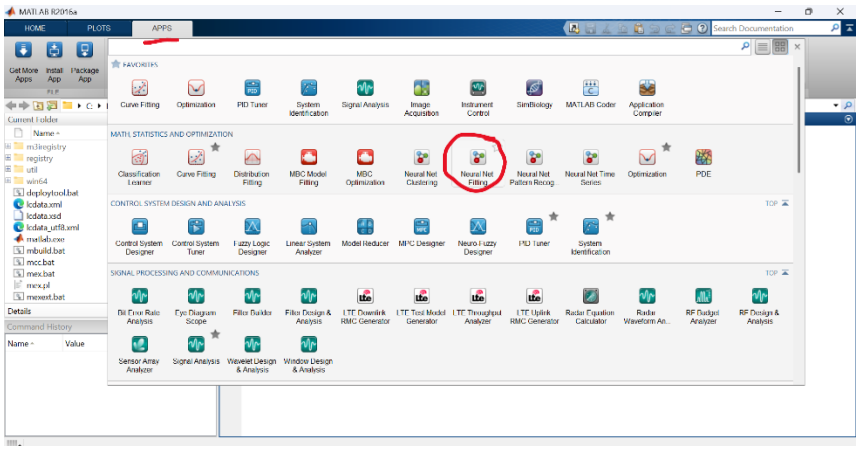
جلسه دوم شبکه عصبی MLP در متلب

خب میخواهیم در این جلسه شروع به کار عملی داشته باشیم در محیط متلب خب برای این کار ما از APP طراحی شده متلب استفاده میکنیم که در قسمت App ها میخواهیم Neural fitting استفاده کنیم که این ابزار به صورت زیر دسترسی داردیم در محیط متلب :

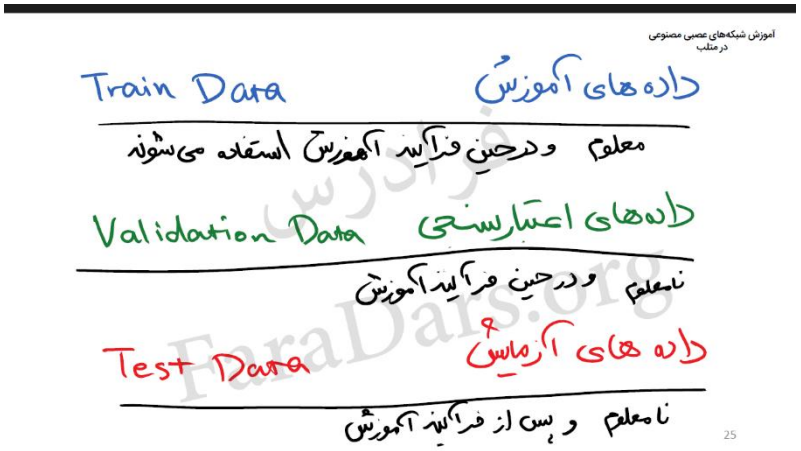


خب برای ادامه روند کاری با این App باید یک سری دیتا از قبل ما ساخته باشیم یا باید داشته باشیم که به سیستم و شبکه عصبی خودمان دهیم برای این کار از همان روند ساخت یک

x = linspace(0,2*pi,20) استفاده میکنیم در command window یا میتوانیم به صورت یک اسکریپت بنویسیم در یک m فایل متلبی سپس با این تابع y = sin(x) را ساخته سپس بعد از آن باید روند کاری خود را د ر محیط اپلیکیشین متلب جلو ببریم قبل از این کار من یک سری مطالب را میخواهم بنویسم که خالی از لطف نیست برای ادامه روند یادگیری :

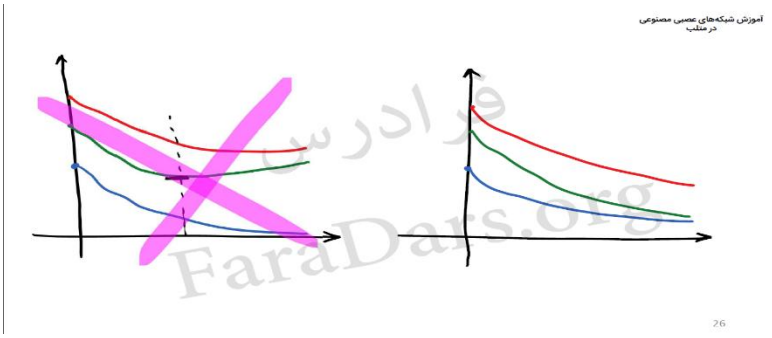
فرض کنید ما یک درسیس مانند ه ریاضی عمومی 1 که شامل یک سری اطلاعات است باید به دانشجویان ترم اول آموزش دهیم . این اطلاعات دارای پیچیدگی های خاص خودشان است اما ما شروع به یاد دهی یا آموزش به دانشجویان میشویم حال در روند رو به جلو انتقال اطلاعات ما نمیدانیم که این دانشجویان آیا این چیزی که به عنوان درس ریاضیات عمومی 1 است فرا گرفته اند یا خیر . خب در همین جا ما یک سری HW را طراحی میکنیم به دانشجویان میدهیم و یک سری کوئیز

های کلاسی طراحی میکنیم حال باید ببنیم دیتاهای خروجی از این دانشجویان چقدر با آن چیزی که مد نظر ما است نزدیک شده یا دور شده که اگر ببنیم روند دانشجویان به خوبی دارد پیش میرود که کل کلاس دارای یک میانگین وزنی خوبی از دئرس ریاضیات هستند پس آموزش درست کار کرده اگر دیتای خروجی از Tets های ما که تمرینات کلاسی و کوئیز ها است خوب نباشد پس یک جای کار مشکل دارد یا آموزش ما است یا مدل درست کار نمیکند. خب برای تمامی چیزی که از بالا تا اینجا آورده شده را در تصویر میکشیم :



