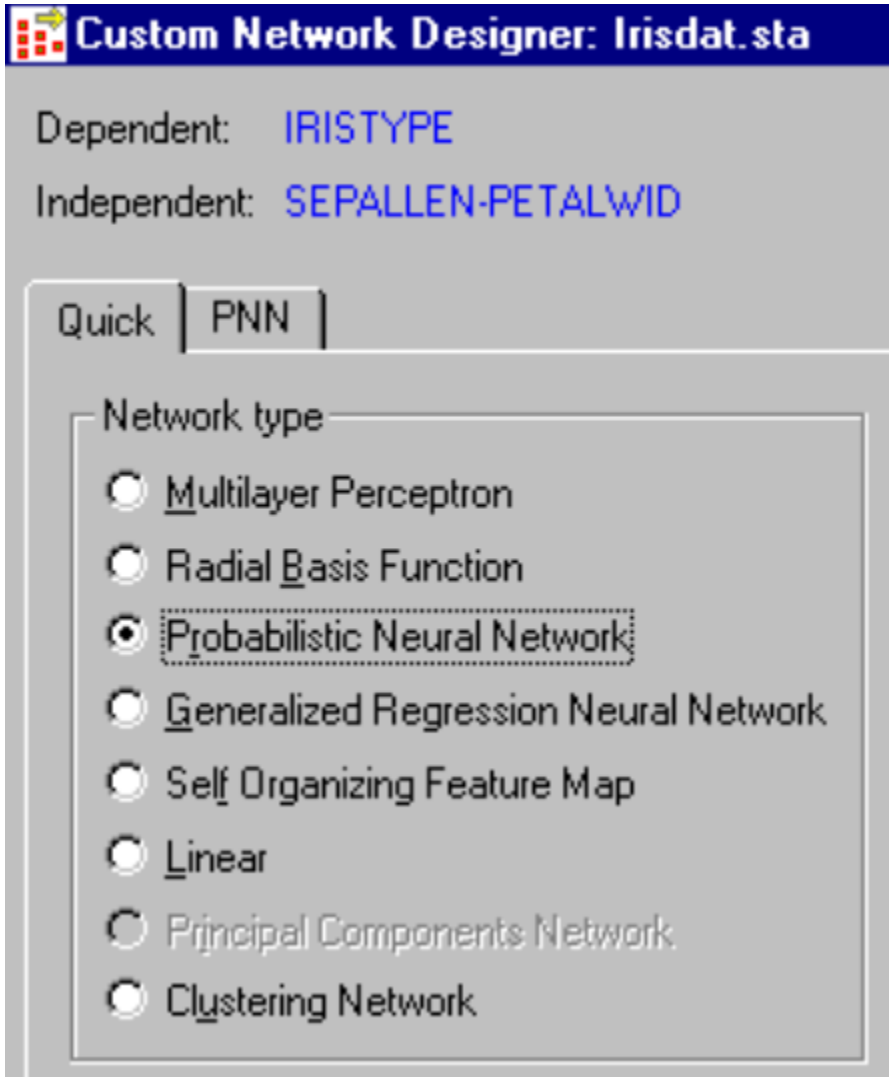
SNN Example 17: PNNs and GRNNs Revisited

Probabilistic Neural Network. A probabilistic neural network (PNN) is used for **classification problems**, and therefore requires a data set with a single nominal output variable (see Patterson, 1996; Bishop, 1995).

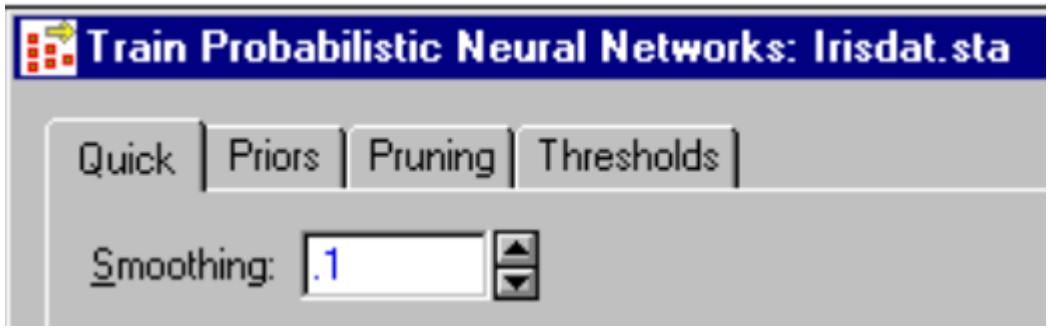
You do not need to specify the number of hidden **units in a PNN - this is automatically set to equal the number of training cases**.

