Overfitting problem and the over-training in the era of data

Particularly for Artificial Neural Networks

Imanol Bilbao
Electric Engineering Department
Engineering School
University of the Basque Country (UPV/EHU)
Bilbao, Spain
imanol.bilbao@ehu.eus

Javier Bilbao
Applied Mathematics Department
Engineering School
University of the Basque Country (UPV/EHU)
Bilbao, Spain
javier.bilbao@ehu.es

Abstract—**When we try to classify a set of data or to create a model to a cloud of points, different techniques can be used. Among them, Artificial Neural Networks are nowadays re-invented with the peak of the Machine Learning, Big Data, etc. In the process to find the best classification and be sure on it, one of the biggest concerns that we can come up against is the problem of overfitting. In this paper, we analyze it and set out a case study.**

Keywords—classification; overfitting; artificial neural networks; machine learning.

I. Introduction

Classification is one of the most important tasks in Machine Learning [1]. In a traditional single-label classification problem, the goal and sometimes the wish is to use or create a model that can find if there are relationships among several predictive attributes, expressed as variables, and what we can take as features that describe the process. In addition, the model has to find some links with the called class labels. These class labels in databases would be (really they are) separated from tuples and attributes, where tuple means the whole set of values of the attributes in a given row and attribute means a table field. Class label is the discrete attribute whose value you want to predict based on the values of other attributes. Each class label is represented by a discrete value. In a single-label classification problem, each example of the dataset is associated to a single label.

Several classification problems are structured in order to link one example with one class level (and only one), and we can consider that these class labels do not have any relation. Thus, procedures used for classifying do not take any relation among different class labels. Therefore, this type of classification is usually called as single-label classification or nonhierarchical classification, where image classification may be a typical example [2, 3].

But a more general type of classification, namely multi-label classification [4] can be possible and there are several studies about it [5, 6, 7, 8]. For this multi-label classification, each example in the dataset is associated to one or more class labels. Recently, this type of classification problem has attracted significant attention given the increasing number of applications from different sources [4, 9, 10, 11], such as bioinformatics, music categorization into emotions, semantic annotation of media, text mining and image recognition. Moreover, in a high number of applications the set of possible labels is organized as a hierarchical system, where some classes will have a higher level (superclass) or a lower level (subclass). In this way and with the same purpose we can act in similar but different directions, like Kocev et al. that structure some functions as predictive clustering trees [12], or Cerri et al. that structure them as a tree-based hierarchy [13].

In this way, neural networks and other machine learning models can be submitted what is called overfitting. An artificial neural network (ANN) is an information processing structure, often considered a universal function approximator, particularly neural networks with unbounded activation functions [14]. ANNs are configured (or learn) to solve a certain problem, for example, classification. Classification refers to the mapping of certain patterns of features into a certain given category. Then, ANN learns from (or is trained with) a large set of feature vectors and the corresponding class or label are available a priori [1].

II. Artificial Neural Networks

An ANN is an independent computational element set (neurons), totally interconnected to each other, that work autonomously but synchronously with the other elements.

Each neuron receives impulses from other neurons and gives them a certain importance or specific weight. After that, the neuron transmits the sign to other neurons, or even to itself (feedback). Restrictions on the number and the connections among neurons limit the type and application field of the ANN.

In order to a correct work of a neural network, that is, to give the correct outputs from a certain input set, the ANN must learn by means of training the guidelines of the necessary calculations to be performed. Normally, these process is based on examples or learning pattern.

978-1-5386-0821-0/17/$31.00 ©2017 IEEE

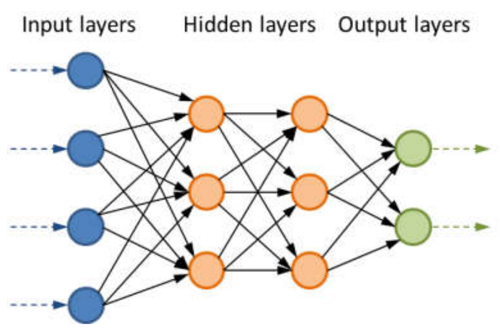

Fig. 1. Schematic of a neural network before training. Circles represent neurons, while arrows represent connections between neurons.

