delta w_{65} = 3.093 \times 10^{-5}$$

$$w_{65\,new} = w_{65\,old} + \Delta w_{65} = 0.1 + 3.093 \times 10^{-5} = 0.10003$$

نلاحظ أنَّ قيمة الوزن ازدادت بالتالي فإنَّ قيمة الجمع الموزون تزداد ،ومنه فإنَّ الخرج سيقترب من الخرج المطلوب .

الطلب الثالث

$$\Delta w_{42} = -\eta \frac{\partial E}{\partial w_{42}}$$

$$\text{Chain Rule} \Longrightarrow \frac{\partial E}{\partial w_{42}} = \frac{\partial E}{\partial y} \times \frac{\partial y}{\partial w_{42}} = \frac{dE}{dy} \times \frac{dy}{dy_4} \times \frac{\partial y_4}{\partial w_{42}}$$

حيث :

$$\frac{dE}{dy} = \frac{d}{dy}(d-y)^2 = -(d-y)$$

$$\frac{dy}{dy_4} = \frac{\partial}{\partial y_4}\left[\frac{1}{1+e^{-(w_{64}.y_4 + w_{65}.y_5)}}\right] = \frac{w_{64} \times e^{-(w_{64}.y_4 + w_{65}.y_5)}}{(1+e^{-(w_{64}.y_4 + w_{65}.y_5)})^2}$$

$$= w_{64}.y(1-y)$$

$$\frac{\partial y_4}{\partial w_{42}} = \frac{\partial}{\partial w_{42}}\left[\frac{1}{1+e^{-(w_{41}.x_1 + w_{42}.x_2 + w_{43}.x_3)}}\right]$$

$$= \frac{x_2 . e^{-(w_{41}.x_1 + w_{42}.x_2 + w_{43}.x_3)}}{1+e^{-(w_{41}.x_1 + w_{42}.x_2 + w_{43}.x_3)}} = x_2 . y_4(1-y_4)$$

ومنه نجد أنَّ :

$$\frac{\partial E}{\partial w_{42}} = -(d-y).w_{64}.y(1-y).x_2.y_4(1-y_4)$$

$$\Delta w_{42} = \eta . x_2 . y_4(1-y_4).w_{64}.\underbrace{y(1-y)(d-y)}_{\delta}$$

$$= \eta . x_2 . \underbrace{y_4(1-y_4).w_{64}.\delta}_{\delta_4}$$

$$\Delta_{42} = \eta . x_2 . \delta_4$$

بناء الشبكات العصبية باستخدام صندوق الأدوات Neural Networks Toolbox

مثال 1 (بوابة XOR)

لتصميم شبكة عصبونية ذات تغذية أمامية تقوم بوظيفة محددة يجب اتباع الخطوات الثلاث التالية:

1. إنشاء الشبكة.
2. تدريب الشبكة على بيانات دخل معلومة.
3. اختبار خرج الشبكة عند بيانات دخل جديدة.

- لشرح الخطوات سنبدأ بأبسط **مثال**، بوابة XOR المنطقية التي تتألف من مدخلين ومخرج وحيد، وهي تمثل الطرح الثنائي بالقيمة المطلقة.

XOR

I_1	I_2	Out
0	0	0
0	1	1
1	0	1
1	1	0

1- إنشاء الشبكة:

في البداية وقبل إنشاء الشبكة يجب تعريف المداخل والمخارج الخاصة بها:

```
p=[0 0 1 1 ; 0 1 0 1];
```

```
T=[0 1 1 0];
```

ثم ننشئ الشبكة باستخدام التابع newff

```
net = newff( minmax(p) , [4 1] , {'logsig','purelin'} , 'trainlm' );
```

- التابع newff له عدة بارامترات، الأول minmax(p) يمثل أدنى وأعلى قيمة في كل دخل.
- البارامتر التالي [4 1] يمثل عدد طبقات الشبكة (المخفية والخرج)، وعدد العصبونات في كل طبقة منها (طبقة الدخل لا تقوم بعمليات معالجة)، يخصص الرقم الأخير دوماً لعدد عصبونات طبقة الخرج، في مثالنا الحالي لدينا طبقة مخفية عدد عصبوناتها 4 وطبقة الخرج عدد عصبوناتها 1.
- البارامتر التالي {'logsig','purelin'} يمثل تابع التفعيل لعصبونات كل طبقة من الطبقات، يجب أن يكون عدد التوابع بنفس عدد الطبقات بكل تأكيد.