There are few parameters to specify for PNN training: a *Smoothing* factor, optional *Prior probabilities* and (for four layer PNNs) an optional *Loss Matrix*

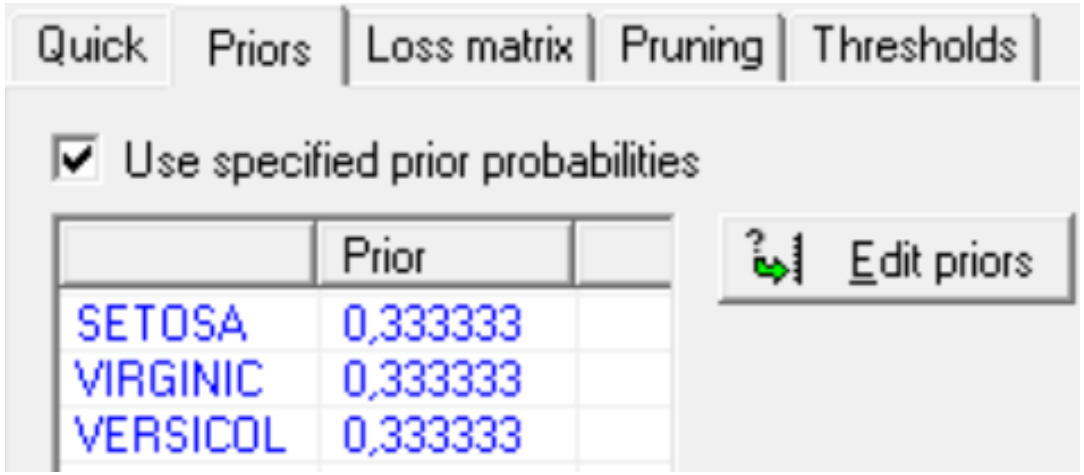


The **Smoothing factor** determines the widths of the Gaussian functions, centered at each training case and stored in the first hidden layer radial units. A small smoothing factor gives a sharp, spiky approximation to the underlying function, which may perform well on the training set but generalize poorly (and therefore perform badly on the selection set). A large smoothing factor gives a "blurred", smooth approximation, which may perform relatively poorly on the training set, but generalizes well to the selection set.

Usually, the algorithm is not too sensitive to the precise choice of smoothing factor. Let us try a few **values between 0.1 (usually a sensible lower limit) and 10**. Enter .1 into the Smoothing box and click OK to train the network.
Observe the **Train Error and Select Error** in the summary statistics at the top of the Results dialog. Then click Cancel to return to the Train Probabilistic Neural Networks dialog. Now, enter 10 into the Smoothing box and click OK to train the network. Again, observe the Train Error and Select Error in the summary statistics at the top of the Results dialog. You can repeat this process with more values if you would like.