The training of a neural network is the modification of the weights in order to achieve that each neuron gives the correct answer or output, in all situations that it has to learn.

It is proved that a multilayer feedforward artificial neural network with one or more than one hidden layer is enough to approximate any non-linear continuous function in a closed interval, provided enough neuron exists in each layer [15]. For example, load flow resolution is a non-linear problem, and therefore, theoretically it may be solved by means of artificial neural networks (ANN). In the fields of load flow and of optimization of load flow some satisfactory tests have been implemented [16, 17]. In these tests the application of ANN improves the results obtained by conventional algorithms of load flow.

The own makeup of the neural networks is not very clear to determinate the connection between inputs and outputs, and the incorporation to the model of empiric knowledge is an arduous task. So, although it is true that the computational speed may be increased and changes in data may be adapted, it is difficult to find a training method that guarantees a total convergence of the ANN in the all cases that the network can study. Moreover, the learning time and the number of necessary patterns that we have to give to the ANN in its training are factors that are not optimized enough [18].

The back-propagation learning rule is used in perhaps 80% to 90% of practical applications. Improvement techniques can be used to make back-propagation more reliable and faster. Various improvement techniques were applied to the different network architectures tested, and it was concluded that the most suitable training method for the architecture selected was the back-propagation method based on the Levenber-Marquardt optimization technique. This technique is more sophisticated than the gradient descent used in the back-propagation technique [19].

III. OVERFITTING

It says that there is overfitting when the error of the proposed curve (model) referred to the data is zero or almost zero, that is, the proposed curve fits all or almost all data with zero error. It seems that it would be perfect, but we have to take into account that the shape of the proposed curve can be of different types and, what is more frequent, it is possible that our data have some noise. So, if our curve or model is too good that fits all our data without any error we must contemplate the possibility that the proposed curve is capturing the noise of the data. In neural networks, sometimes overfitting happens when the model shows low bias but high variance. Normally, overfitting is a result of an excessively accurate or complicated model.

It is obvious that if a learning model excessively pursues the maximization of training accuracy, it can learn a very complex model but fall into overfitting [20]. In this case, rather than "learning" to generalize from a trend, an overfitting model may "memorize" non-predictive features of the training data. For example, Srivastava et al., point out that Deep Neural Networks (DNN) with a large number of parameters are powerful machine learning models but seriously suffer from overfitting [21]. It can be avoided by fitting several models and using techniques as validation or cross-validation in order to compare their predictive accuracies on test data.

The overfitting is noticed when a very small training error and, at the same time, a very high validation error occur. And it is usually due to one of the following causes:

- The first one is related to the optimal size of the network;
- The second one, with the existence of outliers in the input set (this causes the variance of the network parameters to be high);
- A third cause is when too complex resolution algorithms are used;
- And the last one is when the number of data used in training is too high.

IV. CASE STUDY

The regression models allow to evaluate the relationship between a variable (dependent) with respect to other variables as a whole (independents). We can have hypotheses that fit perfectly or very well to the training data but do not reflect well the trend of the model, or perhaps fail to generalize to new examples. This often happens when we have a high number of input parameters which results in very complicated functions with many unnecessary curves and angles.

We take a time series corresponding to the wind power generation in the Spanish electric system for May 21, 2017, data that are available from REE, Red Eléctrica de España, which is the sole transmission agent and operator of the Spanish electricity system [22]. The current operation of wind power units depends mainly on the decentralized decisions of generating firms, whose goal is to maximize their own benefits. All firms compete to provide the service and the price is established through auctions in the energy market. Stock market prediction is a particularly difficult problem because of the number and complexity of the factors influencing its determination. Obtaining accurate forecasts of generation for the electricity market is a fundamental need for producers.

