

Latin Hypersquare Sampling and Genetic Algorithm in SAS/IML® for Hyperparameter Optimization

Andrea Magatti, BIP Business Integration Partners

ABSTRACT

A Hyperparameters Optimization approach for SAS® 9.4 users that do not have access to the “magic” AUTOTUNE option of SAS/VIYA®, using SAS/IML®, some SAS/STAT® and SAS/OR®.

INTRODUCTION

In this document, I’m going to explain how to, practically, replicate one of the best feature available in SAS® Viya®: the AUTOTUNE statement available for some procedures like NNET for Neural Networks.

This approach shows how to:

- Build a factorial design, for a Neural Network model
- Sample the factorial design, using an implementation of LHS-MDU in SAS/IML®
- Grow the best points, obtained from the reduced grid design, with a Genetic Algorithm in SAS/IML®, and running another, smaller, grid search

This paper is aimed to users with good practice with some math, and capability of using SAS/BASE® and SAS/IML®, understanding some procedures of SAS/STAT® and the logic of DOE (Design of Experiments)

DESIGN OF EXPERIMENTS AND LATIN HYPERSQUARE SAMPLING

THE PROBLEM

Any machine learning project needs to keep in mind that the default settings of a parameter of some SAS® procedures, are just a starting point to make some trials and experiment with the specific procedure. For some procedures instead, those parameters are fundamental to achieve good performance.

In this case, we will talk about the hyperparameters that specify the architecture of a Neural Network; in more details, we are talking about the HPNEURAL procedure.

Then HPNEURAL implements a specific kind of Neural Network, as specified by Bishop, C. M. (1995) in the paper “Neural Networks for Pattern Recognition. Oxford: Oxford University Press”

This kind of neural network is an MLP network (multilinear perceptron), that has one or more Hidden Layers, between the Input and Output layer.

In Figure 1 is explained the MLP architecture, where the Input Layer nodes are the input variables provided, and the Output Layer nodes are the target. Each layer is activated by a specific function to exclude linearity. The HPNEURAL has the following activation functions: COS, TANH, and ID

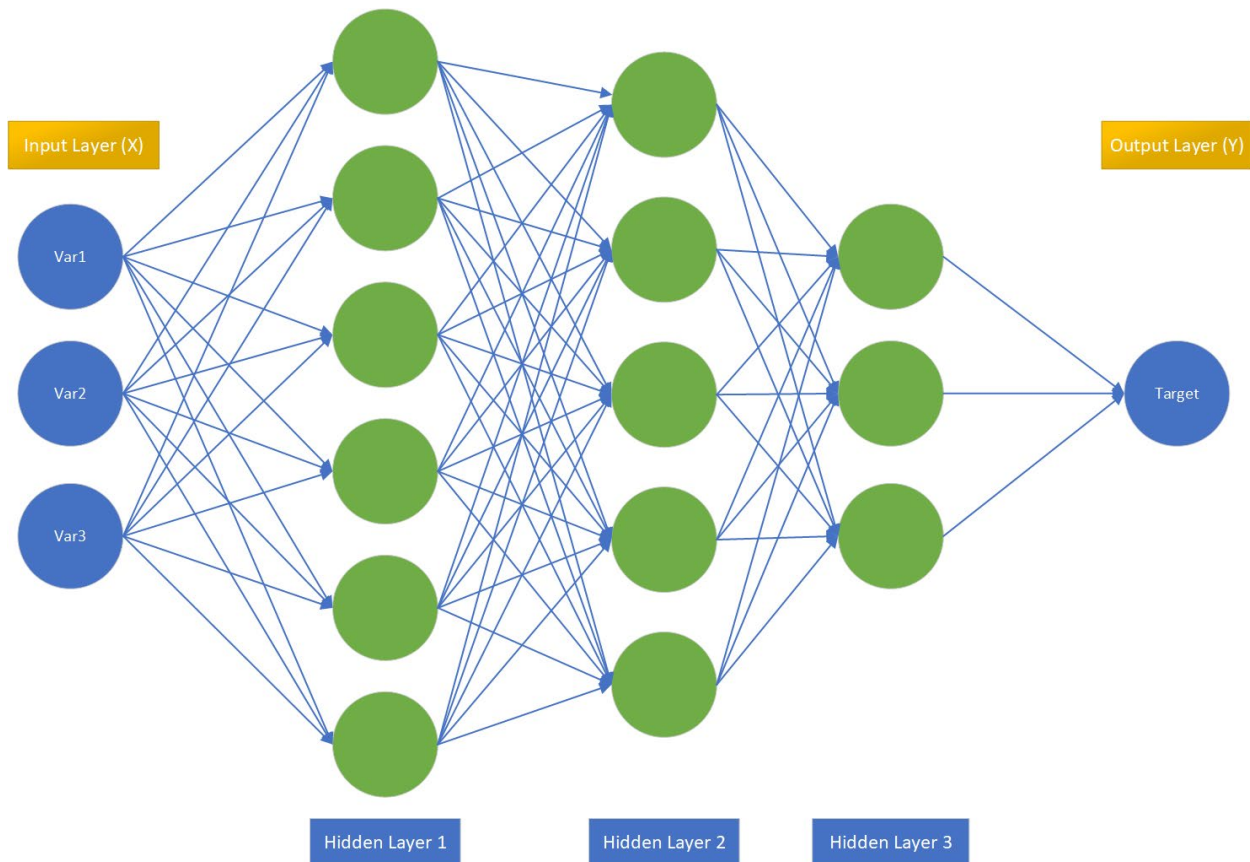


Figure 1

Just looking at the above picture you can get the problem: Which is the right combination of layers, perceptrons, and an activation function, that can optimize the desired target?