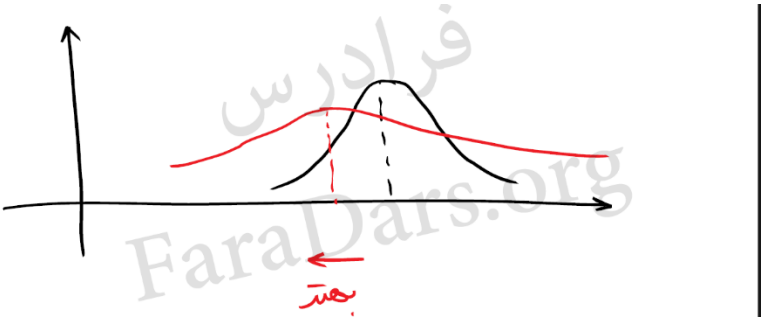
خب در تصویر نوع دیتا ها را مشاهده میکنید دیتای آموزش ،دیتای اعتبار سنجی و تست که همگی اینها در شکل بعدی به خوبی باز هم به نمایش میگذاریم :

در تصویر زیر مشاهده میکنید که دیتای سمت چپهمگپرایی خیلی خوبی دارد در فرایند آموزش یعنی بچه سرکلاس میگن خوب درس ریاضی را میفهمیم در ادامه یک امتحان طراحی کرده به عنوان تست سپس دیتای تست اعتبار سنجی میشه با دیتای آموزش که ببینیم چقدر به آمن چیزی که میخواستیم نزدیک شده است آموزش ما که در اینجا میبینیم تست و اعتبار سنجی دارن با مرور زمان فاصله میگیرند از دیتای آموزشی ما خب اما سمت راست میبینیم دیتای آموزشی خیلی همگرا نیست به آن چیزی که مدنظر مدل یا ما باشیم است اما خروجی کار که اعتبار سنجی با دیتای تست و همین دیتای آموزش است میفهمیم روش کار یا آموزش دادن ما بهتر بوده و عملکردی بهتری از خود گذاشته است پس این مدل در ادامه نتایج بهتری را برای ما به ارمغان میاورد .



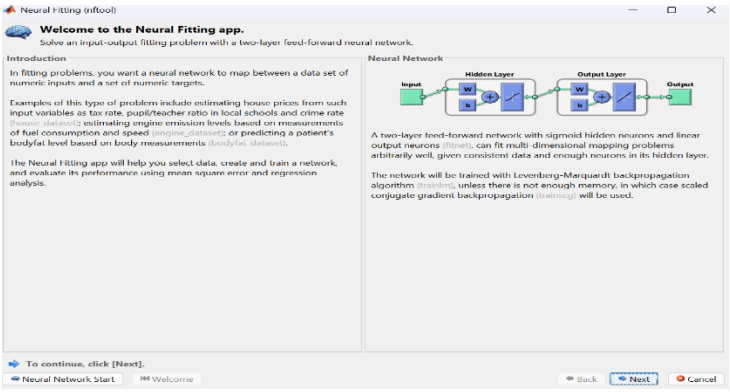
خب در ادامه کار صحبت بر این میشود خب درسته نمودار سمت راستی دارای صحت کمتری است نسبت به نمودار سمت چپی اما عملکرد ما برای راستی آزمایی دیتای های آموزش بهتر است .

اما این نکته را نباید فراموش کنیم که یکی پارامتر های تاثیر گذرا در روند تعین بهتر بود یا نبودن دیتای تصادفی انحراف معیار است که هر چقدر این کم تر باشد بهتر است ، شاید انحراف از معیار یک دیتا کم نباشد و بتواند دیتای زیاد ی در خود جای دهد اما ارزش دقت هم خیلی بالاست چون در سیستم های شبکه عصبی با دیتا هیا stacatic یا تصادفی نیز در گیر هستیم تصویر زیر گوای صحبت های من است :

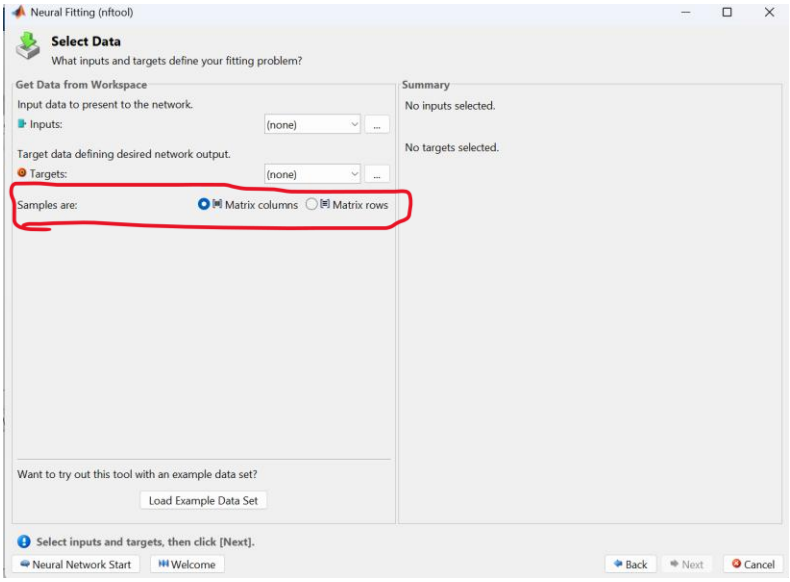


قرمز سهم بیشتری زا اطلاعات در خود دارد اما نمیتواند به خوبی دقت از خود نشان دهد اما مشکی رنگ دیتای کمی از خود دارد اما دقت در آن بالاست .

