A three-stage statistical method for nonlinear regression model selection

Andrés Yáñez Escolano[1], Joaquín Pizarro Junquera[1],
Elisa Guerrero Vazquez[1], and Pedro L. Galindo Riaño[1]

[1] Grupo Sistemas Inteligentes de Computación-TIC 145
Dpto. Lenguajes y Sistemas Informáticos
Universidad de Cádiz
11510 Puerto Real, Spain
{andres.yaniez, joaquin.pizarro, elisa.guerrero, pedro.galindo}@uca.es
http://www2.uca.es/grup-invest/sic/

Abstract. The goal of inductive learning is to build models based on a set of examples. The resulting model is then used to make predictions on previosly unseen examples. Model selection is an important issue in machine learning algorithms, in particular when the sample size is small, in order to find the right trade-off between overfitting and underfitting. If the model is too complex for the sample size, there will be risk of overfitting the training data, while a too restricted model can prevent us from obtaining good predictions. In this paper we propose a three-stage statistical method based on bootstrapping techniques and multiple comparison procedures in order to determine the optimal complexity of a model.

1 Introduction

Given a number of examples of the form $<x_i,y_i>$, our goal is to find function models $y=f(x)$. The goal of these models is not to learn an exact representation of training data itself (memorization), but rather to get the ability to understand the hidden structure of the data, to predict the behaviour of novel data after training on a dataset (generalization).

There are three conditions which are necessary for good generalization: representative and sufficiently large training sets, smooth functions to learn and models with a suitable complexity.

In order to achieve the first condition, methods for selecting good training sets are discussed in statistical textbooks on sample surveys and experimental design, but it is not the goal of this paper.

The second condition, function shape, depends on the problem. In any case, it is unknown and we only have a small set of examples to learn most of the time.

The third condition is the goal of this paper. We try to estimate the suitable complexity of the model. If the model is not sufficiently complex, it will give poor predictions for new data, because it has too little flexibility (bias error). But if the model is too complex, it may fit too much to training data (variance error). The best

generalization is obtained when the trade-off between poor aproximation and overfitting is achieved. Statisticians refer to this problem as the *bias/variance dilemma* [4].

In order to achieve our goal, we propose a three-stage statitistical method based on bootstrapping techniques [3] and multiple comparison procedures (M.C.P.) [6].

This paper is structured as follows. We first outline the underlying idea of the proposed statistical method (section 2). After that, in section 3, the algorithm is described in detail. In section 4, we report on some model selection experiments, comparing our method with other resampling-based techniques. Finally (section 5), the experimental results are analyzed and future works are proposed.

2 Underlying idea

In this section we outline the methodology of our method as follows (fig. 1):

- First stage: resampled data set are obtained from the original data set by bootstrapping techniques.
- Second stage: create a set of models which are not significantly different from the model with minimum test error median.
- Third stage: select the best model from the previous set.

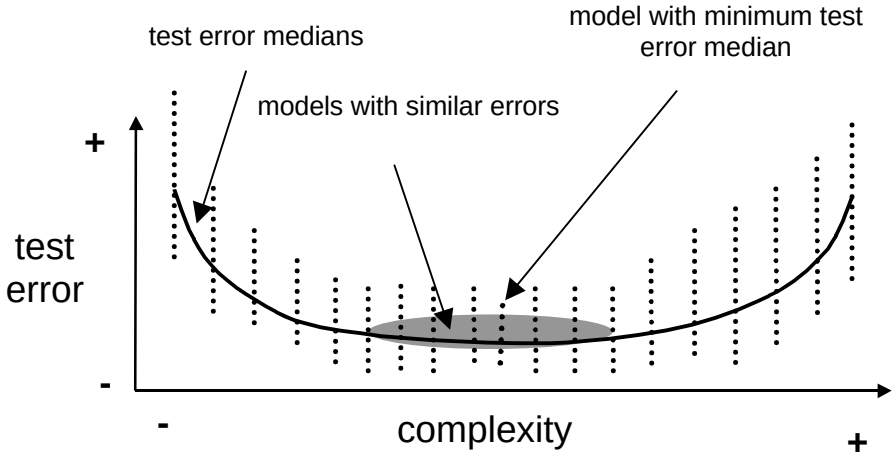

Fig. 1. The model with minimum test error may not be the model with minimum generalization error. Any model of the set of models with *similar* errors are possible candidates.

