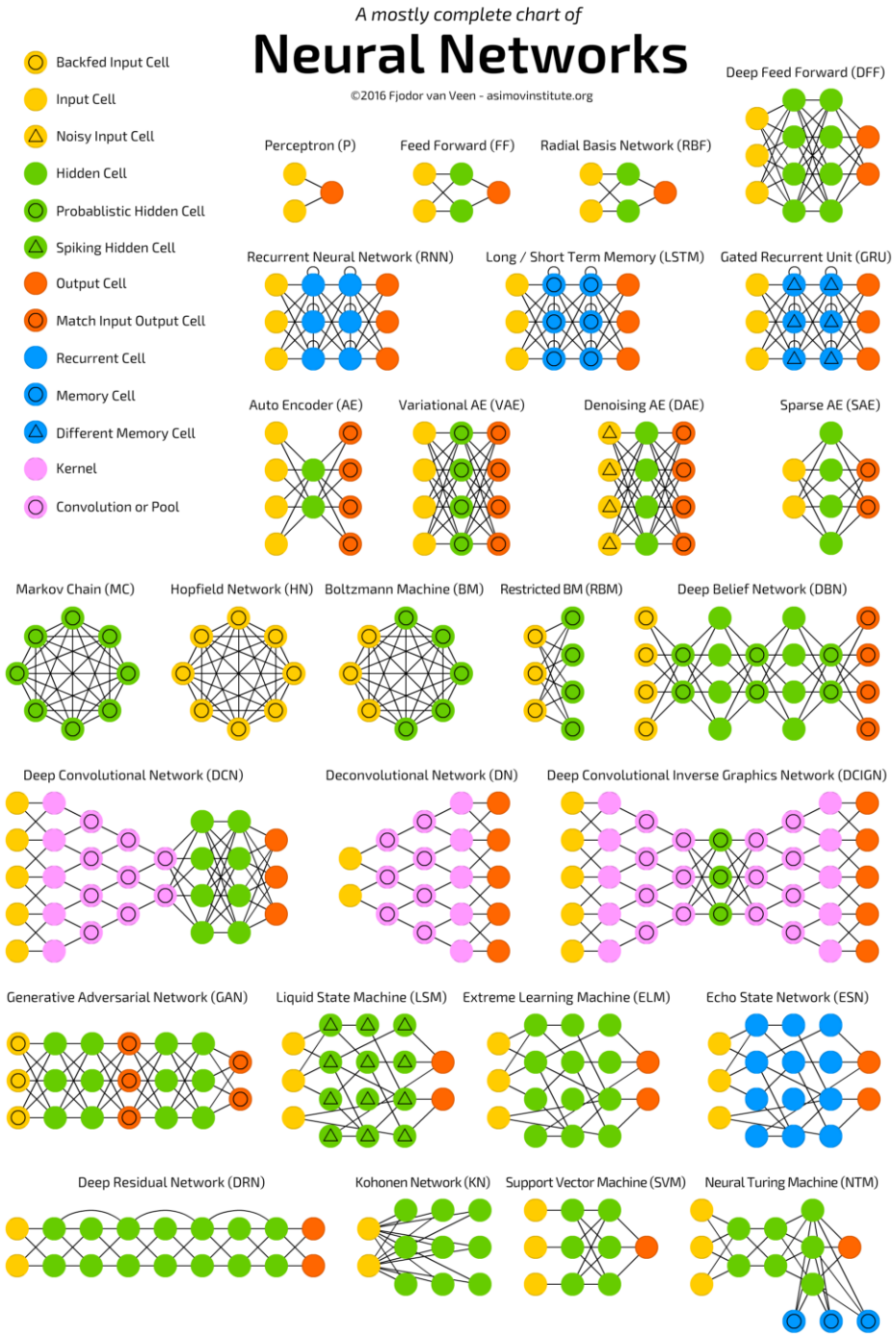
SNN Example 10: PNN and GRNN Networks

The standard neural network architectures (multilayer perceptrons and radial basis functions) infer a parameterized model (the weights forming the parameters) from available training data. The parameterized model (the network) is usually much smaller than the training data, and can be executed quite quickly, although the time taken to train the model may be long.