In this example, the Neural Network has three hidden layers, with six perceptron in the first layer, five perceptron in the second and 3 in the third layer.

When talking about combinations, and combinatorial calculus, in a real-world scenario, we are effectively talking about Design of Experiments (DOE)

DESIGN OF EXPERIMENTS

In simple words, DOE is a method to inspect the variation of information, subject to variations of input conditions.

In our case, it is intended to create a factorial matrix of any hyperparameters associated with the Neural Network optimization.

The hyperparameters used are:

- Hidden Layer's number (1 to 6)
- Perceptron's number for each hidden layer (from 0 through 2 up to 2^8)
- Activation function for each hidden layer (TANH, COS, ID)
- Training dataset length, i.e.: *horizon* (36, 48, 54, 60 months)

- The threshold of explained variance used by the HPREDUCE procedure, i.e., *varexp* (0,97 to 0.985 with a step of 0.005, for a total of 4 steps)

For our test case, we assume that 6 Hidden Layers are enough to capture the variability of the forecasting problem. This number can be obtained through a random test, finalized to understand the range of variations that need to be assumed to avoid the excessive size of the grid matrix.

The first three hyperparameters define the architecture of the neural network, while *horizon* and *varexp* are relative to the training process. That means that design with the inner and outer design of the experiment is needed. Byrne, D. M., and Taguchi, S. (1986) defined this approach, dividing factors between Control Factors and noise factors.

For our purposes, the control factors are typically those, falling in the domain external to the Neural Network procedure (*varexp and horizon and activation function*), while we call noise factors those hyperparameters affecting the architecture (*hidden layers and perceptron*).

To build such a matrix, we are going to use the capability of the FACTEX procedure that, as the first step, creates orthogonal factorial experiment design.

The full code is available, but below are presented and explained some code snippets:

```
proc factex;
  factors H1 H2 H3 H4 H5 H6 / nlev=9;
  output out=OuterArray
    H1 nvals= (0 2 4 8 16 32 64 128 256)
    H2 nvals= (0 2 4 8 16 32 64 128 256)
    H3 nvals= (0 2 4 8 16 32 64 128 256)
    H4 nvals= (0 2 4 8 16 32 64 128 256)
    H5 nvals= (0 2 4 8 16 32 64 128 256)
    H6 nvals= (0 2 4 8 16 32 64 128 256);
run;
```

This code produces a full factorial grid of 9^6 rows, and after removing senseless combination (i.e. a combination with 0 perceptron in the first layer means that there is no hidden layer), the grid is reduced to 299,592 combinations.

When examining the joint distribution of this matrix (random sample with sample rate of 0.1), (Figure 2) using one of the new procedures available on SAS Viya®, the TSNE procedure, it appears that the layers/perceptron distribution is remembering something like a galaxy cluster, due to layers number (6) and the neuron's number built as a power of 2, from 2 to 8.

Such kind of weird space filling is a problem, with this kind of design, since the spatial distribution is not optimal, for our purposes

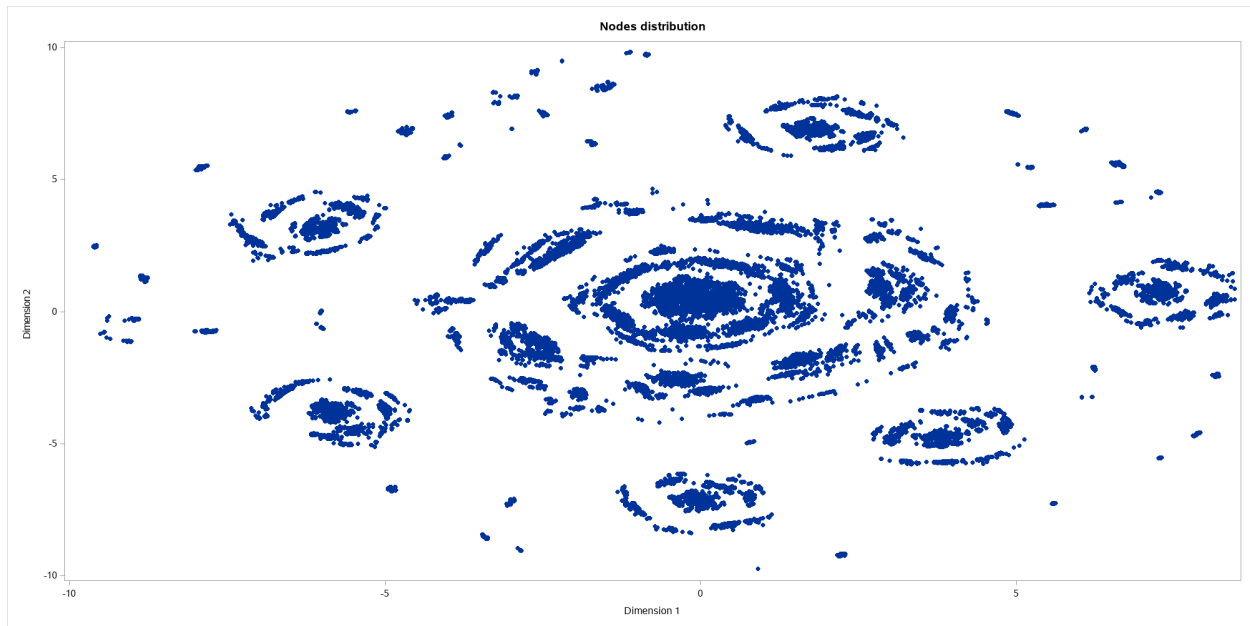


Figure 2

This kind of plot can be made using the TSNE procedure of Viya® 3.4 (SAS Visual Data Mining and Machine Learning: Procedures) with this code:

```
proc tsne
  data          = casuer.design
  nDimensions   = 2
  perplexity    = 5
  learningRate  = 100
  maxIters      = 500;
  input         H: ACT: VAR: ;
  output out    = causer.tsne_out copyvars=(new_id;
run;
```

and the plotting with the SGLOT procedure:

```
proc sgplot data=tsne_out;
  title "Hyperparameters embedding";
  scatter x=_DIM_1_ y=_DIM_2_ / markerattrs=(symbol=CircleFilled);
run;
```

The other way to plot something like this, without using the VIYA procedure, is achievable using the MDS procedure of SAS® 9.4 and the DISTANCE procedure.