2.1 Bootstrapping data set generation

The bootstrap family was introduced by Efron and is fully described in [3]. Given a data set of size n, a bootstrap sample, with the same size than the whole training set, is created by sampling *n* instances uniformly from the data with replacement.

We apply many bootstrapped data sets (ranging from 25 to 200), which are obtained from the whole data set, on models with an increasing complexity and obtain many test error measures per model.

2.2 Models which are not significantly different

We select the model with minimum test error median. We propose to use error medians instead of error means to reduce outlier problems. In any case, as training sample size increases, error medians and error means tend to approach each other.

Then, we estimate the set of models whose errors are *similar* to the model with minimum test error median. We consider two or more models to make similar errors when they are not significantly different.

In our method two groups of statistical tests are applied: omnibus tests and multiple comparison procedures. The first group of tests evaluates several hyphothesis simultaneously and shows us whether several models are o not significantly different. The second one shows us what models are significantly different.

Not all statistical tests are useful for our experiment. We must take into account that we apply the same resampled sets on all models. Given that error measures are not independent, statistical tests for related samples should be applied.

These tests can be classified in two groups: parametric and nonparamentric [1]. The first ones are more powerful and should be used if possible, but they are based on certain assumptions about the population from which the data are drawn.

Table 1 shows the statistical tests which are used in our method ([1], [5], [9]).

Table 1. : Statistical tests for related samples.

Test	Brief description	Assumptions
Friedman test	High power. Omnibus. Nonparametric. Used to compare k > 2 samples.	None
Nemenyi test	Medium power. All pairwise test. Used to compare k > 2 samples.	None
Repeated mesaures ANOVA	High power. Omnibus. Parametric. Used to compare k > 2 samples.	Normality, compound symmetry / sphericity
t paired test	High power. Parametric. Used to compare k = 2 samples.	Normality
Wilcoxon matched pairs test	High power. Nonparametric. Used to compare k = 2 samples.	None

2.3 The selection criterion

After applying multiple comparison procedures, we have to select the best model from the set of models that are not significantly different from the model with mimimum test error median.

In an earlier version of this method [8], the selection criterion was based on Occam's razor [2] (if several models make similar errors, choose the simplest model), but our method exhibited a slight underfitting with small sample sets. Now we propose the following more powerful selection criterion in order to estimate the generalization error of the models:

$$\textit{Estimated Generalization Error} = \textit{Estimated Bias Error} + \textit{Estimated Variance Error} = \textit{Data Set Resubstitution Error} + \textit{median}(\textit{Bootstrapped Set Test Error} - \textit{Bootstrapped Set Resubstitution Error}) \quad \textbf{(1)}$$

The model with minimum estimated generalization error is selected.

3 Methodology description

The steps of the proposed statistical method, which was described briefly in section 2, may be outlined as follows:

1. Take the whole data set and create m ($25 \leq m \leq 200$) bootstrapped data sets.
2. For each resampled set:
   2.1. Train k models whose complexity goes from 1 to k
   2.2. Test these models and obtain k test error measures
3. Select the model with minimum test error median.
4. Apply Nemenyi test to obtain the models wich are not significantly different from the model with minimum test error median.
5. Apply omnibus tests: Repeated Measures ANOVA test, if its assumptions are met or Friedman test in different case.
6. If the global null hypothesis were true (that is, all models of this set make similar errors), go to step 8.
7. If the global null hypothesis were false, apply more powerful multiple comparison procedures (t or Wilcoxon paired tests with step-up Hochberg method for p-values adjustment [7]) and obtain a subset with the models which are not significantly different from the model with minimum test error median.
8. Apply a selection criterion to obtain the best model.

All tests have been applied using $\alpha = 5\%$.

4 Experimental results

In our experiments we study the efficiency of our method with differents functions to learn, sample sizes (15, 25, 50, 100 y 500 examples, respectivily) and models (RBF networks, MLP networks and polynomial functions).