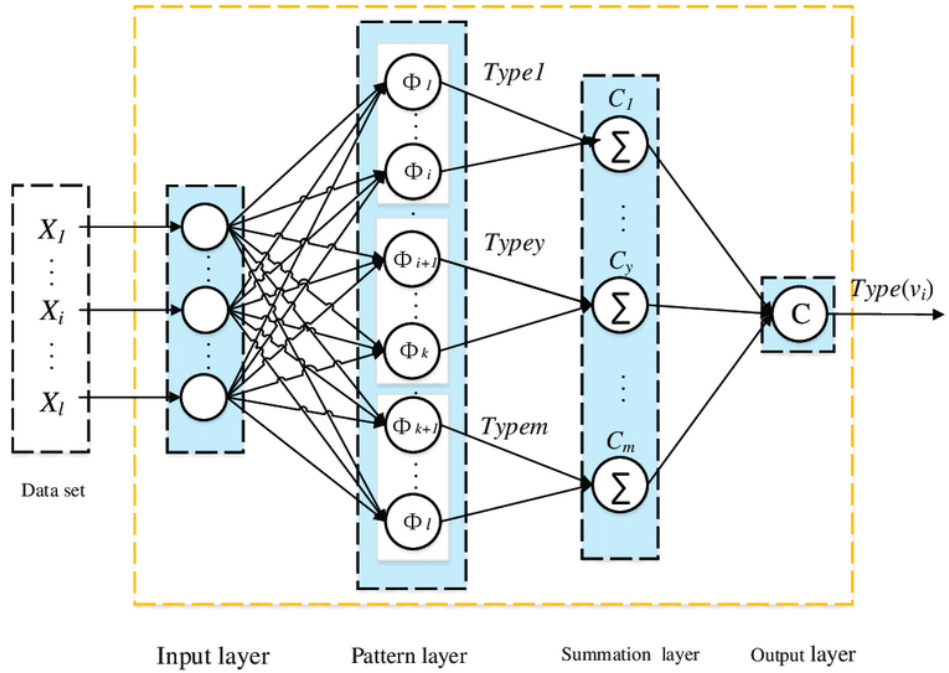
An alternative approach is to model the function more-or-less directly from the training data. This has the advantage that there is no need for training (or, at least, for "training" that is actually very simple, consisting of little more than changing the form in which the training data is held). The disadvantage is that the resulting model is large, and so consumes much memory and is relatively slow to execute.

Probabilistic Neural Networks (PNNs) and Generalized Regression Neural Network (GRNNs) are such methods, "**disguised**" as neural networks, and used for classification and regression respectively. The first layer of these networks contain radial units, which in the PNN actually store every training case, and in the GRNN store a large number of cluster centers (usually not vastly smaller in number than the training set). These radial units have an output activation that is a Gaussian function centered at the stored point (and hence acts, as is were, as "evidence" that the modeled function has some probability density distributed around that point). Subsequent layers combine these outputs into estimates of class probabilities (in the case of PNNs) or of the regression value (in the case of GRNNs).

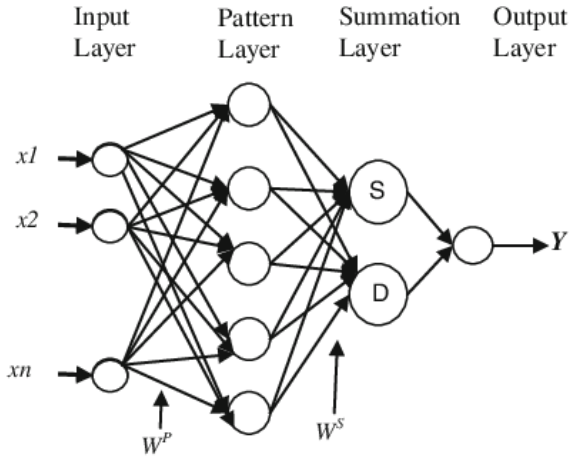
If you have a reasonably small number of cases (say, 500 or less) then it is well worth considering the use of a PNN or GRNN.