At first, we need to calculate the Euclidean distance matrix between variables:

```
proc distance data=test method=EUCLID out=distances;
  var interval(CDF: );
  id new_id;
run;quit;
```

And then reduce the dimensionality using Multidimensional scaling of the MDS procedure:

```
ods graphics on / height=1000 width=1200;
proc mds data=distances level=absolute out=outdim;
  id new_id;
run;
```

Finally, we can plot the scatter distribution of the new calculated variables:

```
ods graphics on / height=1000 width=1200;
Title "MDS";
proc sgplot data=outdim;
  scatter x=Dim1 y=Dim2 / markerattrs=(symbol=CircleFilled);
run;
```

The next step is to replicate the outer array, where each of the observation of the first matrix (Outer Array) is replicated, for each combination of the activation functions:

```
proc factex;
  factors Horizon VarExp Activation /
    nlev=3;
  model resolution=maximum;
  output out=DesignFF pointrep=newArray
    Horizon nvals= (36 42 48)
    VarExp nvals= (.975 .980 .985)
    Activation cvals= ('TANH' 'IDENTITY' 'LOGISTIC');
run;
```

After this last step, the grid design has grown to a staggering number of 8,088,984 realizations, and in a real-world situation (simulation of one year of the hourly forecast) that is a prohibitive number for a feasible grid search.

Furtherly, it's needed to clean up the matrix, since it shows categorical variables, that are not usable by any spacing methods based on proximity.

To vectorize the Activation variable, I'm making use of a trick, using the TRANSREG procedure:

```
proc transreg data=designFF design;
  id H: Horizon VarExp;
  model identity(id)
    class (Activation / zero=sum);
  output out= designMatrix (drop=_TYPE_ _NAME_ Intercept Activation);
run;
```

With this kind use of the TRANSREG procedure, the categorical variable, activation, has been transformed in a vector of vars, with a new var for each level, with the value 0 (not present) and 1 (present)

Before diving into the LHS logics, it's useful to understand the multidimensional space we have built, with the Inner Array and Outer Array design. In the next picture, Figure 3, is

displayed the multidimensional structure of any hyperparameters, (perceptron, hidden layer, horizon, and varexp):

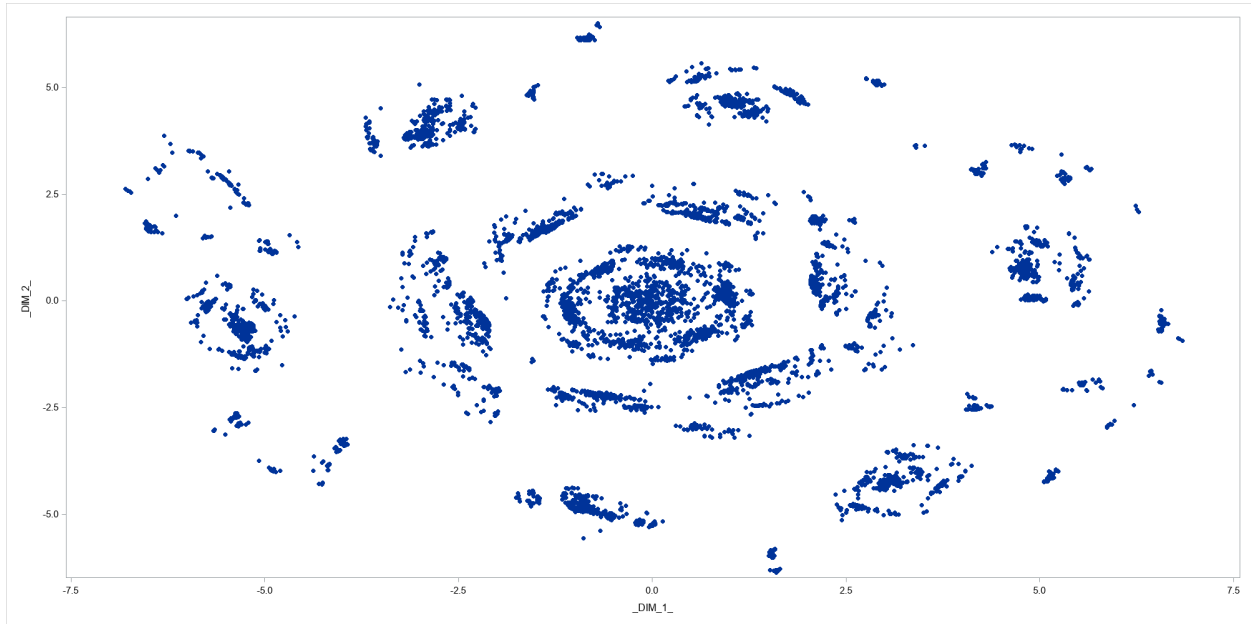


Figure 3

The Figure 3 shows that in some way the space filling of this design has changed a bit from the "galaxy" structure of Figure 2, due to the discrete nature of the Inner Array variables (varexp and horizon) and the Boolean nature of the vectorized Activation variables.