The experimental functions are the following:

$$y = -0.2x^4 + 1.5x^3 - 6x + 3 + x, x \in (-2,+2) \quad \textbf{(2)}$$

$$y = 10 \sin(2x + 6) + x, x \in (-2,+2) \quad \textbf{(3)}$$

where ξ is gaussian noise with zero mean and low variance (2% of generalization sample standard deviation).

In order to explore the behaviour of our method, it was compared with different resampling techniques (.632 bootstrapping, 10-fold cross-validation, leaving one out, random subsampling [3]) applied on several model selection tasks.

4.1 An efficiency measure

We need an efficiency measure that indicates the goodness-of-fit of a model. Given a model j, we propose the next efficiency measure:

$$efficiency_j = \frac{Generalization\ Error(mod\ el\ with\ minimum\ generalization\ error\ )}{GeneralizationError(mod\ el\ \ j\ )} \qquad \textbf{(4)}$$

Small values (near to zero) indicate a poor fit, while values near one correspond to models with a good generalization power.

In order to compute the generalization error, we use a large sample set with ten thousand samples.

4.2 Simulation results

RBF, MLP neural networks and polynomial fuctions are used as models.

We create one thousand data sets per experimental function to apply RBF networks and polynomials functions, and one hundred data sets to MLP networks. Each data set with 15, 25, 50, 100 and 500 examples, respectively.

From each data set we create fifty resampled sets to apply our statistical method. Fifty learning plus testing iterations are performed to obtain random subsampling and bootstrap estimates.

From tables 2 through 7 we can see the means, medians, and standard deviations of efficiency measures obtanied after applying traditional ressampling techniques and both versions of our statistical method (the first uses Occam's razor criterion, while the second one applies eq. 1) on the previous data sets.

4.2.1 Simulation results with RBF networks

RBF are designed as having one hidden layer for which the combination function is the Euclidean distance between the input vector and the weight vector. Gaussian Radial Basis Functions have been used. The placement of the kernel functions has been accomplished using the k-means algorithm. The width of the basis functions has been set to

$$\left\| max(x_i - x_j) \right\| / \sqrt{2n} \qquad \textbf{(5)}$$

where n is the number of kernels, which goes from 1 to k = 14. The second layer is a linear mapping from the RBF activations to the output nodes. Tables 2 and 3 show efficiency results.

Table 2. : Efficiency measures of RBF networks trained with samples generated from (2).

		.632 Bootstrap	10-fold cross-val.	leave one out	10% hold out	statistical method (v.1)	statistical method (v.2)
15	*mean*	0.6268	0.7160	0.7123	0.7005	0.5938	0.7380
	median	0.6620	0.7967	0.7995	0.7840	0.6011	0.7980
	std	0.2701	0.2776	0.2847	0.2885	0.2862	0.2434
25	*mean*	0.7696	0.8083	0.7902	0.8016	0.7171	0.8246
	median	0.8081	0.8847	0.8761	0.8704	0.7894	0.8780
	std	0.1986	0.2233	0.2416	0.2236	0.2424	0.1927
50	*mean*	0.9322	0.9079	0.9030	0.9148	0.8931	0.9223
	median	0.9822	0.9696	0.9697	0.9729	0.9487	0.9712
	std	0.1026	0.1475	0.1620	0.1380	0.1236	0.1098
100	*mean*	0.9748	0.9633	0.9534	0.9655	0.9701	0.9734
	median	0.9891	0.9856	0.9849	0.9873	0.9894	0.9895
	std	0.0452	0.0708	0.0962	0.0709	0.0583	0.0520
500	*mean*	0.9946	0.9948	0.9941	0.9940	0.9919	0.9939
	median	0.9983	0.9982	0.9979	0.9969	0.9920	0.9960
	std	0.0069	0.0066	0.0079	0.0073	0.0070	0.0066

Table 3. : Efficiency measures of RBF networks trained with samples generated from (3).