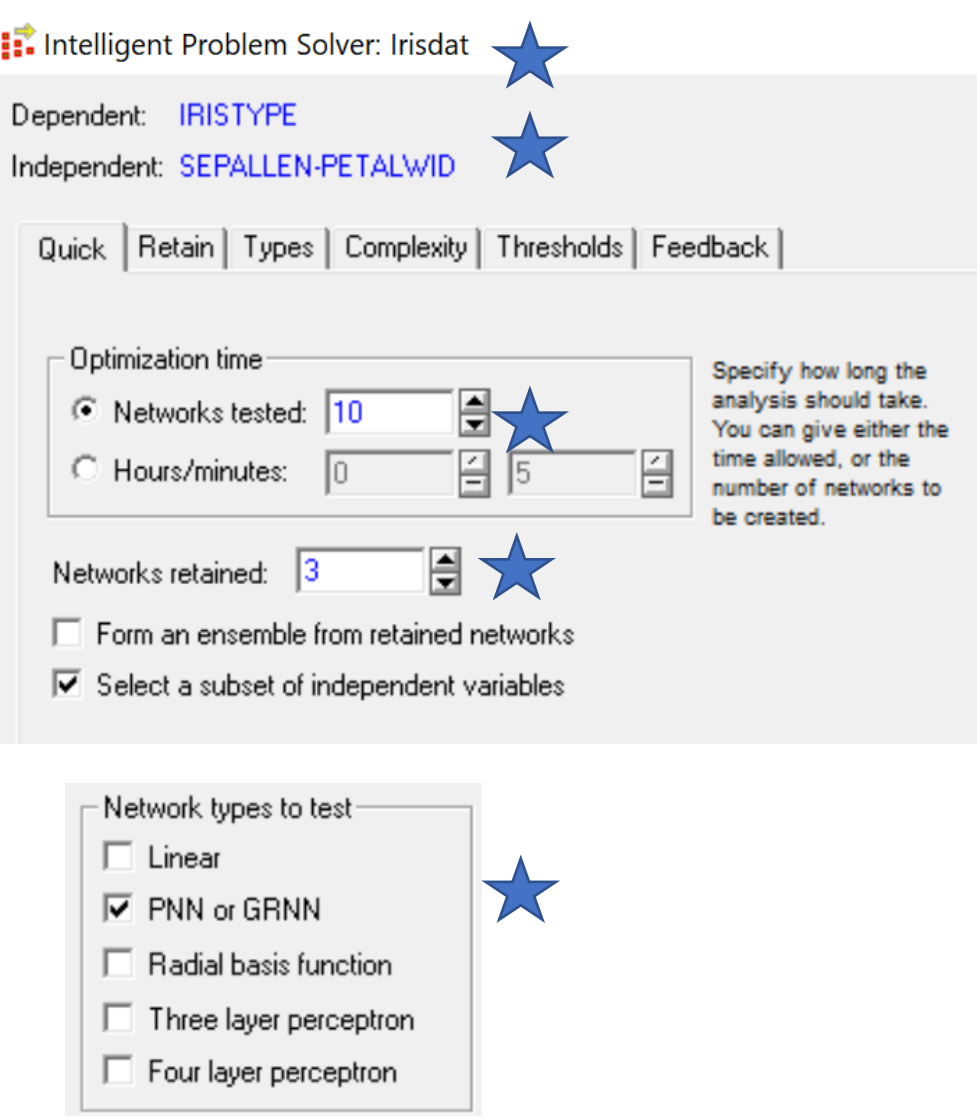


https://www.digitalvidya.com/blog/types-of-neural-networks/



PNN e GRNN nem aparecem como Redes Neurais



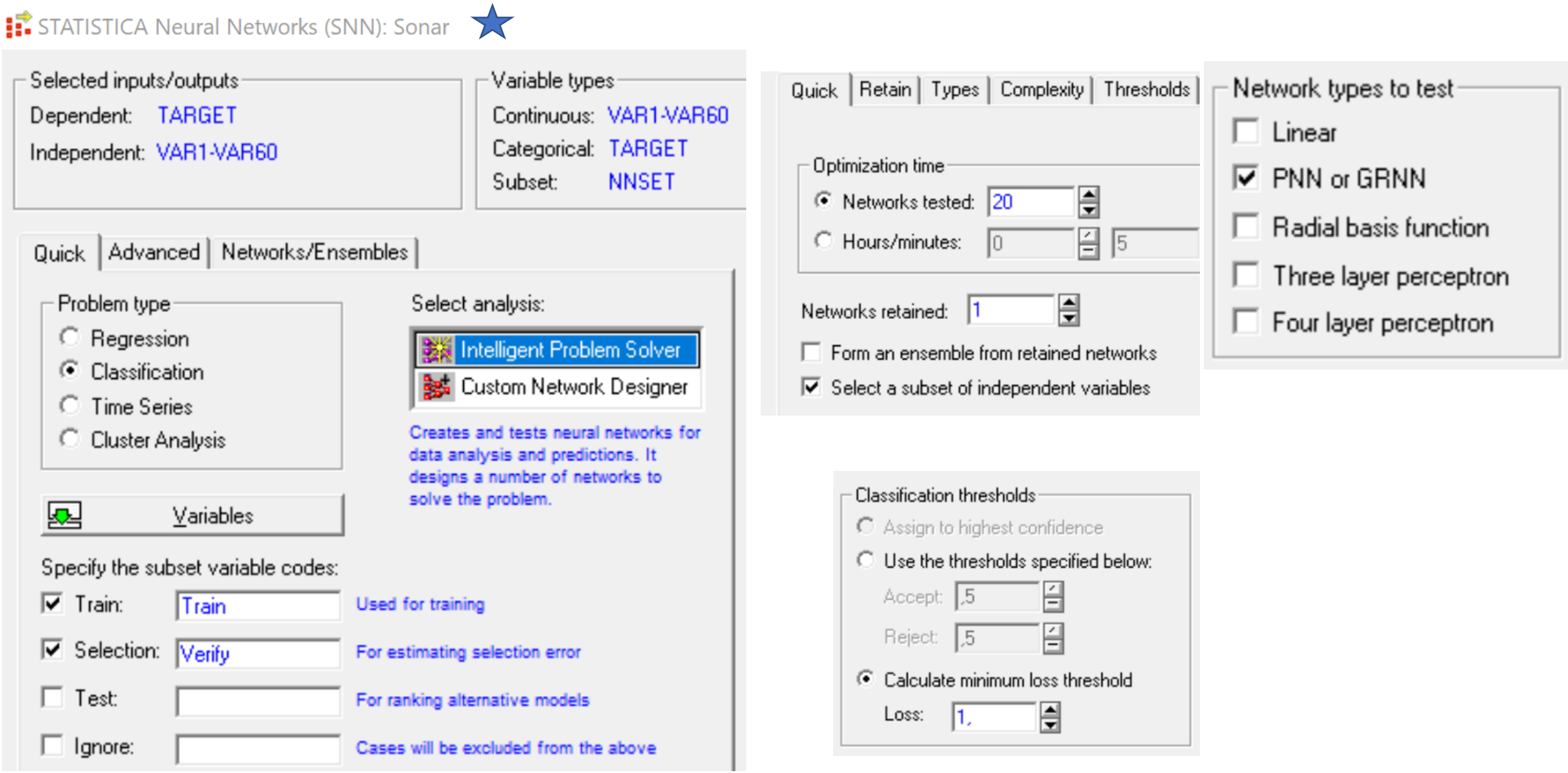


The search should be extremely fast, and the best network is likely to have three inputs (possibly four, depending on the division of your training set).

The very fast training time makes PNNs and GRNNs extremely useful for **initial experiments**, and performance is usually broadly comparable with the other network types. In most problem domains Multilayer Perceptrons seem to achieve somewhat better performance, and are also far more compact. However, this is not always the case.

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	PNN 4:4-76-3:1	1,000000	1,000000	0,945946	0,002809	0,060916	0,165289			4	76	0
2	PNN 4:4-76-3:1	1,000000	1,000000	0,945946	0,011357	0,060062	0,158027			4	76	0
3	PNN 3:3-76-3:1	1,000000	1,000000	0,972973	0,063131	0,059813	0,131993			3	76	0


STATISTICA Neural Networks (SNN): Sonar
Selected inputs/outputs
Dependent: TARGET
Independent: VAR1-VAR60
Variable types
Continuous: VAR1-VAR60
Categorical: TARGET
Subset: NNSET
Quick
Advanced
Networks/Ensembles
Problem type
Regression
Classification
Time Series
Cluster Analysis
Select analysis:
Intelligent Problem Solver
Custom Network Designer
Creates and tests neural networks for data analysis and predictions. It designs a number of networks to solve the problem.
Variables
Specify the subset variable codes:
Train: Train
Used for training
Selection: Verify
For estimating selection error
Test:
For ranking alternative models
Ignore:
Cases will be excluded from the above
Quick
Retain
Types
Complexity
Thresholds
Optimization time
Networks tested: 20
Hours/minutes: 0 5
