SNN Example 6: Advanced Use of the IPS

IPS -Classification Problem

Studying the networks created by the *Intelligent Problem Solver* can give you a very good picture of just what type of network performs well, and what input variables should be used. Running the *Intelligent Problem Solver* with the *Irisdat.sta* data file reveals that Multilayer Perceptrons perform best, that they tend to have a small number of hidden units (five or less), and that they have three or four inputs (with *SEPALLEN* optional).

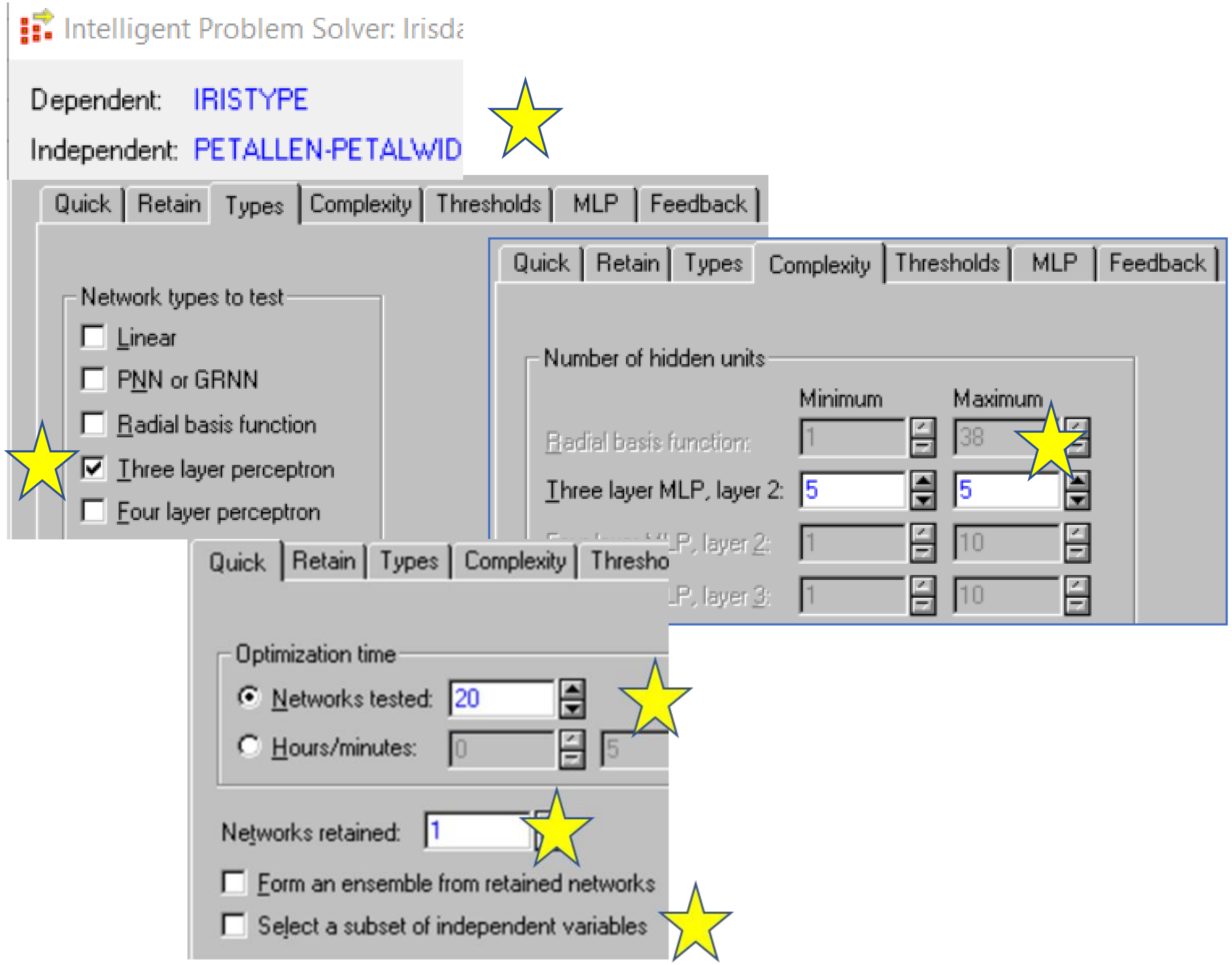
We could simply run the *Intelligent Problem Solver* in "brute force mode" to attempt to find an optimal network - say running it for **several hours**. In that time, it will try out thousands of networks on this relatively simple problem, and no doubt select some extremely good ones. However, the *Intelligent Problem Solver* can work much more efficiently if its design task is more constrained.