		.632 Bootstrap	10-fold cross-val.	leave one out	10% hold out	statistical method (v.1)	statistical method (v.2)
15	*mean*	0.7567	0.8349	0.8219	0.8263	0.8836	0.9089
	median	1.0000	0.9942	0.9891	0.9903	1.0000	1.0000
	std	0.3838	0.2780	0.2883	0.2850	0.2673	0.2061
25	*mean*	0.9690	0.8813	0.8649	0.8832	0.9677	0.9185
	median	1.0000	0.9886	0.9870	0.9922	1.0000	1.0000
	std	0.0913	0.2083	0.2283	0.2103	0.1042	0.1750
50	*mean*	0.9708	0.9387	0.9278	0.9435	0.9830	0.9675
	median	1.0000	0.9929	0.9924	0.9941	1.0000	1.0000
	std	0.0781	0.1267	0.1467	0.1205	0.0646	0.0932
100	*mean*	0.9786	0.9704	0.9637	0.9725	0.9944	0.9889
	median	0.9984	0.9963	0.9969	0.9973	1.0000	1.0000
	std	0.0433	0.0621	0.0813	0.0578	0.0166	0.0266
500	*mean*	0.9940	0.9949	0.9945	0.9950	0.9984	0.9972
	median	0.9982	0.9986	0.9987	0.9986	1.0000	1.0000
	std	0.0086	0.0077	0.0085	0.0076	0.0037	0.0050

Our method (v.2) shows a good mean efficiency, which is higher than the others methods for small sample sets (15, 25 examples) and similar when the sample set size grows up. The first version of this method produces the best results when the function to learn is (3), but the worst ones when the sample sets come from (2).

4.2.2 Simulation results with MLP neural networks

Multilayer perceptrons networks are designed as having hyperbolic tangent sigmoid transfer function in the hidden layer and linear transfer function in the output layer. They are trained using Levenberg-Marquardt algorithm and no regularization technique has been applied. The number of hidden units ranges from 1 to 12.

Tables 4 and 5 show the simulation results. We can notice that our method (v. 2) offers better performance with medium sample size (25 and 50 examples).

Table 4. : Efficiency measures of MLP networks trained with samples generated from (2).

		.632 Bootstrap	**10-fold cross-val.**	**leave one out**	**10% hold out**	**statistical method (v.1)**	**statistical method (v.2)**
15	*mean*	0.5725	0.5565	0.5706	0.5427	0.5352	0.5444
	median	0.5232	0.5499	0.5434	0.4952	0.4471	0.4993
	std	0.2754	0.3348	0.3198	0.3212	0.2675	0.3135
25	*mean*	0.5641	0.5534	0.5796	0.6237	0.5057	0.7266
	median	0.4387	0.5367	0.5738	0.6858	0.3854	0.8744
	std	0.3050	0.3476	0.3286	0.3217	0.3043	0.3172
50	*mean*	0.7789	0.7376	0.7553	0.7520	0.7148	0.8849
	median	0.8909	0.9049	0.8637	0.9356	0.8706	0.9941
	std	0.2798	0.3344	0.2848	0.3368	0.3436	0.2113
100	*mean*	0.9018	0.8466	0.8457	0.8461	0.8336	0.9080
	median	0.9625	0.9383	0.9322	0.9357	0.9391	0.9778
	std	0.1791	0.2484	0.2392	0.2560	0.2662	0.1853
500	*mean*	0.9812	0.9420	0.9270	0.9308	0.9308	0.9504
	median	0.9879	0.9856	0.9934	1.0000	1.0000	1.0000
	std	0.0300	0.1503	0.1878	0.1704	0.1704	0.1536

4.2.3 Simulation results with polynomial functions

We consider the problem of finding the degree of a polynomial that better fits a set of data. Polynomials with degrees ranging from 1 to 15 are used (tables 6 and 7).

The performance of our method (v. 2) is the best with small sample set, and similar to the other methods when we apply large sample sets. The statistical method (v.1) only works well when the models belong to the same family that the function to learn (see table 6).

One of the most striking aspect is the low efficiency mean of .632 bootstrapping with small sample sets when the function to learn is (3).

Table 5. : Efficiency measures of MLP networks trained with samples generated from (3).