Thus, we scale the data in the plane from 0 to 7 for a better view of the representation. We will use different polynomials to fit these points that are our training data.

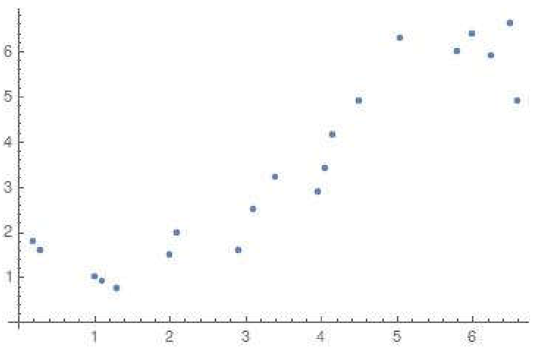

Fig. 2. Training data.

The adjustment by a polynomial of linear order (Fig. 3) represents a case of adjustment of low quality (sub-optimal). Although it is not a disastrous model, data are not well represented by a line. In order to not use a particular and tendentious assessment, a cost function (error function) is used. The equation of the model or proposed curve for a quadratic (second order) polynomial is done by (1), while the cost function is shown in (2), where *n* is the number of the training data or points.

$$h_\theta(x) = \theta_0 + \theta_1 \cdot x + \theta_2 \cdot x^2 \tag{1}$$

$$cost = \frac{1}{2n}\sum_{i=1}^{n}\left(h_\theta(x^i) - y^i\right)^2 \tag{2}$$

The value obtained by the cost function is 0.3079. As it can be seen, it is a case of underfitting. It says that there is underfitting when the error of the proposed curve referred to the data is enough high in several of the values of our data or the average of the error of the whole curve is high. In this case, we say that the proposed curve cannot capture the underlying trend of our data.

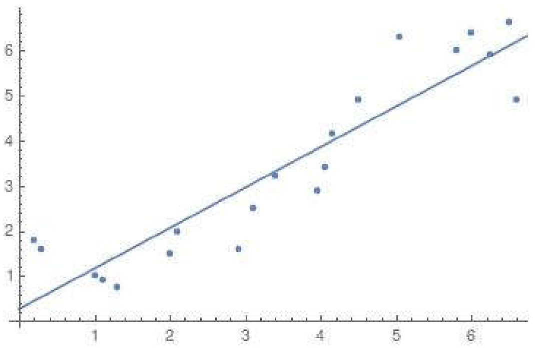

Fig. 3. Linear fitting.

The setting has poor quality. Apparently the data would fit better if the hypothesis corresponded to a higher degree polynomial, like in Fig. 4 with a quadratic polynomial.

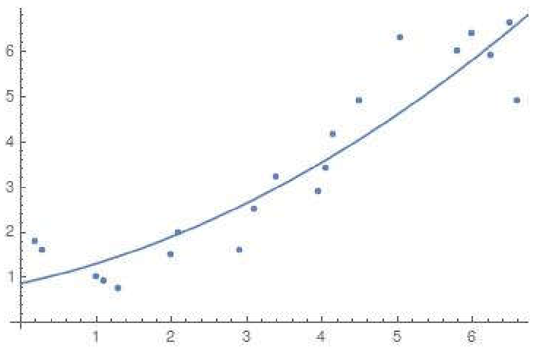

Fig. 4. Quadratic fitting.

The fit by a quadratic order polynomial (Fig. 4) presents a better fit to the data, especially in comparison with the linear fit. The curve is adjusted to the data in a way that reduces the minimum cost function to 0.2668.But some points are still far from the proposed curve, particularly in the extremes of the domain. It will be compound using a one more grade polynomial: cubic polynomial that is shown in Fig. 5.