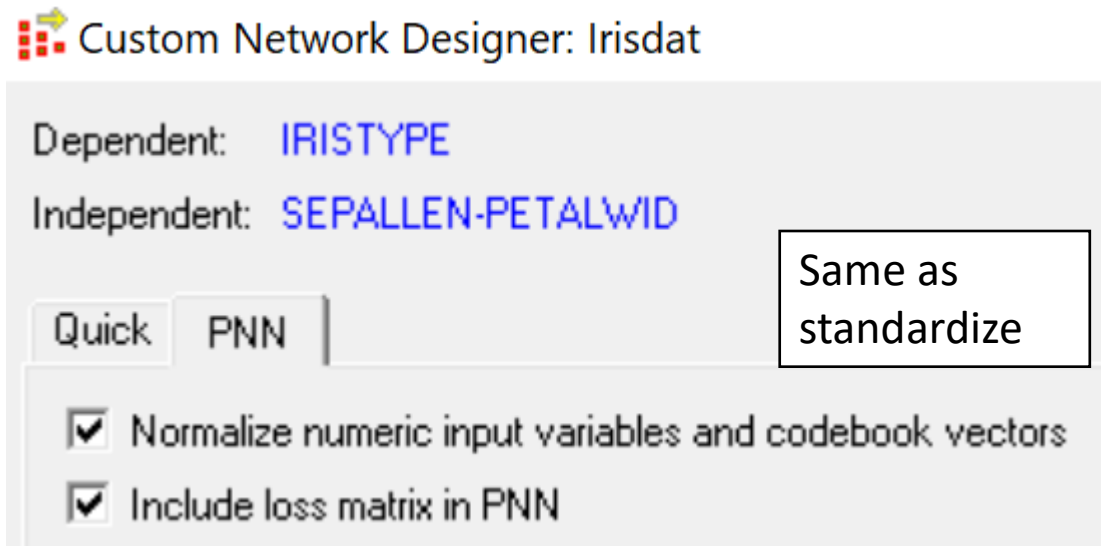


The figure to watch here is the **selection error** (you can always reduce the training error by reducing the smoothing factor, at least until you exceed the precision of the machine when the Gaussian curves are calculated). If you try out a smoothing factor of 0.01, you will likely get an extremely low training error and a high selection error.



The **Prior probabilities** are used in situations where you know that the **training set is biased**; for example, if in a two-class problem the population at large has only 1% of cases positive, with the rest negative, but your training set has 50% positive and 50% negative. As the PNN estimates class probabilities by adding "density estimates" centered at each case, the resulting probabilities will be highly biased unless adjusted to reflect the imbalance between population and data set representation. However, a common practice is to draw cases randomly, in which case there is no need for adjustment. Similarly, if you don't know the prior probabilities, the working assumption must be that the data set fairly represents the population distribution. In either case, prior probabilities do not need to be assigned. If you do want to change the prior probabilities, you would select the Use specific prior probabilities check box and then click the Edit button and enter the values in the edit spreadsheet.