- البارامتر الأخير 'trainlm' يمثل نمط تدريب الشبكة.

2- تدريب الشبكة:

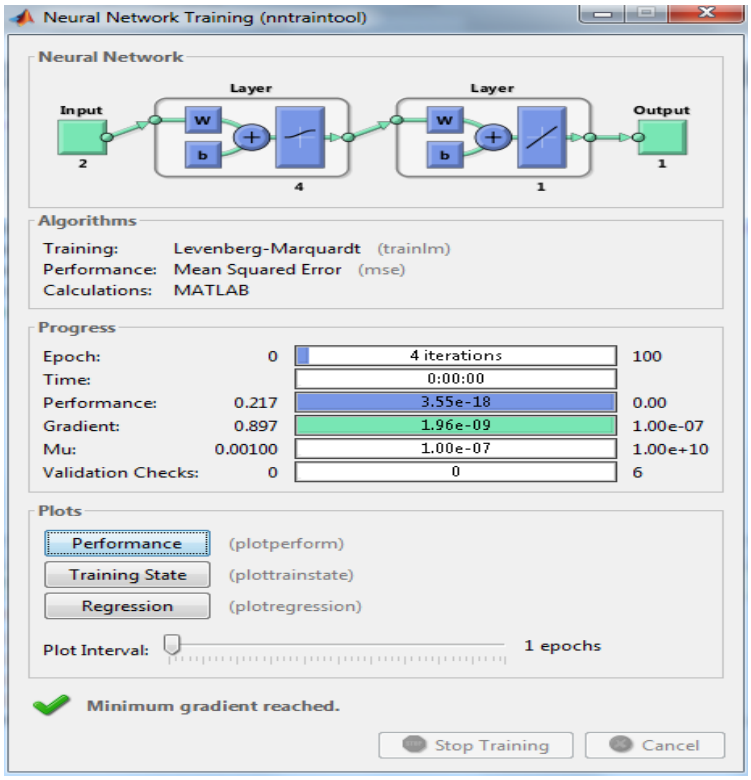
```
net.trainParam.epochs=100;

net.trainParam.show=10;

net.trainParam.goal=1e-6;

net = train (net,p,T);
```

- يمكن الدخول لبارامترات تدريب الشبكة وتعديلها مثل عدد مرات التدريب epochs وعدد مرات عرض النواتج show والهدف المرغوب الذي سيتوقف التدريب عند الوصول إليه goal وغيرها من البارامترات.

- بعد تحديد كامل الخصائص يمكن تدريب الشبكة باستخدام التابع train بإدخال ثلاث بارامترات، اسم الشبكة وبيانات الدخل وبيانات الخرج المرغوب.

```
net = train (net,p,T);
```

يمثل الشكل المبين جانباً أداة تدريب الشبكات العصبونية في برنامج ماتلاب ، وتظهر بنية الشبكة وعدد عصبونات طبقاتها وخوارزمية تدريبها وكامل تفاصيلها .