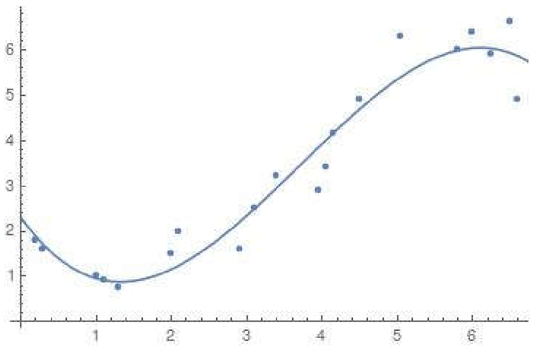

Fig. 5. Cubic fitting.

In this case, the value of the cost function is 0.1191, and visually the setting seems optimal or close to it.

In the following figures, data are adjusted successively by fourth-order polynomials (Fig. 6), sixth (Fig. 7) and tenth (Fig. 8). In all these cases there is a continuous reduction of the cost function, being the corresponding values: 0.0994, 0.0793, and 0.0497.

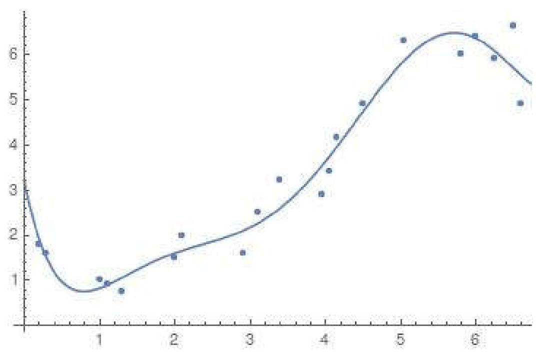

Fig. 6. Fourth-order fitting.

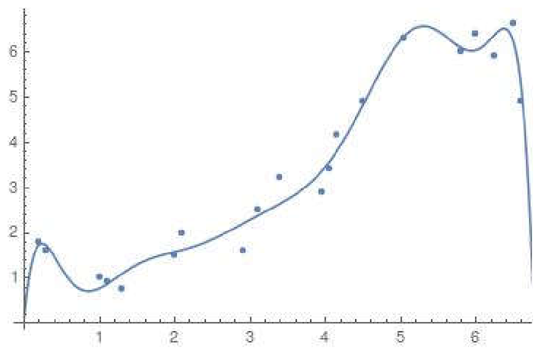

Fig. 7. Sixth-order fitting.

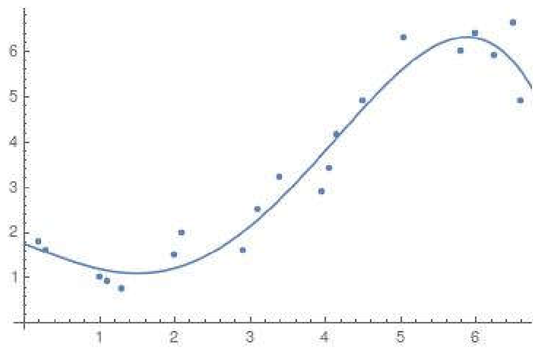

Fig. 8. Tenth-order fitting.

It is noticed that some parts of the models are too forced for a better fitting to the training data, in spite of being the tenth-order polynomial quite closed and at the same time smooth.

As the order of the polynomial increases, the minimum of the cost function decreases constantly. It is expected that for a very high order, the polynomial goes through all the points making the value of the null cost function. The question is whether a higher-order polynomial means a better representation of data, and what is critical, a greater capacity to represent data not previously included in the training set. Table I shows values of the cost function for polynomials of different orders, and in Fig. 9 and 10 models for high orders can be seen.