در ادامه ما درمحیط متلب کار با نرم افزار را شروع میکنیم که بعد کلیک کردن روی آیکون محیط زیر باز میشود :

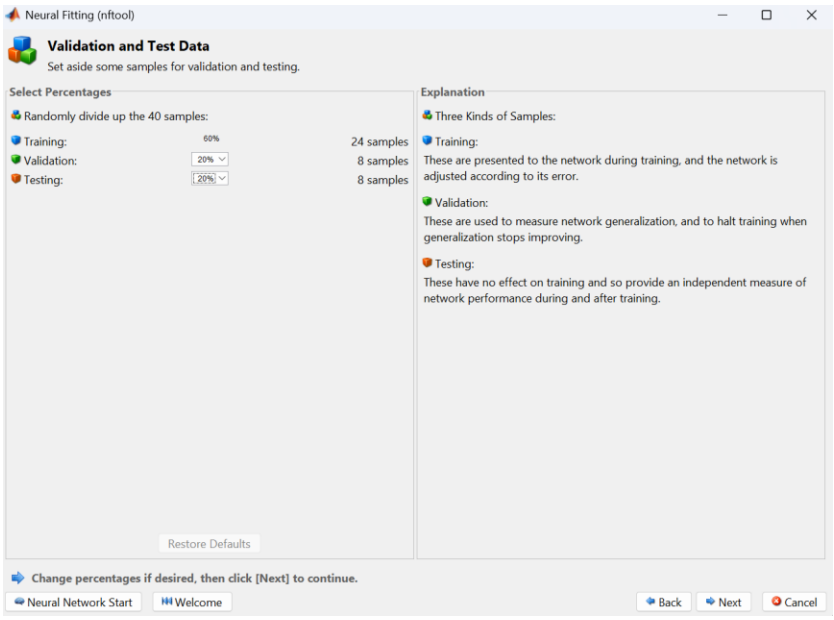


خب همان طور مشاهده میکنید یک سری توضیحات درمورد اپلیکیشن نوشته شده سپس دکمه Next زده در انجا وار محیط بعد میشویم گکه دیتا های خود را از محیط workspace به این اپلیکیشن باید اضافه کنیم .



خب در اینجا دیتای ورودی و خروجی یا همالن تارگت را باید بدهیم سپس باید توجه داشته باشیم که دیتای ورودی خروجی ما ستونی هستند زوج مرتب هایی که است یا سطری که در گزنیه ای که دورش خط کشیده ام باید دقت شود .

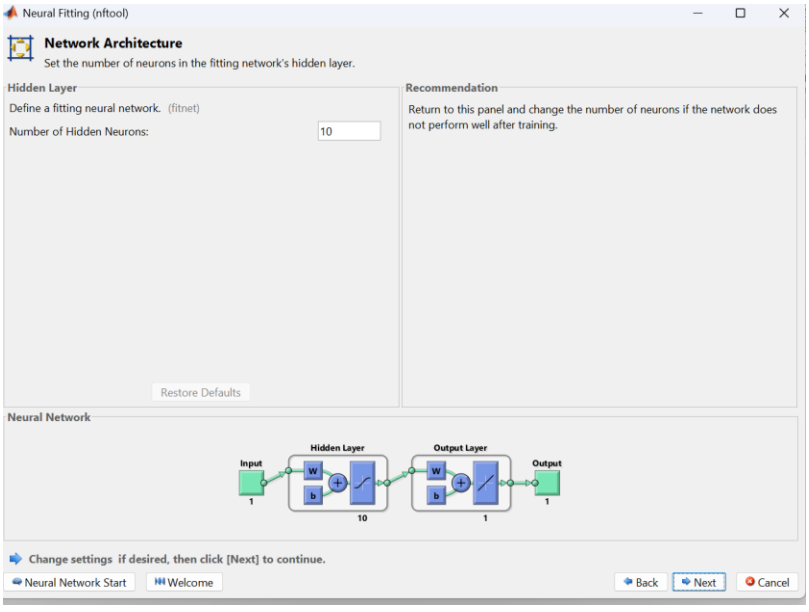
خب در ادامه کار دکمه Next را زده و به صقحه بعد میرویم که جایی است باید در آموزش ،اعتبار سنجی و تست را برای شبکه عصبی را تعین کنیم که به شکل زیر است :



خب برای این کار میبینید محیط متلب به خوبی برای ما آپشن های خیلی خیلی خوبی را طراحی کرده تا بتوانیم این قسمت از طراحی شبکه عصبی مصنوع یخود را به خوبی انجام دهیم .

تمامی صحبت های این قسمت را کرده ام برای اطلاعات بیشتر هم باز هم در خود این محیط یک سری توضیحات داده که خوامدش خالی از لطف نیست .