It is now clear that this DOE, built using traditional methods, can't simply ensure a good spatial uniformity, and even a full grid search with this kind of design, can't ensure to find an absolute minimum (i.e., MAPE, etc.). So it is useful to remember one of the Euler or Chai Seok-Jeong, first publication (around 1700 ac) about Latin Square, that is a matrix where that is an $n \times n$ array filled with n different symbols, each occurring exactly once in each row and exactly once in each column.

A JOURNEY TOWARD LATIN HYPERSQUARE SAMPLING

A mathematical primer to LHS is simply achieved remembering the old Japanese game "Sudoku". This simple game explains the basic behind Latin Hypersquare, that is a " $n \times n$ " array filled with symbols each occurring exactly once in each row and exactly once in each column. The "hyper" prefix means that we are not dealing with a bidimensional array but with a multidimensional array, elsewhere called "hypercube".

LHS sampling is typically used where a model can be realized a limited number of times, due to large computational requirement. That is exactly our case study since, running, or realizing 8,088,984 test case is not possible in most cases and doesn't ensure multi-dimensional uniformity.

The original LHS concept was developed by McKay, Conover, Beckman (1979, 1992) as a method to overcome the limits of traditional stratified sampling techniques.

Without digging in the mathematical details, that are available in the cited paper, it is possible to understand the available methods to reduce the number of realizations (observations) to obtain a significative reduction of computing time.

Random Sampling

Random Sampling has always been the obvious method to reduce the population number, and there are plenty of methods in the literature that can be used to understand the distribution. In the following code is presented a sample SAS® code that uses the SURVEYSELECT procedure to get a sample of the full population:

```
proc surveyselect data= designMatrix
  method=srs ranuni
  seed=220810
  out= designMatrix_sample
  samprate=.0025
  outorder=random
  noprint;
run;
```

Stratified Sampling

Stratified Sampling, is surely a better refinement to Random Sampling since the entire space is partitioned in I strata, and then a random sample is extracted from each partition. In the following code is presented a sample SAS® code that uses the SURVEYSELECT procedure to get a stratified sample of the full population, using a two-level stratification using the Horizon and VarExp hyperparameters:

```
proc surveyselect data= designMatrix
  method=srs ranuni
  seed=220810
  out= designMatrix_sample
  samprate=.0025
  outorder=random
  noprint;
  strata Horizon VarExp;
run;
```

Latin Hypersquare (Hypercube) Sampling

Latin Hypersquare Sampling is a logical extension of the Stratified Sampling where each of the input variables has a portion of its distribution so that we can divide the space of each variable into N strata, with an equal marginal probability, and the sample from each stratum. The N intervals of each variable combine to create N^k cells which cover the sample space. These cells are then labeled with coordinates corresponding to the interval to examine the sampling schema.

Latin Hypersquare Sampling with Multidimensional Uniformity

LSHMDU is a variation of basic LHS that enforces multidimensional uniformity of the input parameters through sequential realization elimination.

The mathematical details of this method are available in the paper of Deutsch and Deutsch, "Latin hypercube sampling with multidimensional uniformity (2012), available in the reference sections.

LHSMDU works mostly in the same way as plain LHS, but before to apply a fully stratified sampling LHSMDU impose a degree of multidimensional uniformity on the sampling matrix.

A realization elimination algorithm is used to enforce a degree multi-dimensional uniformity. Initially, the program generates realizations ($L \times M$) where M is a small number greater than one making the initial sampling matrix M times larger than the final, and L is the desired number of samples.

Each of the ML realizations is composed of N uniform random numbers in the range of 0 to 1. The average Euclidean distance of each realization to its two nearest neighbors is calculated to eliminate realizations.

The algorithm takes the two nearest neighbors instead of the single nearest neighbor to prevent realization pairs from being generated.

The algorithm eliminates realizations with the smallest average distance to its two nearest neighbors. The program repeats this distance calculation and elimination scheme until only L realizations remain. If the variables are assumed independent, then the realizations are ranked, and the algorithm applies the traditional LHS stratification scheme on the samples.