TABLE I. VALUES OF THE COST FUNCTION

Grade of the polynomial	*Value of the cost function*
1 (linear)	0.3079
2	0.2668
3	0.1191
4	0.0994
6	0.0793
10	0.0497
20	0.0003
40	6.02×10^{-12}

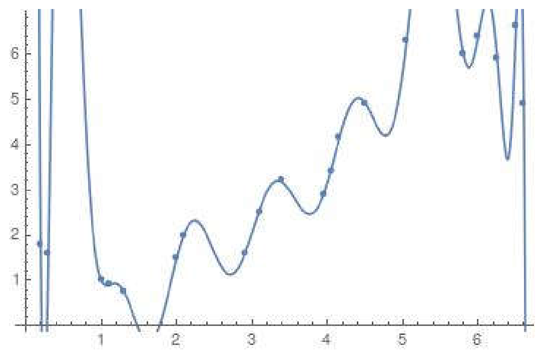

Fig. 9. Twentieth-order fitting.

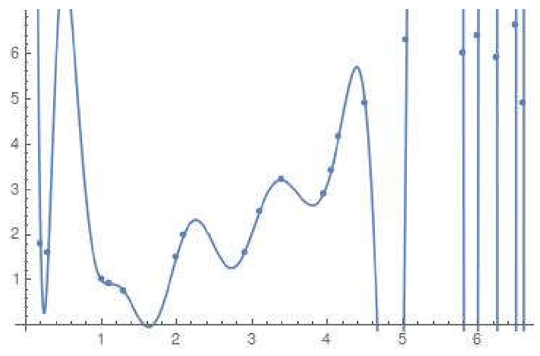

Fig. 10. Fourtieth-order fitting.

It is clear that sixth and tenth order (and higher) cases are poor representations of the data, as long as the ultimate goal is to obtain a generalist function of the behavior represented by the data.

Thus, last cases are typical cases of over-adjustment. In these cases, the values of the parameters θ_j are strongly dependent on the particular data of the training set. If another set of data were used, the parameters θ_j would be very different from those obtained. Hence, the parameters have been over-adjusted to the data.

Therefore, it remains to be determined how to obtain a good fit for the training data set (adjustment of model parameters), and in turn to be generalist of the behavior of the phenomenon represented (adjustment of the model complexity).

To solve this question, different techniques can be used. The so-called regularization technique reduces the risk of over adjustment. Other one is the validation error (cross-validation). In this case, the original data set is divided, removing a small subset of them. Once the adjustment is made, the parameters θ_j obtained are used to determine the estimated value of the output of this data.

Therefore, we can say that the adjustment objective should be the model that best adjusts (lowest error) with the least possible complexity.

V. Conclusions

When we try to fit a curve to a pattern, that is, to find a rule to create a model of a cloud of points, overfitting is one of the biggest concerns. This one of the main problems for classification using Artificial Neural Networks.

Comparisons among different orders of polynomials show that it is difficult to take the best adjustment, but cost function can be very helpful to avoid overfitting, and underfitting as well. A higher degree polynomial might have a very high accuracy (or very low error) on the training data, but it is expected to fail badly on test dataset or when the chosen training data will be different.

References

[1] S. Boucheron, O. Bousquet, G. Lugosi, Theory of classification: A survey of some recent advances, ESAIM: Probability and Statistics 9, pp. 323–375, 2005.

[2] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. Computer Vision and Image Understanding , 106(1):59–70, 2007.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition , pages 248–255, 2009.

[4] G. Tsoumakas, I. Katakis, I. Vlahavas, "Mining Multi-label Data", Data Mining and Knowledge Discovery Handbook, Oded Maimon and Lior Rokach (Eds.), Springer US, Boston, MA, pp. 667–685, 2010.

[5] A.G.C. de Sá, G.L. Pappa, A.A. Freitas, "Towards a Method for Automatically Selecting and Configuring Multi-Label Classification Algorithms", in Proceedings of the 19th Annual Conference Companion on Genetic and Evolutionary Computation, Berlin, Germany, (GECCO'17 Companion), 2017. DOI: http://dx.doi.org/10.1145/3067695.3082053

[6] Sawsan Kanj, Fahed Abdallah, Thierry Denoeux, Kifah Tout. "Editing training data for multi-label classification with the k-nearest neighbor rule". Pattern Analysis and Applications, Springer Verlag, 19 (1), pp.145-161, 2016.