Use Oversampling and Undersampling techniques to balance the dataset

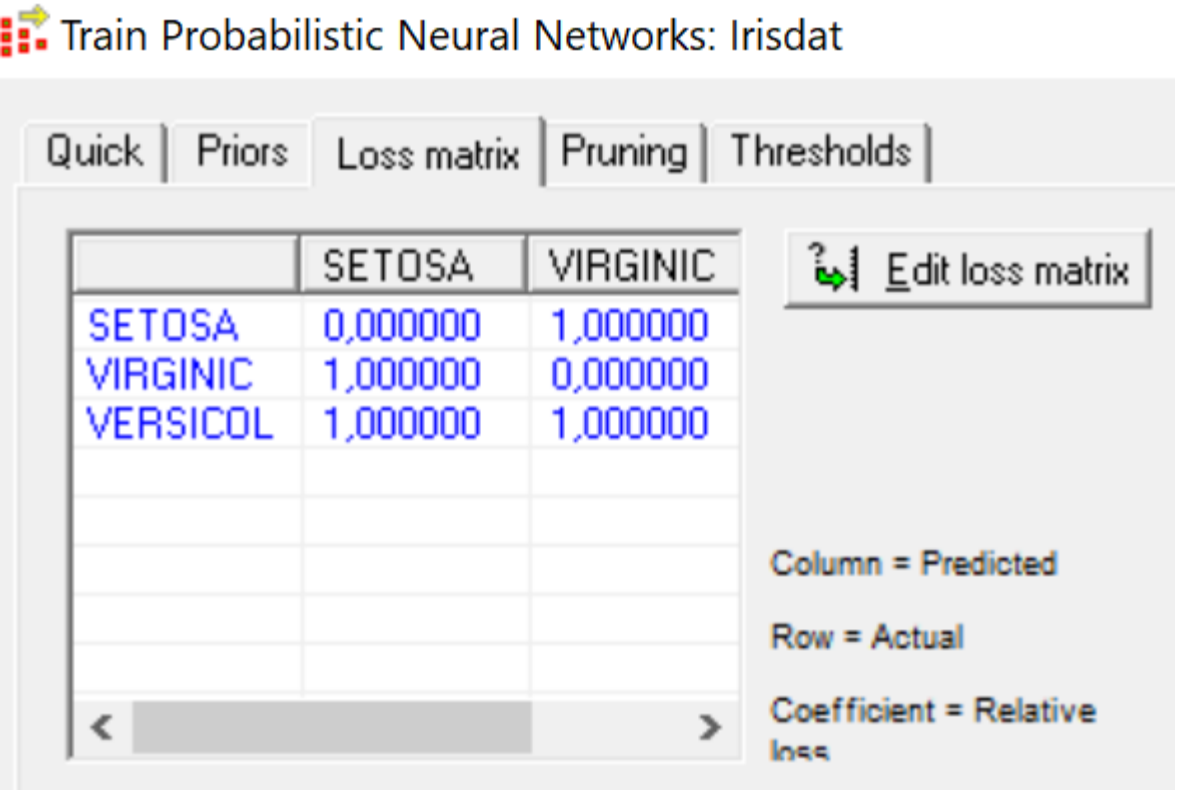


Having experimented with a range of smoothing factors, and created a PNN with an optimal selection error, you can execute it in the usual fashion. PNNs confidence levels are interpretable as class probability estimates.

Assuming that there is an element of noise in the data, the network (however good) will inevitably assign the wrong class sometimes. The basic PNN is designed to make as few mistakes as possible. However, in reality some misclassifications are often more expensive mistakes than others. You can compensate for this by adding a loss matrix.

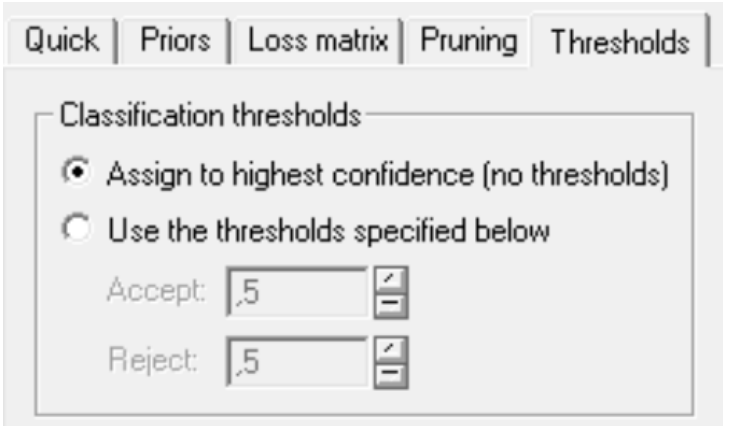
Now click OK and STATISTICA Neural Networks will create a four layer PNN. The fourth layer has the same number of units as the third, and contains the **loss matrix.**

On the Train Probabilistic Neural Networks dialog, click on the Loss matrix tab.