The SAS/IML program that implements LHSMDU follows this schema: Figure 4

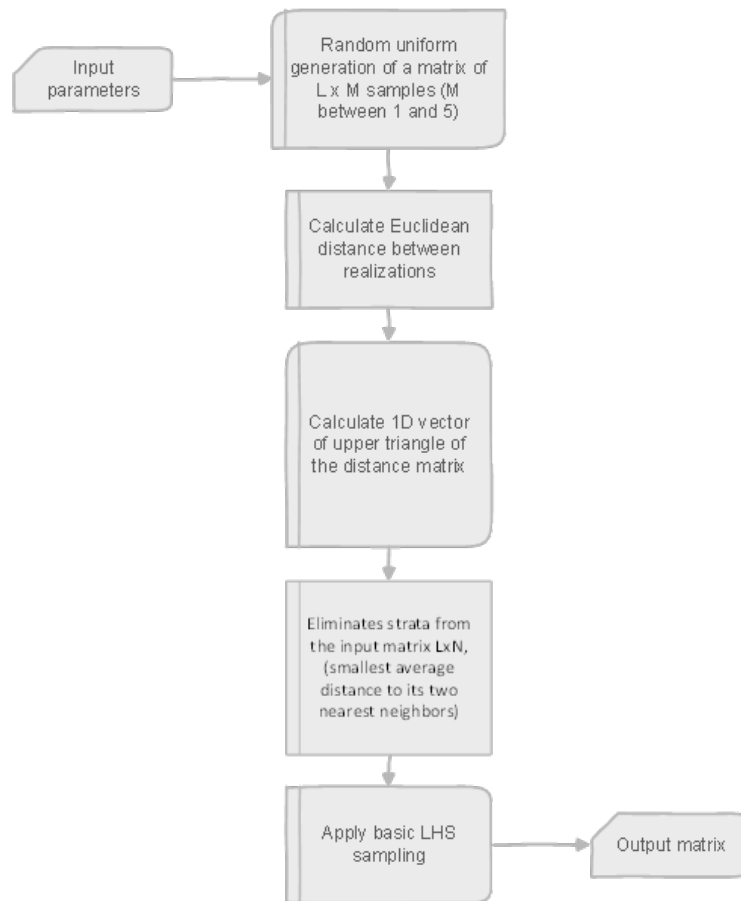


Figure 4

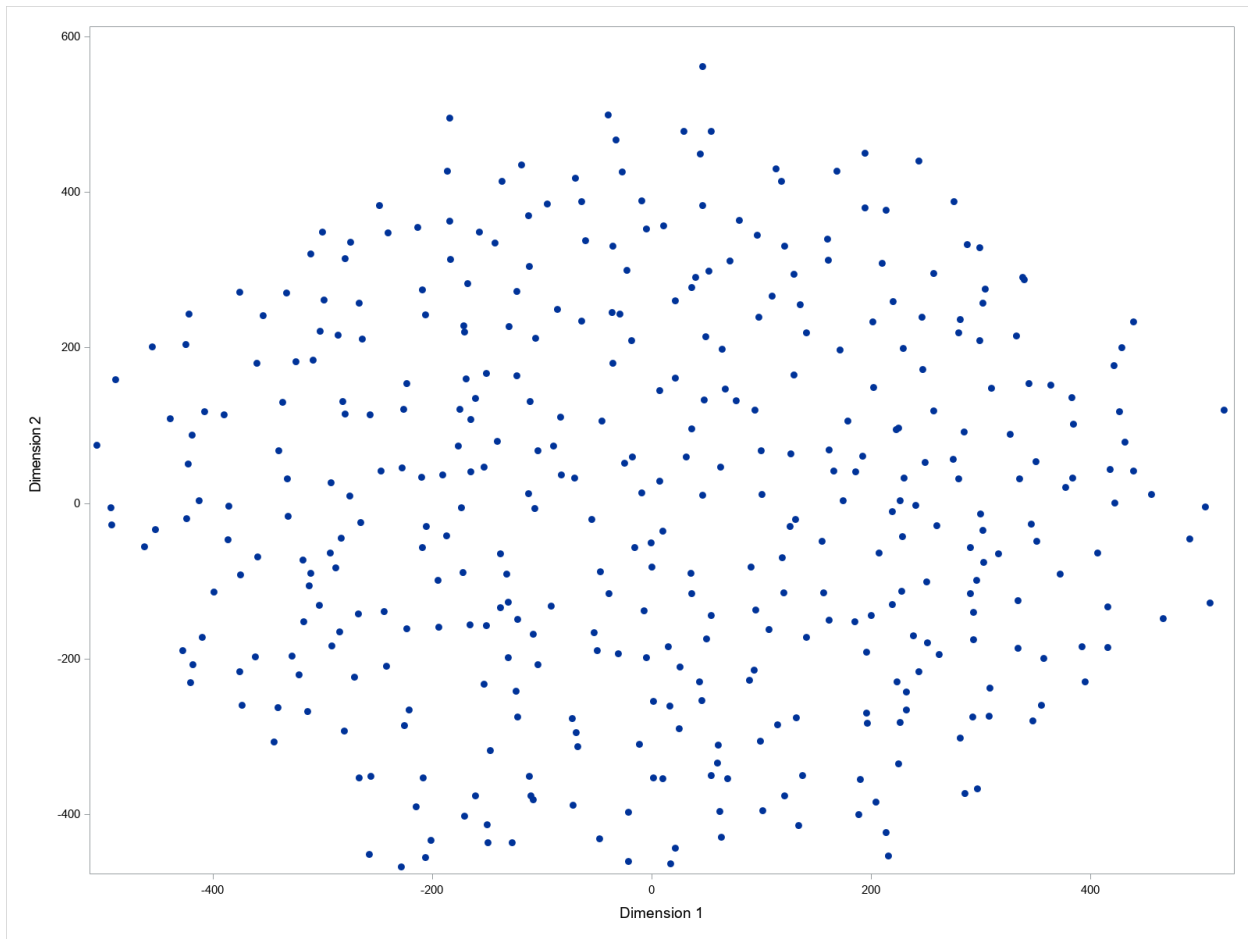


Figure 5

In Figure 5 are plotted 400 samples, for 11 variables. The variables are the hyperparameters used by HPNEURAL procedure:

- Hidden Layer's number (1 to 6)
- Perceptron's number for each hidden layer (from 2 up to 512)
- Activation function for each hidden layer (TANH, COS, ID)
- Training dataset length, i.e., *horizon* (from 36 to 60 months)
- The threshold of explained variance used by the HPREDUCE procedure, i.e., *varexp* (0,97 to 0.985)