SNN Example 6: Advanced Use of the IPS

IPS -Classification Problem

Using the **advanced features** of the *Intelligent Problem Solver*, you can specify exactly the input variables, number of hidden units and network type, among other features. This can give you a much more accurate picture of what can be achieved. In problems with larger numbers of cases and variables, it can also help you to locate a very good solution with acceptable execution time.

	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)
1	RBF 1:1-28-3:1	0,907895	0,945946	0,837838	0,363839	0,349971	0,376537	KM,KN,PI		1	28	0
2	Linear 1:1-3:1	0,789474	0,810811	0,756757	0,349613	0,348854	0,334048	PI		1	0	0
3	Linear 3:3-3:1	0,802632	0,810811	0,837838	0,314011	0,331873	0,302111	PI		3	0	0
4	Linear 4:4-3:1	0,894737	0,864865	0,864865	0,304465	0,313651	0,287650	PI		4	0	0
5	RBF 2:2-28-3:1	0,855263	0,972973	0,972973	0,371588	0,264571	0,309672	KM,KN,PI		2	28	0
6	MLP 3:3-6-3:1	0,986842	1,000000	0,945946	0,149407	0,165215	0,336638	BP100,CG20,CG0b		3	6	0
7	MLP 4:4-8-3:1	0,98684[illegible]	1,000000	0,972973	0,155088	0,164108	0,300051	BP100,CG20,CG0b		4	8	0
8	MLP 4:4-8-3:1	0,98684[illegible]	1,000000	0,918919	0,131997	0,161949	0,396370	BP100,CG20,CG0b		4	8	0
9	RBF 3:3-28-3:1	0,960526	0,972973	0,918919	0,110280	0,147155	0,212798	KM,KN,PI		3	28	0
10	RBF 4:4-28-3:1	1,000000	0,972973	0,972973	0,086879	0,141561	0,165511	KM,KN,PI		4	28	0



Camplexity tab.

As an **example**, we will train a Multilayer Perceptron network with all four available inputs, and five hidden units. As we are looking for a very specific type of network, we will retain only the best network created by the *Intelligent Problem Solver*.

On the Quick tab, *clear the Select a subset of independent variables check box to ensure that all the networks tested use all four input variables. Click on the* Types tab *and select the Three layer perceptron check box (four layer networks are only occasionally needed, for some rare classes of difficult problem). Clear all the other check boxes.*

Click on the Complexity tab. Here you can specify how many hidden units should be used for each network type. You can specify a minimum and maximum, and STATISTICA Neural Networks will experiment with figures between these two limits. Enter the value 5 for both the Minimum and Maximum for Three layer MLP, layer 2. Specifying the same value for the minimum and maximum ensures that the size is fixed to a given number.

Index	Profile	Train Perf.	Select Perf.	Test Perf.	Train Error	Select Error	Test Error	Training/Members	Note	Inputs	Hidden(1)	Hidden(2)
1	MLP 2:2-5-3:1	0,973684	1,000000	0,891892	0,032588	0,025041	0,066513	BP100,CG20b		2	5	0

Model Summary Report (Irisdat)

If you run the *Intelligent Problem Solver* for a sufficient time period, you may reduce the selection error to 0.12 or even lower. However, the performance is unlikely to improve - there will always be a couple of cases that are misclassified, as there is some overlap between classes and so no technique can ever achieve perfect classification on this data set.

The existence of an (unknown) upper bound on performance (the so-called **Bayes error**) should always be borne in mind when designing neural networks.

A more valuable line of enquiry may be to try reducing the number of input variables and hidden units, in an attempt to produce a more compact, and therefore more reliable, network with equivalent performance.
Run the *Intelligent Problem Solver* again, this time without the *SEPALLEN* input variable. You are likely to discover that networks with equal, and possibly even slightly better, performance than the four input case can be found.

Even with just the *PETALLEN*, and *PETLWID* variables, it is possible (although more difficult) to find networks with extremely good performance, perhaps even with performance to match the three and four input networks. However, the relative difficulty in finding these networks is an indication that the class boundaries in two dimensions are more complex than in three (you can verify the complexity visually using the response surface window - a good solution with two inputs requires a sharp peninsula-shaped response surface), and therefore retaining at least one extra input variable is justified.