3- اختبار الشبكة:

```
y=sim(net,p)
```

باستخدام التابع sim وبإدخال بارامترين، اسم الشبكة المدرّبة والبيانات التي نريد اختبار الشبكة عندها، نلاحظ النتيجة التالية لخرج الشبكة:

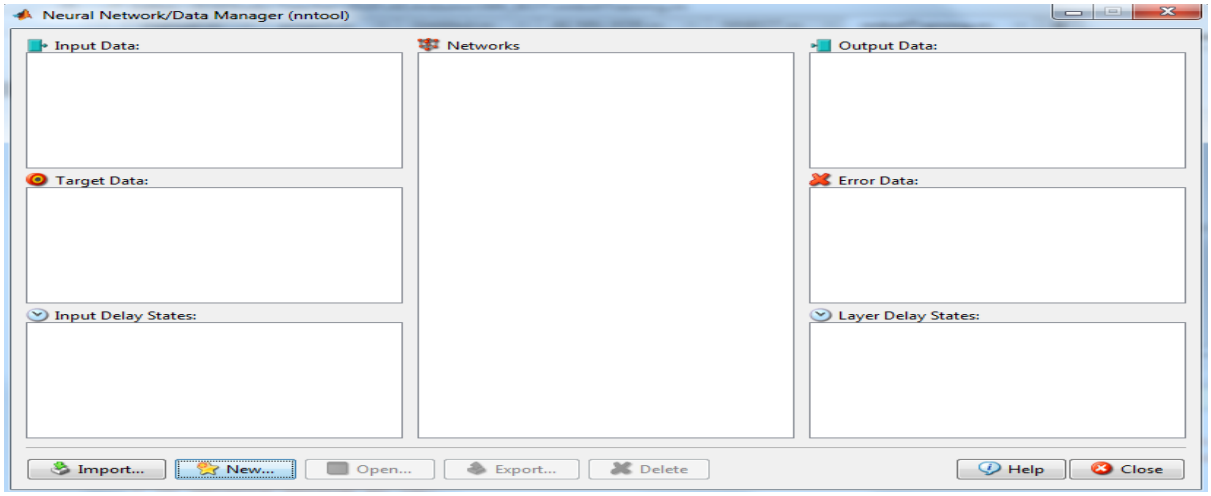
```
y =     0.0000    1.0000    1.0000    0.0000
```

لكن في الأمثلة العملية يجب الاختبار عند قيم لم يتم التدرب عليها مسبقاً.

واجهة ماتلاب للشبكات العصبونية:

يمكن استدعاؤها بالتعلمية :

```
<<nntool
```

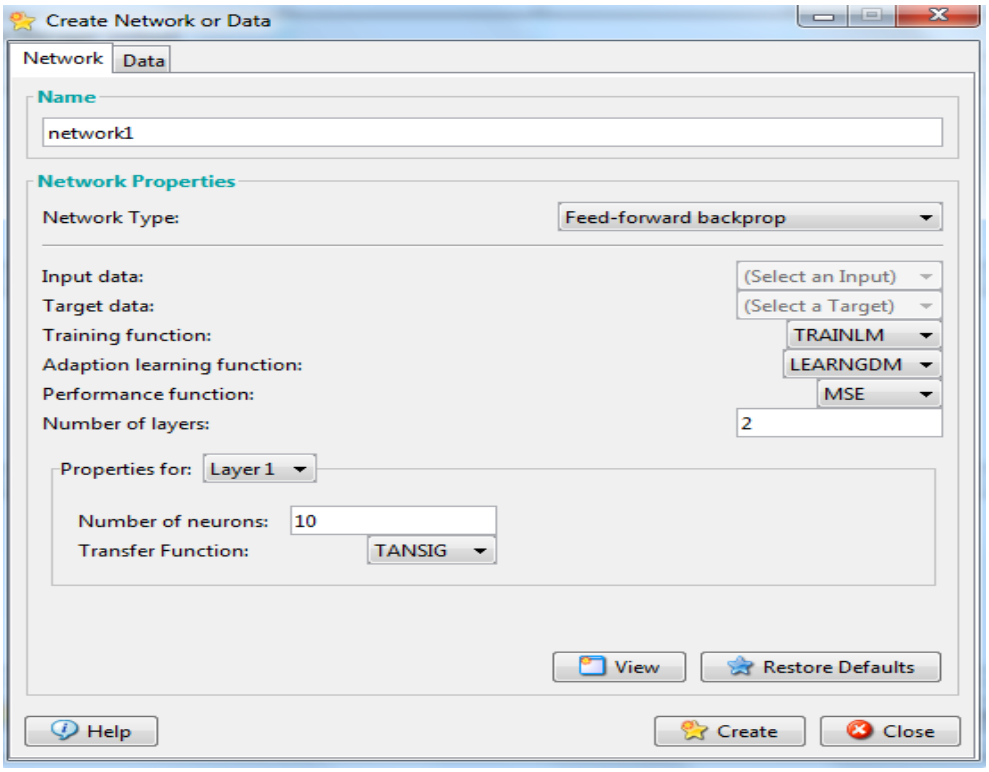


من هنا يتم استيراد الشبكة أو بيانات الدخل والخرج (Import)، أو يمكن إنشاء شبكة جديدة (New)

عند إنشاء شبكة جديدة (New) عندها تظهر النافذة التالية:

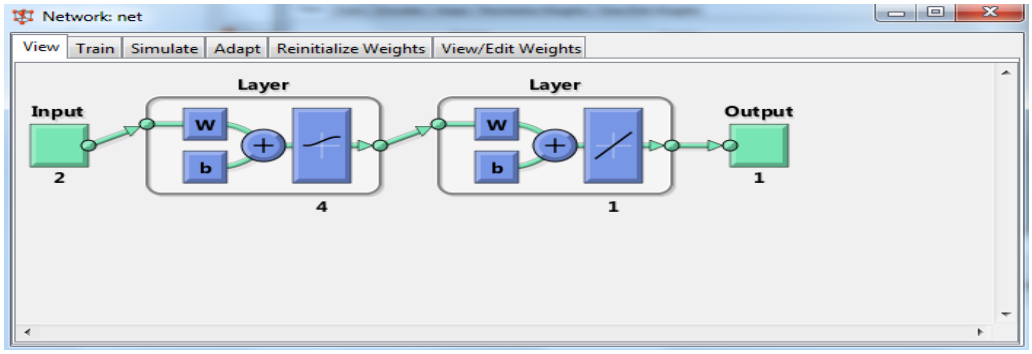
في هذه النافذة، وفي القائمة الأولى Network يمكن تحديد كامل تفاصيل الشبكة من اسمها ونوعها وعدد طبقاتها وعدد عصبونات كل طبقة إضافة لتابع تدريب كل طبقة وتابع التدريب الكلي وتابع الخطأ المستخدم.

في القائمة الثانية (Data) يمكن إدخال بيانات الدخل والخرج للشبكة.