The IML program will compute the LHSMDU sampling building a dataset for N dimensions (hyperparameters) and 400 realizations (samples). Each number produced is a continuous number between 0 and 1, so you need to rescale any variable to align the number to the range of any hyperparameter. To fix this natural, characteristics, we can use this snippet code, using the QUANTILE function:

```

data LHSMDU_hyperparms (keep=CDF_ );
  Fa = 2;
  Fb = 512;
  Act_fa = 0;
  Act_fb = 1;
  Hor_fa = 24;
  Hor_fb = 52;
  Var_fa = 975;
  Var_fb = 985;
  set lhsmdud_input;

  CDF_H1 = round(Fa + quantile("Uniform", H1)* (Fb-Fa) );
  CDF_H2 = round(Fa + quantile("Uniform", H2)* (Fb-Fa) );
  CDF_H3 = round(Fa + quantile("Uniform", H3)* (Fb-Fa));
  CDF_H4 = round(Fa + quantile("Uniform", H4)* (Fb-Fa));
  CDF_H5 = round(Fa + quantile("Uniform", H5)* (Fb-Fa));
  CDF_H6 = round(Fa + quantile("Uniform", H6)* (Fb-Fa));
  CDF_ACT_ID = round(Act_fa+quantile("Uniform", ACT_ID)*(Act_fb-Act_fa));
  CDF_ACT_TA = round(Act_fa+quantile("Uniform", ACT_TA)*(Act_fb-Act_fa));
  CDF_ACT_LO = round(Act_fa+quantile("Uniform", ACT_LO)*(Act_fb-Act_fa));
  CDF_HOR = round(hor_fa+quantile("Uniform", HORIZON)*(hor_Fb-hor_fa));
  CDF_VAR = round(Var_fa+quantile("Uniform", VAR)*( Var_Fb-Var_fa))/1000;
  new_id = put(_N_, z4.);
run;

```

It is necessary to examine the effective distribution of each hyperparameter checking the distribution of each variable using the UNIVARIATE and FREQ procedure, with this kind of code:

```

proc univariate data=test;
  var CDF;;
  histogram CDF_H1/ endpoints=-50 to 560 by 1;
  histogram CDF_H2/ endpoints=-50 to 560 by 1;
  histogram CDF_H3/ endpoints=-50 to 560 by 1;
  histogram CDF_H4/ endpoints=-50 to 560 by 1;
  histogram CDF_H5/ endpoints=-50 to 560 by 1;
  histogram CDF_H6/ endpoints=-50 to 560 by 1;
  histogram CDF_ACT_ID/ endpoints=0 to 1 by 1;
  histogram CDF_ACT_TA/ endpoints=0 to 1 by 1;
  histogram CDF_ACT_LO/ endpoints=0 to 1 by 1;
  histogram CDF_HOR/ endpoints=24 to 52 by 1;
  histogram CDF_VAR/ endpoints=.975 to .985 by .001;
run;

```

In the next plots, we can see the histogram to understand if the random generation of the number between 0 and 1, the LSHMDU algorithm has been able to produce multidimensional variables. For the sake of simplicity, we have plotted only a subset of the available hyperparameters.

In Figure 5 is plotted the distribution of the number of neurons of the first hidden layer, with number spanning from 2 to 512. It is possible to see some occurrences with more than 25% distribution since the number of the initial sample (400 realizations) and the rounding of the previous code can change the "real" distribution