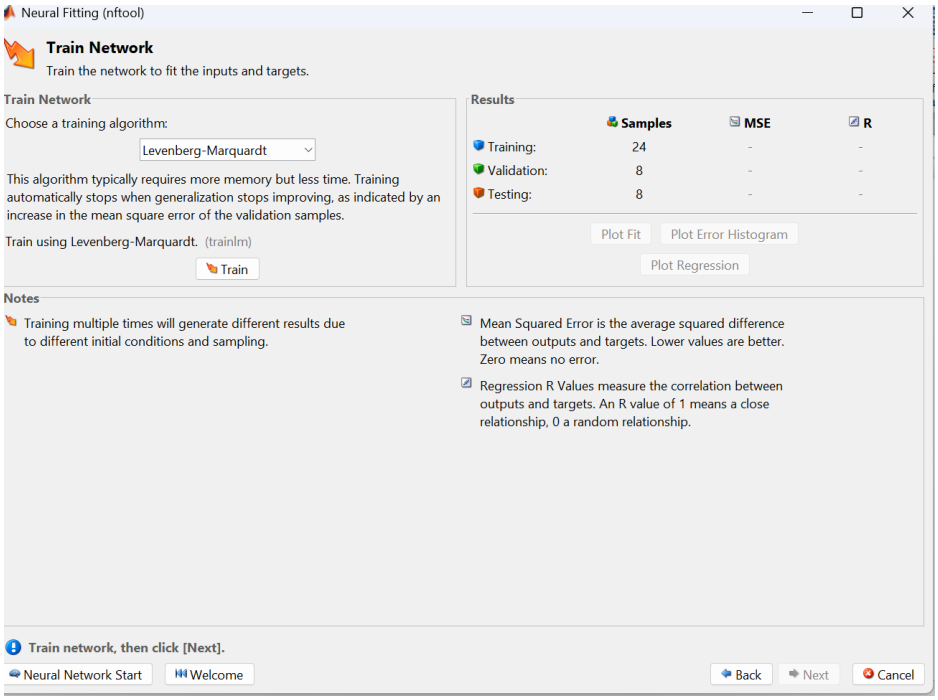
در ادامه کار باید لایه های پنهان را مشخص کنیم که این یک صحبت تخصصی است که فعلا از این موضع گذر میکنیم و هر چند لایه ای که دارد را به عنوان کار خودمان میگیریم .



خب در اینجا میبینید که تعداد لایه های ما را 10 تا داده که همین مقدرا را میگذاریم تغیر نکند .

در ادامه راه باید وارد محیط اصلی کار میشویم که الگوریتم مورد استفاده را مشخص کنیم که چه چیزی است و در نهایت شروع به آموزش دادن الگوریتم شویم .

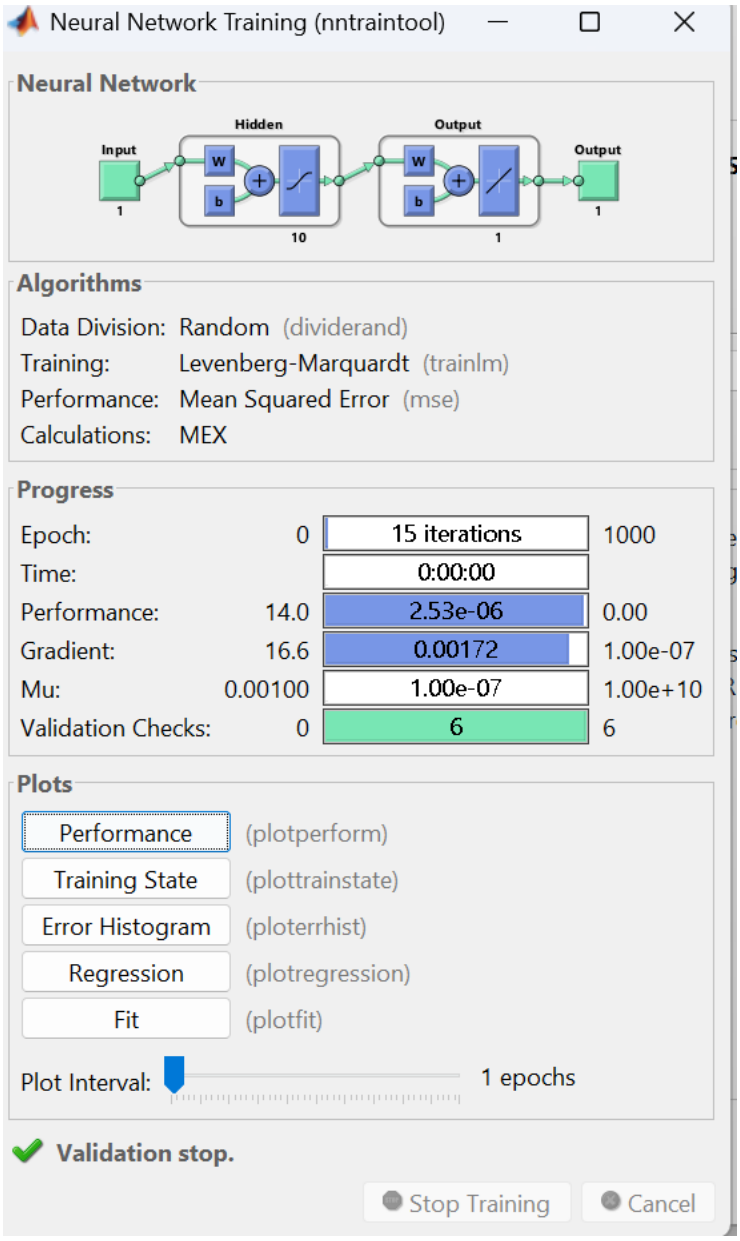
این محیط Traning است که باید از موی کشویی که داریم نوع آموزش شبکه عصبی را مشخص کنیم که در اینجا Levnberg به عنوان cruve fitting استفاده میکنیم چون در این سیستم شبکه عصبی هدف ما داده هایی که داریم را یک منحنی بر روی آن برازش کنیم که این متد و