Networks retained: 1
Form an ensemble from retained networks
Select a subset of independent variables
Network types to test
Linear
PNN or GRNN
Radial basis function
Three layer perceptron
Four layer perceptron
Classification thresholds
Assign to highest confidence
Use the thresholds specified below:
Accept: ,5
Reject: ,5
Calculate minimum loss threshold
Loss: 1,

Model Summary Report (Sonar)

Index	Profile	Train Perf.	Select Perf.	Test Perf.	Train Error	Select Error	Test Error	Training/Members	Note	Inputs	Hidden(1)	Hidden(2)
1	PNN 55:55-103-2-2:1	0,893204	0,759615	0,00	0,603071	0,556322	0,00			55	103	2

The *Intelligent Problem Solver* should discover a network with approximately fifty input variables, and a performance rating of 0.85 or above. This performance is quite sensitive to the data set division; however if you repeat the experiment, reassigning the subsets each time, you may get quite different figures. This illustrates the dangers of building models with a large number of input variables and a small number of cases.

Note: A feature of PNNs is that, with the right control parameters, they almost invariably achieve a 100% correct classification rate on the training set. This should not be too surprising, as the technique models a probability **density function** by locating a Gaussian peak at each training case, and if these peaks are sharp enough (and there are not two cases of different classes exactly coincident) then this 100% performance can be guaranteed. Unfortunately, this tells us nothing about the generalization performance of the networks. Even more so than with other types of network, you should not allow performance on the training set to influence you. Assess performance by considering the selection and test sets only.