		.632 Bootstrap	10-fold cross-val.	leave one out	10% hold out	statistical method (v.1)	statistical method (v.2)
15	*mean*	0.4238	0.4930	0.5192	0.4886	0.4359	0.4666
	median	0.3856	0.4140	0.4724	0.4204	0.3854	0.4004
	std	0.2781	0.3082	0.3105	0.3313	0.3168	0.3001
25	*mean*	0.6841	0.6753	0.6571	0.7516	0.6784	0.8160
	median	0.7776	0.8175	0.7567	0.9303	0.9381	0.9944
	std	0.3265	0.3438	0.3432	0.3216	0.3695	0.2698
50	*mean*	0.8935	0.8648	0.8304	0.8372	0.8573	0.9064
	median	0.9997	0.9661	0.9376	0.9355	1.0000	0.9997
	std	0.2176	0.2240	0.2495	0.2500	0.3043	0.2058
100	*mean*	0.9415	0.8276	0.8462	0.8215	0.8588	0.9348
	median	0.9682	0.9661	0.9774	0.9570	1.0000	0.9996
	std	0.0671	0.3033	0.2940	0.3111	0.3178	0.1794
500	*mean*	0.9883	0.9028	0.8998	0.8956	0.8983	0.9330
	median	0.9935	0.9996	0.9973	0.9999	1.0000	1.0000
	std	0.0145	0.2720	0.3357	0.2822	0.2868	0.2318

Table 6. : Efficiency measures of polynomial functions trained with samples generated from (2).

		.632 Bootstrap	10-fold cross-val.	leave one out	10% hold out	statistical method (v.1)	statistical method (v.2)
15	*mean*	0.7771	0.8108	0.7848	0.7893	0.8089	0.8396
	median	0.9031	0.9206	0.9082	0.9109	0.9126	0.9245
	std	0.2963	0.2591	0.2817	0.2786	0.2664	0.2242
25	*mean*	0.8986	0.8544	0.8419	0.8520	0.8940	0.8865
	median	0.9312	0.9331	0.9316	0.9340	0.9250	0.9390
	std	0.1354	0.2113	0.2325	0.2201	0.1381	0.1607
50	*mean*	0.9588	0.9245	0.9098	0.9279	0.9460	0.9520
	median	0.9997	0.9939	0.9927	0.9911	0.9939	0.9988
	std	0.0671	0.1466	0.1750	0.1320	0.0690	0.0818
100	*mean*	0.9807	0.9653	0.9551	0.9675	0.9820	0.9821
	median	1.0000	0.9999	1.0000	1.0000	1.0000	1.0000
	std	0.0461	0.0836	0.1100	0.0746	0.0393	0.0376
500	*mean*	0.9950	0.9955	0.9930	0.9954	0.9992	0.9980
	median	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	std	0.0092	0.0086	0.0249	0.0088	0.0035	0.0056

Table 7. : Efficiency measures of polynomial functions trained with samples generated from (3).

		.632 Bootstrap	10-fold cross-val.	leave one out	10% hold out	statistical method (v.1)	statistical method (v.2)
15	*mean*	0.3443	0.6152	0.6111	0.6074	0.3463	0.6253
	median	0.3202	0.6619	0.6664	0.6582	0.3038	0.6591
	std	0.2614	0.3556	0.3621	0.3633	0.2842	0.3525
25	*mean*	0.6191	0.7784	0.7722	0.7843	0.7235	0.8420
	median	0.4971	0.9379	0.9311	0.9421	0.9300	0.9747
	std	0.3271	0.2993	0.3051	0.2922	0.3261	0.2559
50	*mean*	0.9459	0.8953	0.8785	0.9115	0.9481	0.9326
	median	0.9802	0.9682	0.9670	0.9727	0.9819	0.9756
	std	0.1170	0.1911	0.2176	0.1660	0.1214	0.1350
100	*mean*	0.9670	0.9536	0.9377	0.9583	0.9723	0.9700
	median	0.9805	0.9771	0.9760	0.9783	0.9777	0.9798
	std	0.0590	0.0870	0.1316	0.0755	0.0328	0.0437
500	*mean*	0.9953	0.9951	0.9945	0.9952	0.9930	0.9946
	median	0.9999	0.9999	1.0000	0.9999	1.0000	0.9999
	std	0.0076	0.0077	0.0096	0.0076	0.0108	0.0090

5 CONCLUSIONS AND FUTURE WORK

In our experiments, we have studied the efficiency of resampling methods in comparison with a three-stage statistical method.