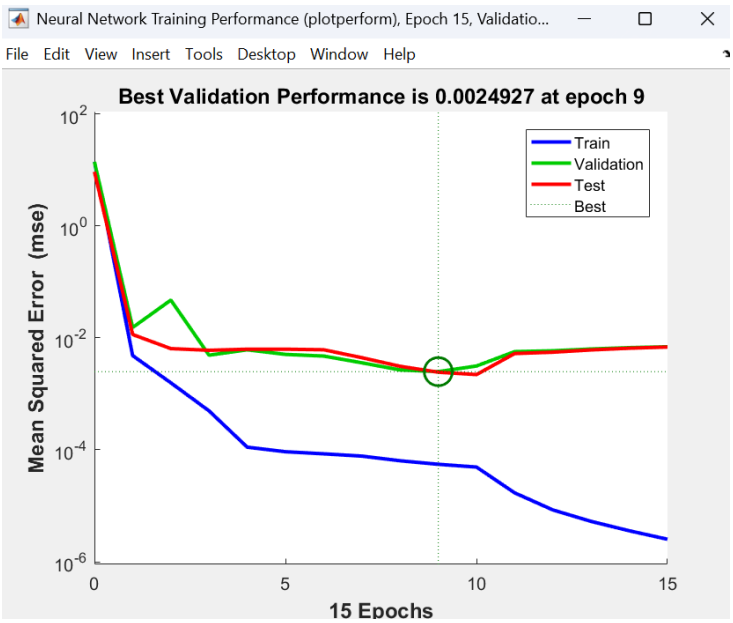
سیستم و تمامی این کار ها برای همین است . این هم بگم داد ه ورودی ما یک تابع سینوسی است که به عنوان ورودی گرفته ایم . خب در ادامه کار دکمه Train را زده و دیتا ها شروع به آموزش دیدن میکنند و بعد گذشت زمانی بستهخ به نوع سخت افزار شما نرم افزار این آیکون را به نمایش میگذراد که حاوری بسیاری از اطلاعات است :

خب این پنجره بعد از این که تمام ششده روند کاری ظاهر میشود خب در قسمت الگوریتم که نوشته شده دیتاهای مرتبط به نوع الگوریتم نوع متد خطایی که استفاده نموده شده که اینجا روش MSE است و... در قسمت پردازش اطلاعات اولین Epoch یعنی تعداد بار هایی که آموزش دیده دیتا ها دومی زمان است سومی و چهارمی میزان دقت و خطایی که داریم را نشان میدهن د گزینه آخر هم میزان چک کردن دیتا ها را نشان میدهد که در این فرایند چند بار اجازه یافته دیتا های خروجی را چک کنیم این خودش یک محدودیت دارد که از یک حدی بیشتر نباید رود .

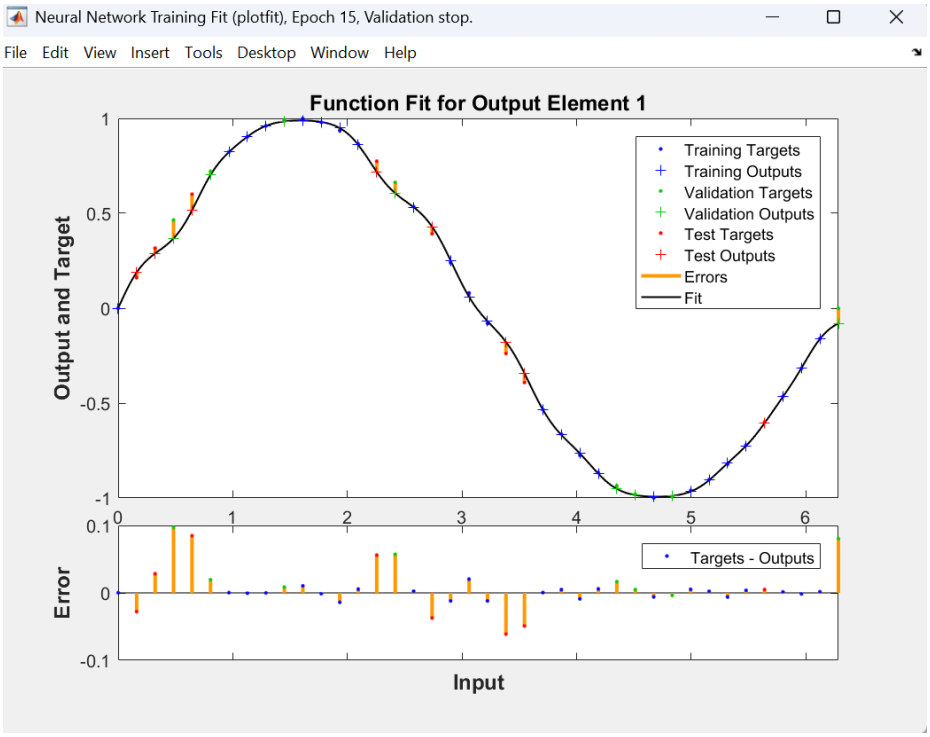
سپس د ر قسمت Plot نمودار ها را به ما میدهد که در شکل های زیر نمودار های



خروجی گرفته شده از آن ها میاورم :

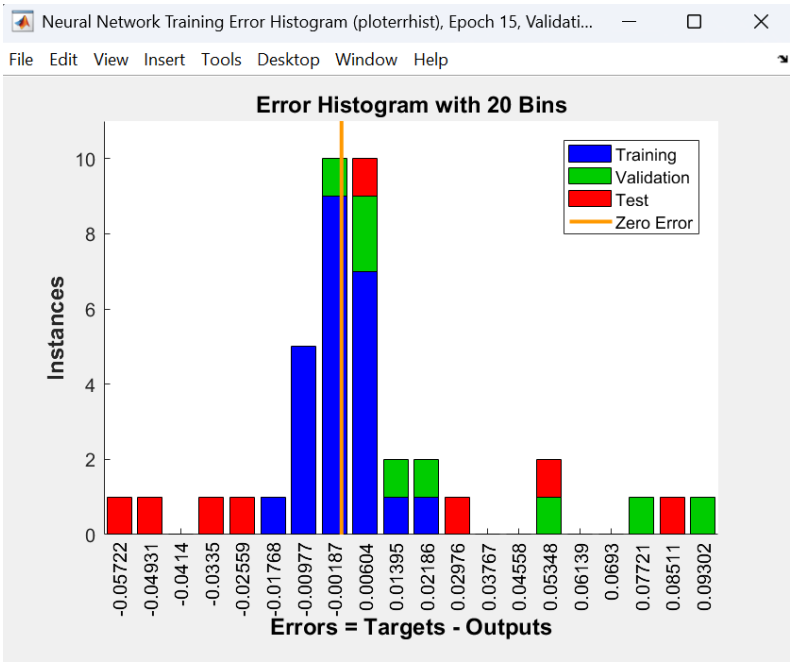


این نمودار مربوط به خطا و تستکردن اطلاعات و میزان همگرایی و داستان هایی که گفتیم است که فکر کنم توضیحات کافی داده ام میتوانید خودتان این نمودار را آنالیز کنید .