	SETOSA	VIRGINIC	VERSICOL
SETOSA	0	1	1
VIRGINIC	1	0	1
VERSICOL	1	1	0

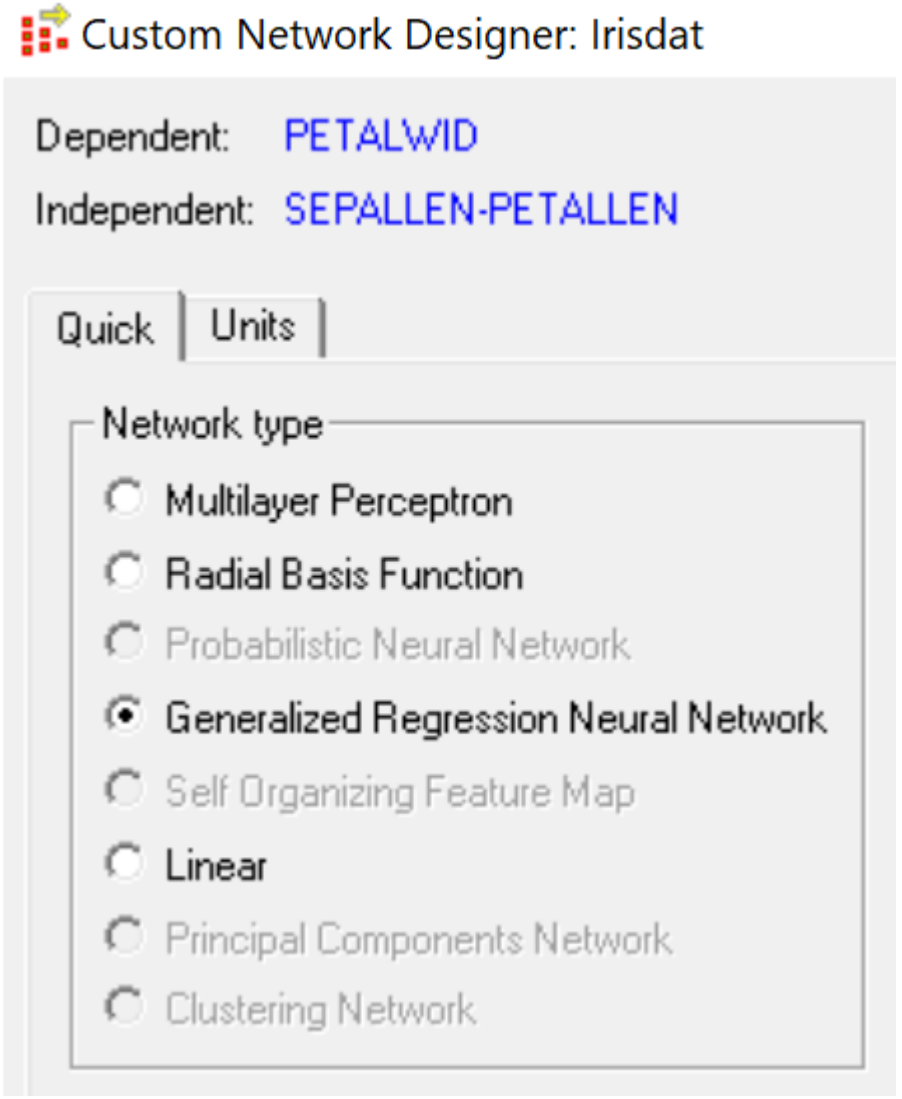
To establish importance!



The entries here represent the relative costs of various **types of misclassification**, with the columns representing the predicted class of the case, and the rows representing the actual class. The leading diagonal of the Loss Matrix always contains zeroes, as there is no cost involved in getting the right answer. You can alter the Loss Matrix to indicate misclassification costs. For example, setting the third column, second row to 10 (and leaving all other off-diagonals as 1) means that mistakenly classifying a Versicol as a Virginic is considered ten times as bad as any other type of mistake. Consequently, the number of correct classifications of Versicols should rise, at the cost of some misclassifications of Virginics.

When using a loss matrix, you are usually interested in making the minimum cost decision, and eschewing any "unknown" or "undecided" states. Click on the *Thresholds tab* *and make sure that the Assign to highest confidence (no threshold) option is selected.*

Generalized Regression Neural Networks. A generalized regression neural network (GRNN) is **used for regression problems**. For the purposes of this example, we will train a GRNN to predict the *PETALWID* variable of the *Irisdat.sta* data set from the *SEPALLEN*, *SEPALWID,* and *PETALLEN* variables.