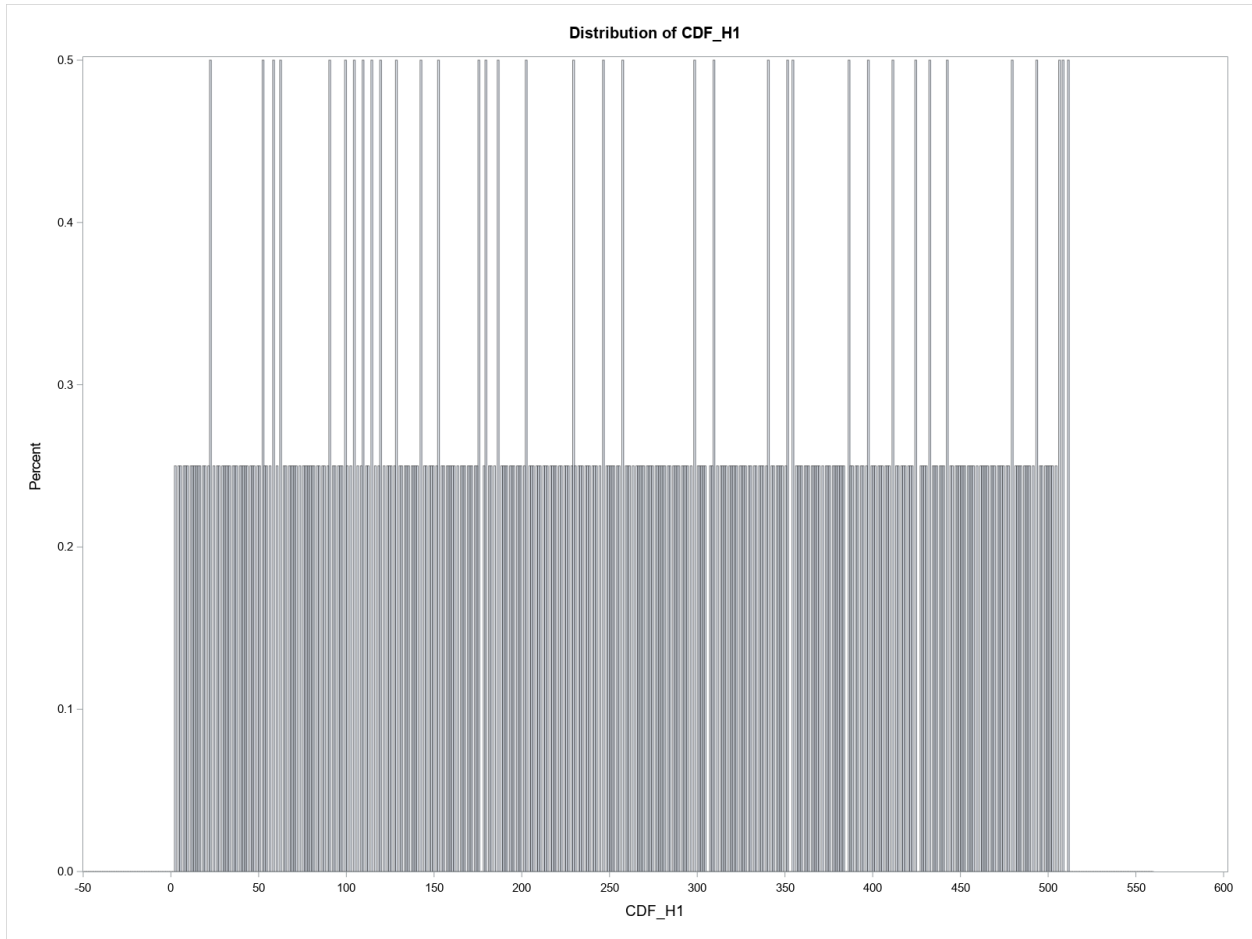


Figure 6

In Figure 6, we can see the distribution of the Activation function Identity used for any Hidden layer. In this case, due to the boolean characteristic of this variable, the distribution is perfectly uniform, as requested



Figure 7

In the last plot, Figure 8, shows the distribution of the Horizon hyperparameter. The distribution is mostly uniform, beside the first and last occurrence. The rounding function, of the previous code, is the probable cause of this behavior.

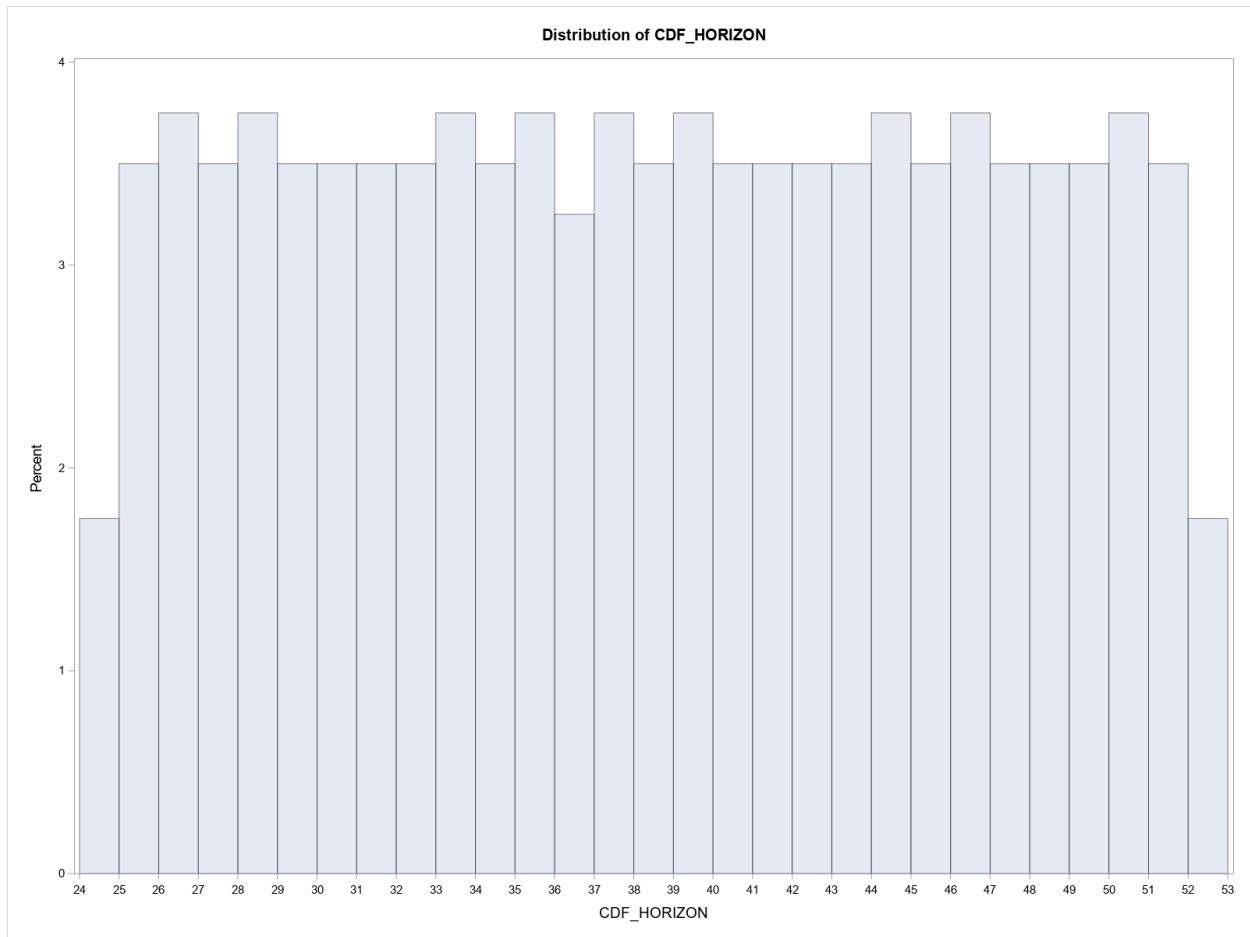


Figure 8

GENETIC ALGORITHMS

After the creation of experiment design, we can choose which road to follow, since, the first LHSMDU design, has the characteristic of covering almost any place of the multidimensional space of the hyperparameter, can be enough, can find the sub-optimal combination of the parameters, to reach our target.

Of course, the targets are typically depending by your business case, since in some case is necessary to reach the minimum mape index or reducing some variance index.

The idea behind the Genetic algorithms is to "grow" the initial population, following the principles of natural selection and evolution. It is possible to use the genetic approach when you cannot use the calculus-based techniques.