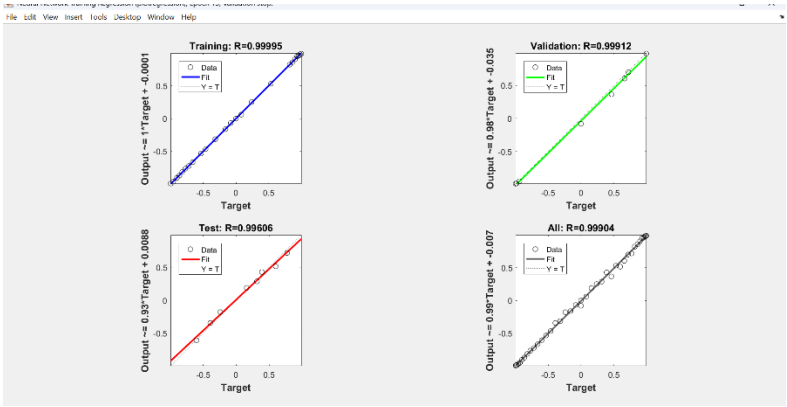


اینم نموداتر دیگه ما که میزان فیت شدن شبکه عصبیی بر روی اطلاعات اصلی ما را نشان میدهد که چقدر دقت داشته ایم و نحوه عملکر و....

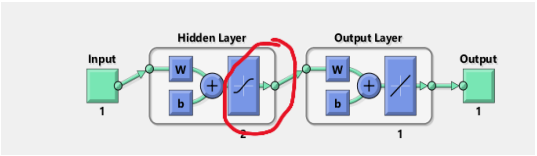
اینم نمودار هیستوگرام است که میزان گستره دیتا ها و انحراف معیار ها را میتوان از طریق این خروجی گرفت .



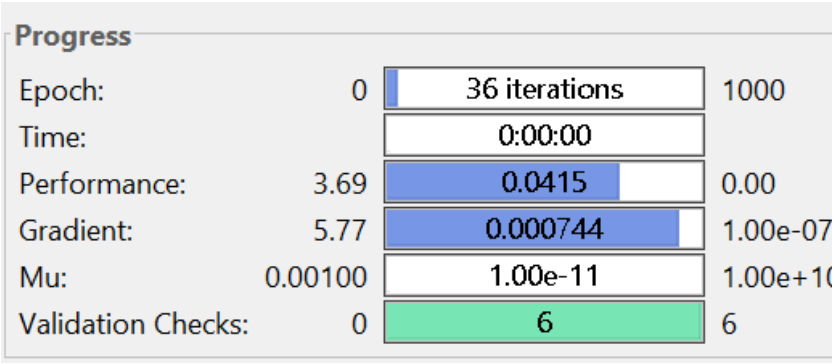
در نهایت این نمودار زیر هم رگرسیون های نختلفرا برای دیتاهای خروجی و... و آنالیز دیتا را به ما میدهد.



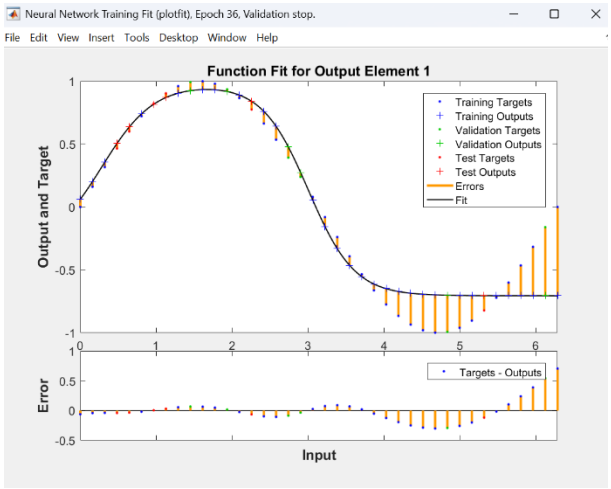
خب اگر یادتان باشد در قسمت لایه ها ی همان نرون ها گفتم هر عددی است را بگذاریم و دوباره برایتان توضیح میدهیم در اینجا ابتدای کار باید در مورد یک تابع توضیح بدم که تحت عنوان سیگموید SIGMOIND شناخته میشه در کارای هوش مصنوعی که با این المان در متلب یا گراف های توصیفی نشان داده میشود.