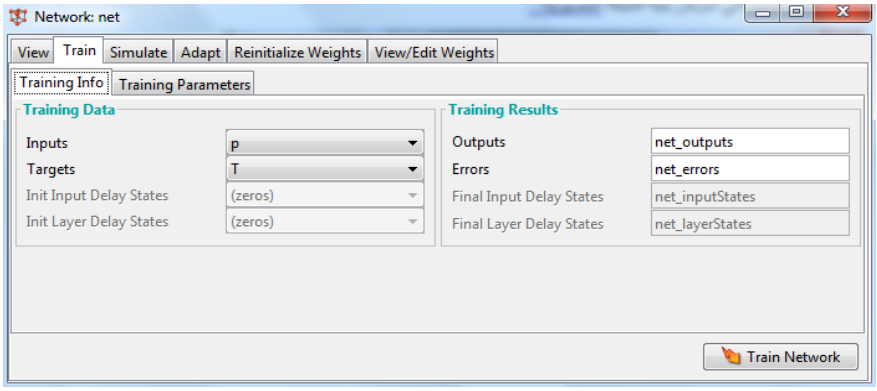


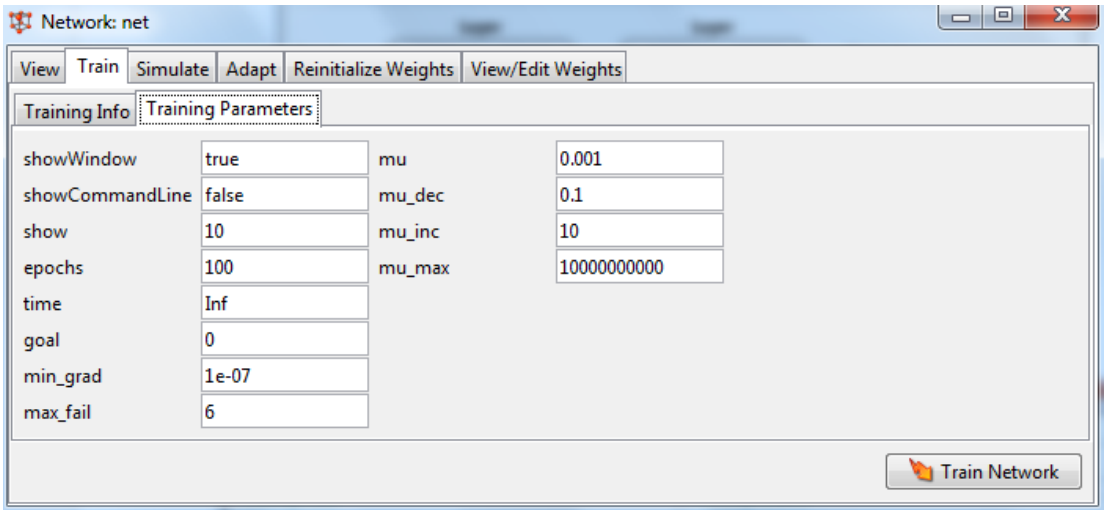
بعد تشكيل الشبكة (يمكن بالطبع تصديرها إلى فضاء عمل ماتلاب) أو التعامل معها من داخل الواجهة كتدريبها واختبارها، وذلك بتحديد الشبكة من النافذة الوسطى (سواء أكانت مستوردة أو منشأة بواسطة الواجهة) واختيار Open.

عندها ستظهر النافذة التالية التي تعرض بنية الشبكة العصبونية:

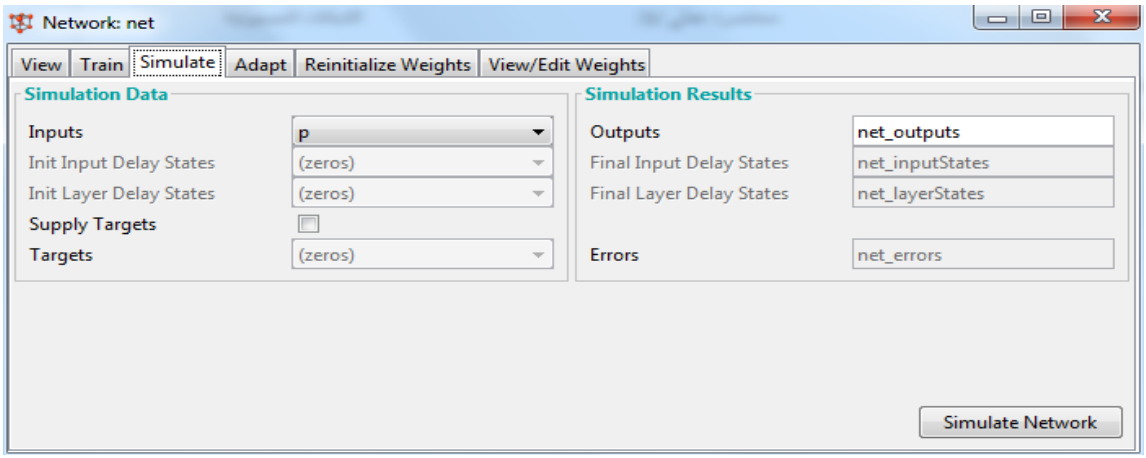


نقوم اختيار Train لتدريب الشبكة وفق بيانات دخل وخرج محدد، وبارمترات محددة (عدد المرات والهدف وغيرها)، ثم نضغط (Train Network):