The typical phases of the genetic approach are the following:

- Initialization: i.e., the creation of the first generation of the population, but in our specific case, we already have a nice population, built using the LSHMDU approach
- Regeneration: this means that a new generation arises from the initial population. In our case, since the initialization can start from a sub-sample of the LHSMDU design,

selecting only the best spots in the available space (i.e., the points with the best achievement of our relevant index), the "generation" phase should start only from those wonderful points with good performance. The process goes on using a selection stage, where it is possible to copy the parents to the next generation, or you can manage the parents passing through a crossover operator. Some of the offspring are mutated, via a specific operator, to add some variability to the next generation, besides the simply legacy of their parents (crossed or not). The newborn population then replaces the original population and is tested against the objective function (i.e., minimize the mape index)

- Repeat: the process will check some criterion to halt the process itself. Such criterions are the number of iteration of the regeneration phase, or the achievement of good enough result, such as a low mape, with low variance.

The following Figure 8 is a simple illustration of the typical phase of a genetic optimization:

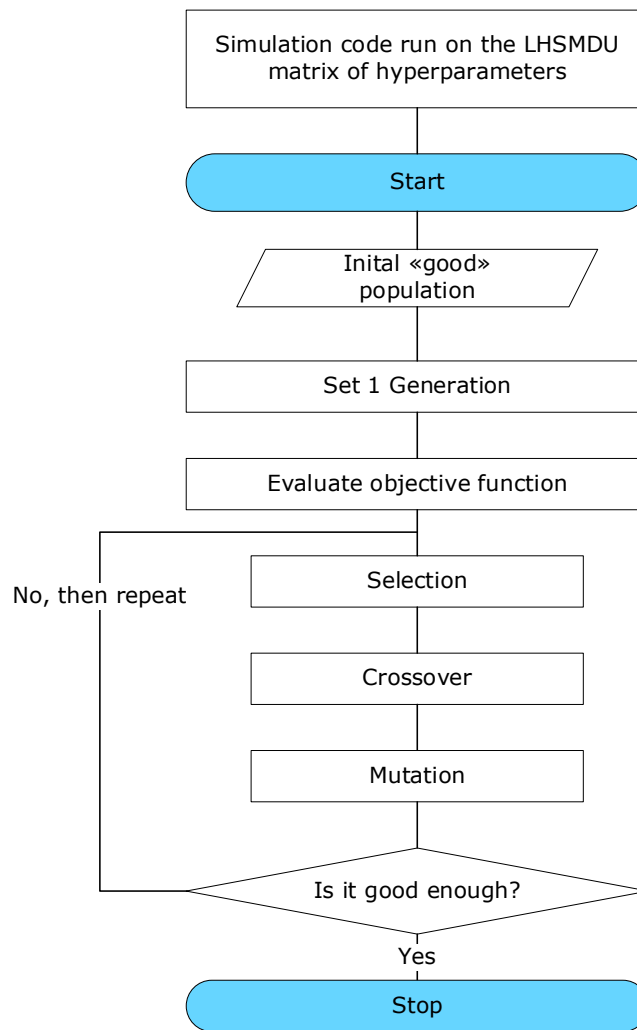


Figure 9

CONCLUSION

The introduction of the autotune options in many procedures of SAS® Viya, in particular for the SAS Visual Data Mining and Machine Learning module, has left many 9.4 users with no specific tool capable of “optimizing” the optimization process for any Machine Learning project.

This document aims to help users to understand and use the SAS/IML® to implement an effective method to reduce the optimization space for the hyperparameters of a Neural Network but is usable for any other procedure that requests extensive parametrization, like the HPFOREST, or the HPSVM.

The approach starts explaining what a DOE is, and how a first DOE can be built using the procedures available in SAS/QC®, and how to use the TRANSREG procedure to vectorize categorical variables.

The paper then tries to explain the different sampling techniques, to show the methodology and the expected result of each method.

The document presents an IML implementation of a particular enhancement over basic Latin Hypersquare Sampling, called LHS with multidimensional uniformity, and shows through techniques of space reduction, a visualization of the starting point (discrete factorial design) versus the LHSMU results.

In the final section, we explain how to use a Genetic Algorithm to furtherly enhance the best point found with the LHSMU approach.

In the end, it is safe to state that the actual frontier, is to find a way to make calculable the problem of optimizing any machine learning methodology, and how mathematical concepts, coming from different fields of application, are extremely useful.

REFERENCES

Zhou, Zhi-Hua (2012). *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall.

Jared L. Deutsch and Clayton V. Deutsch (2009). *Latin Hypercube Sampling with Multidimensional Uniformity*

McKay, Conover, Beckman (1979, 1992). *A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output From a Computer Code*

Rick Wicklin. *Statistical Programming with SAS/IML®*. Copyright © 2010, SAS Institute

Rick Wicklin. *Simulating Data with SAS®*. Copyright © 2013, SAS Institute Inc

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford: Oxford University Press

D. M. Byrne and S. Taguchi. *The Taguchi Approach to Parameter Design*. Proceedings of the 1986ASQC Quality Congress Transaction, 1986

ACKNOWLEDGMENTS

I'm thankful to my colleagues Gabriele Oliva, Gabriella Jacoel, Carlo Binda, Clio Rizzolo and Antonio Ponzio for their tough work during the last three years, in several SAS® projects, to give me the possibility to find new ways to achieve our target.

CONTACT INFORMATION

Comments and questions are valued and encouraged. Contact the author at:

Andrea Magatti
BIP Business Integration Partners
+393351807676
andrea.magatti@mail-bip.com
<https://www.businessintegrationpartners.com/>

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.