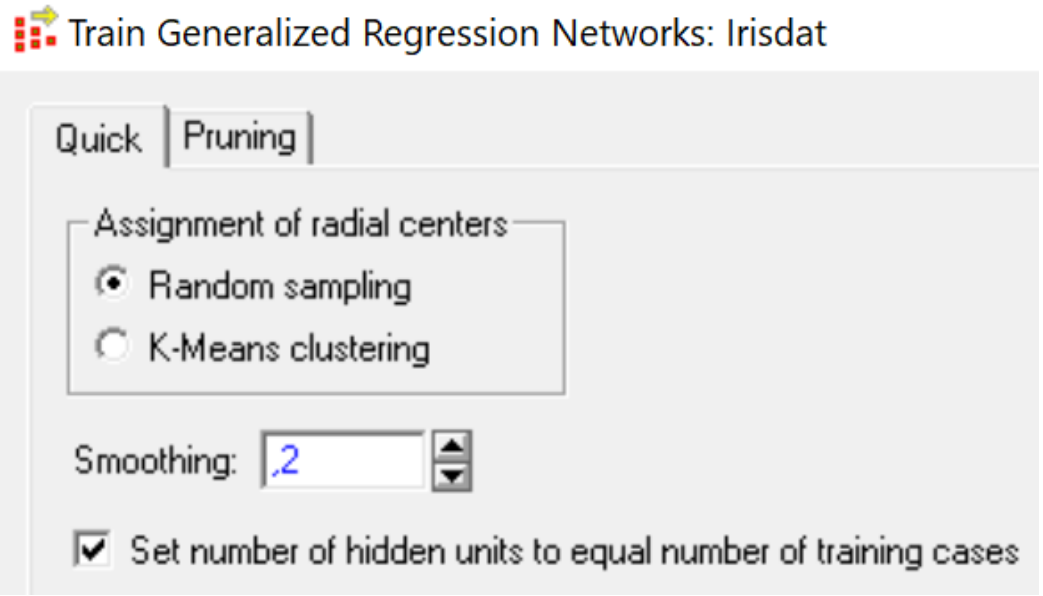


Select the **Regression option** button in the Problem type group box and select Custom Network Designer on the Neural Networks Startup Panel - Quick tab.

Click the Variables button, and select **PETLWID as the Continuous Output**, **and SEPALLEN, SEPALWID, and PETALLEN as the Continuous Input variables**.

By default, STATISTICA Neural Networks creates a GRNN with the same number of hidden units as training cases. However, unlike a PNN, a GRNN can optionally be specified with less units than training cases, and a clustering algorithm used to assign centers, and you can alter the number of hidden units on the Units tab.

However, usually the number of centers cannot be vastly less than the number of training cases without sacrificing performance (it might be half or a third, rather than an order of magnitude less). The simplest approach is to use the full set of training cases.



Make sure that the Set number of hidden units to equal number of training cases check box is selected (this is the default). If this is selected, the Assignment of radial centers options may be ignored.

Like a PNN, the single training factor to be selected is the Smoothing factor, which has the same meaning as in PNNs: it controls the deviation of the Gaussian kernel functions located at the radial centers. In this case, a smoothing factor of around 0.1 is adequate.

Click the OK button to run the GRNN training algorithm. This configures the network to perform kernel-based estimation.

When the network is trained, click the Descriptive Statistics button on the Results dialog to generate the regression summary statistics

Model Summary Report (Irisdat)

Index	Profile	Train Perf.	Select Perf.	Test Perf.	Train Error	Select Error	Test Error	Training/Members	Note	Inputs	Hidden(1)	Hidden(2)
7	GRNN 3:3-76-2-1:1	0,323290	0,319223	0,383294	0,414856	0,405275	0,491959	SS		3	76	2

	PETALWID	PETALWID.7
1	0,200000	0,283885
2	2,200000	1,759048
3	1,500000	1,689391
4	2,400000	1,880762
5	1,500000	1,708828
6	0,300000	0,280951
16	1,900000	1,613583
17	2,300000	1,926056
18	0,500000	0,292974

	Regression (7) (Irisdat)
	PETALWID.7
Data Mean	1,199333
Data S.D.	0,759693
Error Mean	0,024783
Error S.D.	0,258956
Abs E. Mean	0,213894
S.D. Ratio	0,340869
Correlation	0,941969