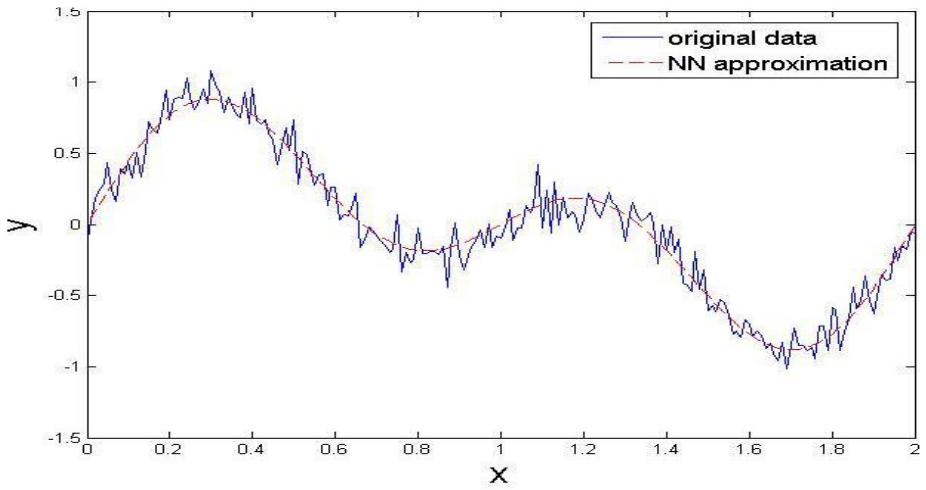
الخطوة الأخيرة هي اختبار الشبكة عند بيانات الاختبار، وذلك باختيار (Simulate) وتحديد بيانات الدخل الاختباري ثم اختبار الشبكة، ثم الضغط على (Simulate Network) ستصدر النتائج إلى فضاء عمل ماتلاب ليتم التعامل معها لاحقاً كما يقتضي التطبيق.



مقاربة التوابع Function Approximation

عملية مقاربة التوابع هي عملية ضبط أوزان الشبكة بحيث تنتج نفس استجابة تابع غير معروف عند نفس الدخل للنظامين (التابع والشبكة).

إن شبكة عصبونية ذات تغذية أمامية مؤلفة من طبقة مخفية واحدة بتوابع تفعيل sigmoid في هذه الطبقة، وخطية في طبقة الخرج، قادرة على مقاربة أي تابع مستمر بشرط أن تحتوي على عدد كافٍ من العصبونات في الطبقة المخفية .



مثال 1 – مقاربة تابع :

ليكن لدينا التابع التالي (مجموع إشارتين جيبيتين) والمطلوب مقاربته باستخدام الشبكات العصبونية.

$$y = 0.5 * \sin(\pi x) + 0.5 * \sin(2\pi x), \quad x \in [0,2]$$

- في البداية يجب الحصول على قيم التابع y عند قيم دخل محددة x لتدريب الشبكة عليها، وهذه القيم للتابع y تنتج عن تطبيق المعادلة السابقة على كل قيمة من قيم الشعاع x مع إضافة ضجيج للمعادلة.

```
x = 0:0.01:2.0;
y=0.5*sin(pi*x)+0.5*sin(2*pi*x)+0.1*randn(size(x));
```

الآن أصبح بالإمكان البدء بخطوات تصميم الشبكة العصبونية

1- إنشاء الشبكة:

- يمكن إنشاء الشبكة إما بالتابع المعرف سابقاً (newff) بالتعليمة التالية:

```
net=newff(minmax(x),[4 1],{'tansig','purelin'},'trainlm');
```

- أو باستخدام أحد التوابع البديلة مثل feedforwardnet:

```
net=feedforwardnet(4,'trainlm');
```

التابع feedforwardnet:

- عند إنشاء الشبكة باستخدام التابع feedforwardnet فإن البارامترات المطلوبة هي فقط عدد العصبونات في كل طبقة مخفية (طبقة مخفية واحدة بـ 4 عصبونات في مثالنا) ، وتابع التدريب الكلي للشبكة (في مثالنا trainlm):

```
net=feedforwardnet(4,'trainlm');
```

- بالتالي لا يمكن مباشرة تحديد تابع التفعيل في طبقة ما كما كان الحال في التابع newff، بل نحتاج للدخول إلى خصائص الطبقة المطلوبة في الشبكة وتعديل تابع التفعيل على الشكل التالي:

```
net.layers{1}.transferFcn='tansig';
net.layers{2}.transferFcn='purelin';
```

2- تدريب الشبكة:

- في البداية نحدد بارامترات التدريب المرغوبة (150 تكرار والهدف 1e-6):

```
net.trainParam.epochs = 150;
net.trainParam.goal = 1e-6;
```

- ثم ندرب الشبكة على الدخل والخرج المرغوب:

```
net = train(net,x,y);
```

3 - اختبار الشبكة:

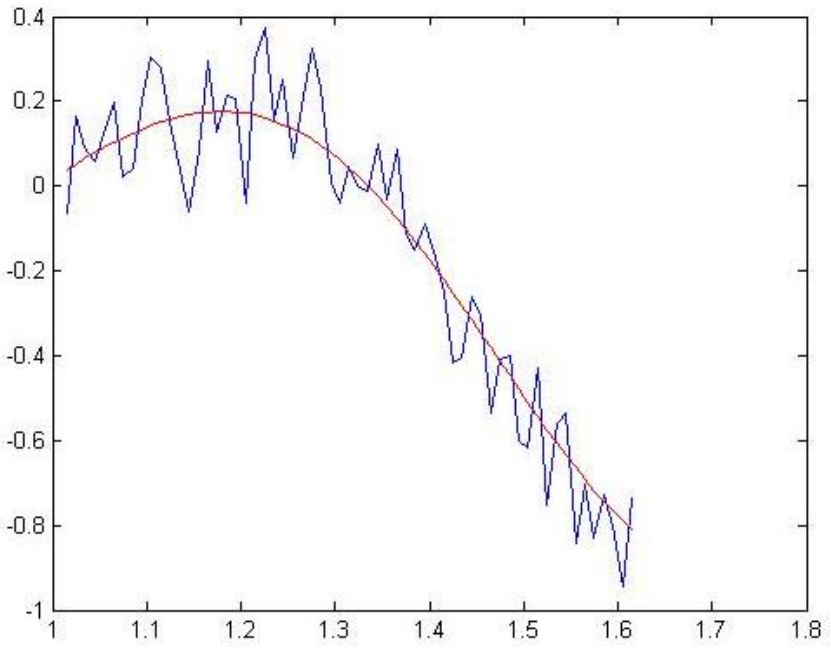
- يجب اختبار الشبكة عند بيانات دخل لم يتم التدرب عليها مسبقاً، بالتالي ننشئ شعاع قيم مغاير للقيم التي تدربت عليها الشبكة ثم نختبرها عنده:

```
x1 = 1.015:0.01:1.615;
y1 = 0.5*sin(pi*x1)+0.5*sin(2*pi*x1)+0.1*randn(size(x1));
out1 = sim(net,x1);
```

لتوضيح نتيجة المقاربة نقارن بالرسم بين خرج التابع y1 عند الدخل x1 (الخرج الفعلي) والخرج الناتج عن الشبكة العصبونية out1 عند نفس الدخل x1. :

```
plot(x1, y1, 'b',x1,out1,'r');
```

حيث يشكل الخط الأزرق الخرج الفعلي للتابع بعد إضافة التشويش، بينما يمثل الخط الأحمر خرج الشبكة العصبونية بعد إتمام عملية مقاربة التابع بنجاح.


0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
1
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8