خب همان طور که میبنید شبیه یک تابع نمایش میدهد اما این همان چیزی است که لایه های ما تولید میشوند که سیگموند از کلمه sigma و oid تشکیل شده که هر تابعی که S شکل باشد راگویند مانند تانژانت هاپربولیک و... خب در متلب هم این تابع استفاده میشه خب در قسمت لایه ما دو لایه نگذاشتیم حال با همان دیتای اولیه بیاییم ببنیم چه فرمی میدهد fitting ما که نمایش آمن به شکل زیر است :

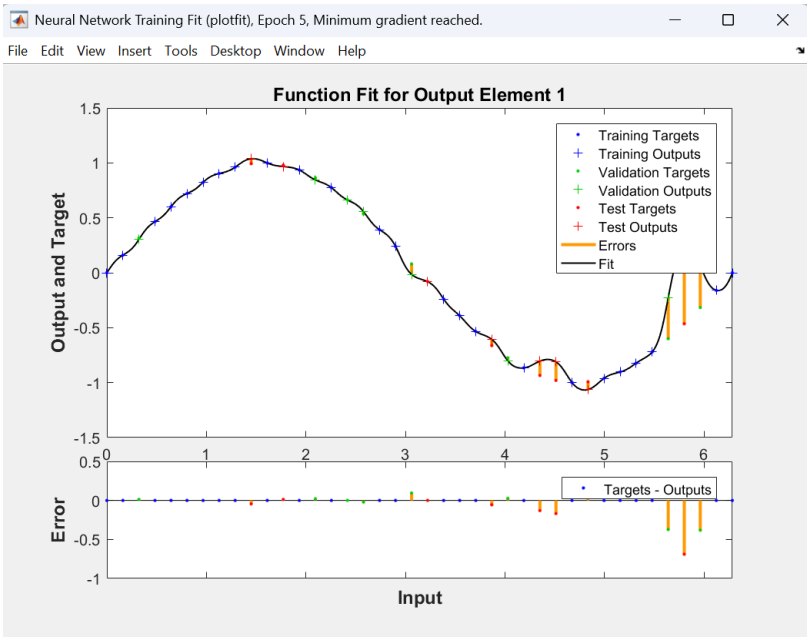


این اما از تعداد تکرار هایی که انجام داده برای تقریسب زنی تابع که تعدادش بیشتر شده ولی چیزای دیگه در همان order است اتما نکته ای که برای من و شما خیلی مهمه بزارید دیتای fitting ببنید که چه چیزی رخ داده :



خب همان طور مشاهده میکنید با 2 لایه نتوانسته به خوبی کار تقریب زنی تابع هدف ما را انجام دهد منحنی آبی همان sigmoind است حال اگر تعداد لایه ها بیشتر کنیم بزارید ببینیم چه چیزی رخ میدهد :

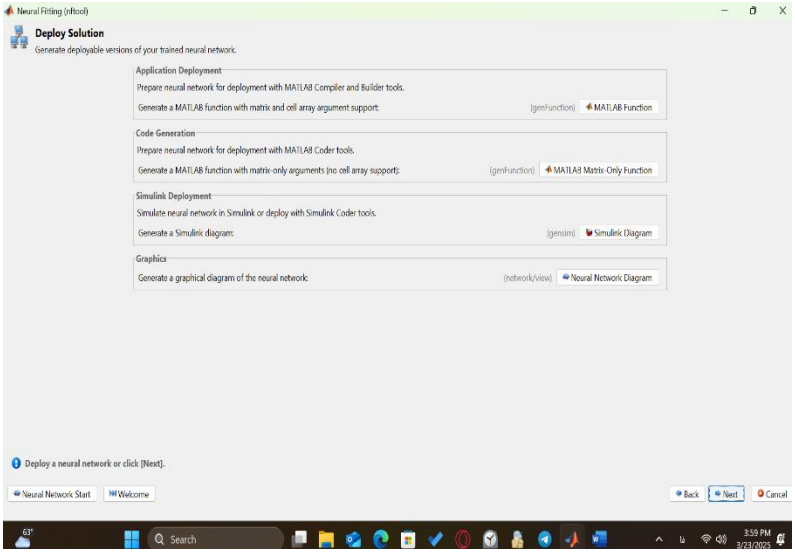
خب در این تصویر هم تعداد دیتاها بیشتر شده هم لایه ها را برای آموزش بیشتر کردم که خروجی کار به فرم زیر شده است :



ملاحظه میکنید که تابع ما بیشتر قریب خورد با ایجاد لایه بیشتر که برایش ساختیم و مابقی اطلاعات هم از اپلیکیشن میتوانیم خروجی بگیریم . خب میبنید خروجی کار با این که لایه ها افازیش پیدا کردن اما اون چیزی که باید بشه نیست خب دلیل هم اینه که نمیشه تمامی الگو ها و لایه ها به درستی کنترل نمود خروجی کار دقیقا اون چیزی نمیشود که مدنظر ما است .

خب برای این شبکه عصبی میدونید چه اتفاقی رخ داده است مانند این میموند که ما بینهایت قدرت مغزی و حافظه ای داریم آیا بنظر شما خواندن ریاضی و فهمیدن ان کار منطقی است ؟ خب خیر حفظ میکنید آن را و جلو میبرید آن این شبکه هم بخاطر بیش اندازه دارای لایه و حافظه است پس حافظه یادگیری زیادی دارد برای کار کردن در نتیجه درست عمل نمیکند.

خب در ادامه کار میخواهیم به یکی از دیگر پارامتر های قابل دسترسی در Nueral Network بپردازیم که برای این امر باید در قسمتی که دیتا میخواهیم Traning کنیم دسترسی به به اسکریپ فایل های معمولی و پیشرفته را داریم . که برای این کار به شرح زیر است :



خب در این تصویر مشاهده میکنید دوتا گزینه اول دسترسی شما میدهند به پشت پرده این ابزاری که دارید کار میکنید برای اطلاعات و کارای بیشتر شما میتونید در آن جا خودتان دیتا ها و توابع های مختلف را دستکاری کنید تا خروجی مدنظر را بگیرید این محیط خیلی دسترسی بهتری را برای شمات میدهد حتما استفاده کنید من دوتا نمونه فایل از این محیط برای شما در ریپازیتوری میزارمخ که به اسم های :

DBZ_NeuralNetwork_Advance

DBZ_NeuralNetwork_Simple

این دو فایل هستند که میتونید ببنید در گیت هاب بنده .