Cross-validation, leave-one-out and random subsampling techniques give similar efficiency measures, but they are less prowerful than the proposed method (v. 2). The best selection among these resampling techniques would be 10-fold cross-validation, which minimizes the computational cost.

Our experiments show that .632 bootstrapping technique produces the worst performance with small sample sets (see fig. 2) when RBF networks and polynomial functions are used, but it improves and is better than other resampling techniques as the sample set size increases.

Finally, our experiments show that our method (v. 2) appears as preferable, because its efficiency is similar or higher than the rest of methods most of the time, especially with small sample sets. When the model provides a good description of the observed data, our method applied with Occam's razor as selection criterion produces a better mean efficiency (see tables 3 and 6).

Future work will be addressed to the application of our method to other models and the consideration of more powerful M.C.P.s and selection criteria.

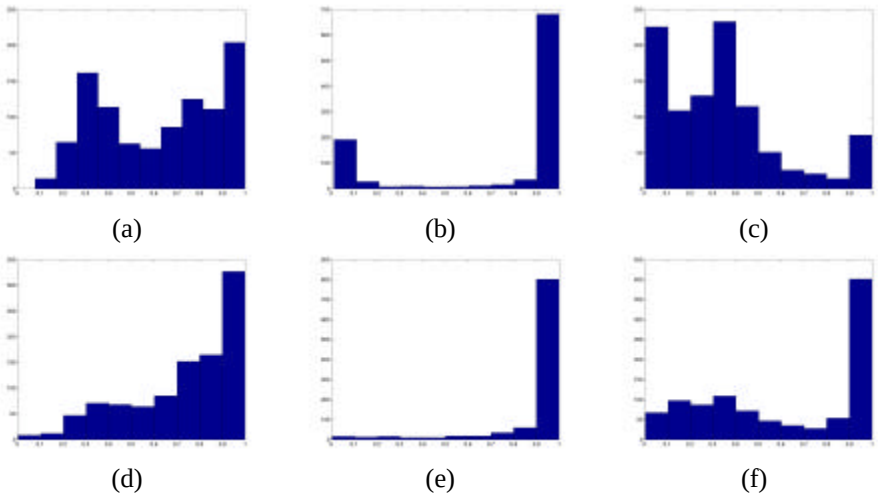

Fig. 2. Comparison of .632 bootstrapping versus our statistical method (v.2) with small sample set size (15 examples). Fig. shows the distribution of efficiency in the following simulations: function (2)-RBF networks (fig. a and d), function (3)-RBF networks (fig. b and e) and function (3)-polynomials functions (fig. c and d). Fig. a, b and c correspond to .632 bootstrapping technique.

Acknowledgements

This work has been supported by the Junta de Andalucía (PAI research group TIC-145).

REFERENCES

1. Conover, W. J.: Practical Nonparametric Statistics, John Wiley & Son (1999).
2. Domingos, P.: The role of Occam's razor in knowledge discovery. Data Mining and Knowledge Discovery, 3(4), 409-425 (1999).
3. Efron, B., Tibshirani, R.: Introduction to the bootstrap, Chapman & Hall (1993).
4. Geman, S., Bienenstock, E., Doursat, R.: Neural networks and the bias/variance dilemma. Neural Computation, 4(1), 1-58 (1992).
5. Girden, E. R.: Anova repeated measures, Sage Publications (1993).
6. Hsu, Jason C.: Multiple comparisons. Theory and methods. Chapman & Hall (1999)
7. Lasarev, M. R.: Methods for p-value adjustment, Oregon Health & Science University, http://medir.ohsu.edu/~geneview/education/dec19_h.pdf (2001).
8. Yañez, A., Guerrero, E., Galindo, P., Pizarro, J.: A resampling and multiple testing-based procedure for determining the size of a neural network. Proceedings of 10th European Symposium on Artificial Neural Networks, pp. 65-70, Bruges, Belgium (2002).
9. Zar, J. H.: Biostatistical Analysis, Prentice Hall (1996).