خب یک قسمت هست شما میتونید برای کار خود در همین قسمت کد نویسی اضافه کرد تحت عنوان توابع Transfer function که این ها همان Activator ها هستند که در شبکه های عصبی خیلی مهم و ضروری هستند که دیتا ها را چگونه برای ما فعال سازی کند که برای اینن کار به صورت زیر در کد نویسی عمل میکنیم اما برای اطلاعات بهتر از این توابع یکی به help NeuralNewtWork Matlab حتما مراجعه کنید دوم کتاب لوران فارست یکی از بهترین مراجع آموزشی در زمینه هوشی مصنوعی و شبکه عصبی است را پیشنهاد برای خواندن میکنم .

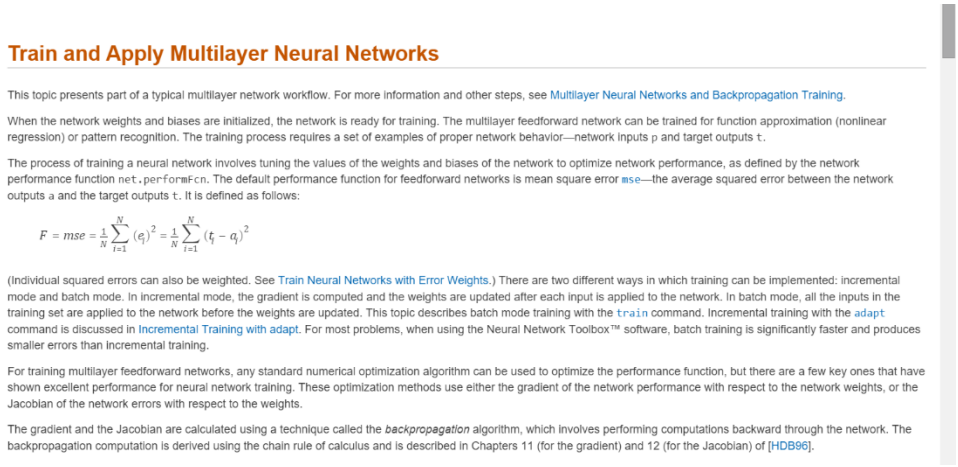
```
% Set Transfer Functions for Each Layer
net.layers{1}.transferFcn = 'radbas';  % Hidden layer
net.layers{2}.transferFcn = 'tansig';  % Output layer
```

خب همان طور میبنید

برای فعال سازی ما این دو تابع را فرا خوادنی کرده ایم که بتونیم خروجی بهتری بگیریم .

شما حتی در قسمت training دیتا هم میتونیم االگوریتم شبکه عصبی خودمان را تغیر دهیم تا بتونیم بهترین نتیجه را بگیریم که به شرح زیر است برای اطلاعات بیشتر میتونید به help در Matlab قسمت traning بروید تا متد های مختلف آنجا اورده شده است :

این که help متلب است که میتوانید برای کار خودتان بیشتر از همه از این کمک بگیرید سپس توابعی که نوشتهخ شده میتوانید استفاده کنید را به شما نمایش میدهم



Train and Apply Multilayer Neural Networks

This topic presents part of a typical multilayer network workflow. For more information and other steps, see Multilayer Neural Networks and Backpropagation Training.

When the network weights and biases are initialized, the network is ready for training. The multilayer feedforward network can be trained for function approximation (nonlinear regression) or pattern recognition. The training process requires a set of examples of proper network behavior—network inputs `p` and target outputs `t`.

The process of training a neural network involves tuning the values of the weights and biases of the network to optimize network performance, as defined by the network performance function `net.performFcn`. The default performance function for feedforward networks is mean square error `mse`—the average squared error between the network outputs `a` and the target outputs `t`. It is defined as follows:

$$F = mse = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(t_i - a_i)^2$$

(Individual squared errors can also be weighted. See Train Neural Networks with Error Weights.) There are two different ways in which training can be implemented: incremental mode and batch mode. In incremental mode, the gradient is computed and the weights are updated after each input is applied to the network. In batch mode, all the inputs in the training set are applied to the network before the weights are updated. This topic describes batch mode training with the `train` command. Incremental training with the `adapt` command is discussed in Incremental Training with adapt. For most problems, when using the Neural Network Toolbox™ software, batch training is significantly faster and produces smaller errors than incremental training.

For training multilayer feedforward networks, any standard numerical optimization algorithm can be used to optimize the performance function, but there are a few key ones that have shown excellent performance for neural network training. These optimization methods use either the gradient of the network performance with respect to the network weights, or the Jacobian of the network errors with respect to the weights.

The gradient and the Jacobian are calculated using a technique called the *backpropagation* algorithm, which involves performing computations backward through the network. The backpropagation computation is derived using the chain rule of calculus and is described in Chapters 11 (for the gradient) and 12 (for the Jacobian) of [HDB96].

:

Function	Algorithm
trainlm	Levenberg-Marquardt
trainbr	Bayesian Regularization
trainbfg	BFGS Quasi-Newton
trainrp	Resilient Backpropagation
trainscg	Scaled Conjugate Gradient
traincgb	Conjugate Gradient with Powell/Beale Restarts
traincgf	Fletcher-Powell Conjugate Gradient
traincgp	Polak-Ribiére Conjugate Gradient
trainoss	One Step Secant
traingdx	Variable Learning Rate Gradient Descent
traingdm	Gradient Descent with Momentum
traingd	Gradient Descent

توابعی هستند برای آموزش دهی الگوریتم شبکه عصبی شما طراحی شده است به صورت دیفالت در متلب حضور دارند حال مثلا من از الگوریتم Trianlm استفاده کرده ام :

```
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.
```

اینم کاری که من انجام داده ام .

یک قسمت دیگر مهم داریم که خطای ما را حساب میکند برای این کار میتوان به تصویر زیر نگاهی کرد که در کد هم آورده شده است :

```
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean Squared Error
```