[7] Younes, Z., Abdallah, F., Denoeux, T., Snoussi, H., "A Dependent Multilabel Classification Method Derived from the -Nearest Neighbor Rule", EURASIP Journal on Advances in Signal Processing 2011 (1), 645,964, 2011. DOI 10.1155/2011/645964

[8] Sucar LE, Bielza C, Morales EF, Hernandez-Leal P, Zaragoza JH, Larrañaga P, "Multi-label classification with Bayesian network-based chain classifiers", Pattern Recognit Lett 41:14–22, 2014.

[9] M.L. Zhang, Z. H. Zhou., "A Review on Multi-Label Learning Algorithms", IEEE Transactions on Knowledge and Data Engineering, 26, 8, pp. 1819–1837, 2014.

[10] M. Fontani, T. Bianchi, A. De Rosa, A. Piva, M. Barni., "A framework for decision fusion in image forensics based on Dempster–Shafer theory of evidence", IEEE Trans Inform Forens Sec 8(4):593–607, 2013.

[11] A. Osojnik, P. Panov, S. Džeroski, "Multi-label classification via multi-target regression on data streams", Mach Learn, 106, pp. 745–770, 2017.

[12] D. Kocev, C. Vens, J. Struyf, S. Džeroski, Tree ensembles for predicting structured outputs, Pattern Recognit. 46(3), pp. 817–833, 2013. http://dx.doi.org/10.1016/j.patcog.2012.09.023.

[13] R. Cerri, G.L. Pappa, A.C.P. Carvalho, A.A. Freitas, An extensive evaluation of decision tree-based hierarchical multilabel classification methods and performance measures, Comput. Intell. 31(1), pp. 1–46, 2015. http://dx.doi.org/10.1111/coin.12011.

[14] A. Clare, R.D. King, "Knowledge Discovery in Multi-label Phenotype Data", in Proc. of the European Conference on Principles of Data Mining and Knowledge Discovery (PKDD), Springer-Verlag, London, UK, pp. 42–53, 2001.

[15] J.A.K. Suykens, J.P.L. Vandewalle, B.L.R. De Moor, Arificial Neural Networks for Modelling and Control of Non-Linear Systems, Kluwer Academic Publishers, 1996.

[16] M. Suresh, T.S. Sirish, T.V. Subhashini, T. Daniel Prasanth, "Load Flow Analysis of Distribution System Using Artificial Neural Networks". In: Satapathy S., Bhateja V., Udgata S., Pattnaik P. (eds) Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications. Advances in Intelligent Systems and Computing, vol 515. Springer, Singapore, 2017.

[17] H.R. Baghaee, M. Mirsalim, G.B. Gharehpetian, H.A.Talebi, "Application of RBF neural networks and unscented transformation in probabilistic power-flow of microgrids including correlated wind/PV units and plug-in hybrid electric vehicles", Simulation Modelling Practice and Theory, vol. 72, pp. 51-68, 2017.

[18] J. Bilbao, E. Bravo, M. Rodríguez, O. García, C. Varela, P. González, N. M. Tabatabaei, "Neural Networks for Load Flow", Scientific Bulletin of the University of Pitesti, Series Electronics and Computer Science, number 8, vol. 2, pp. 1-6, 2008.

[19] A.J. Mazon, I. Zamora, J. Gracia, J. Bilbao, J.R. Saenz, "Falneur: ANN based software to fault location in electrical transmission lines", IASTED International Conference on Applied Informatics, 2001.

[20] X. Feng, Y. Liang, X. Shi, D. Xu, X. Wang, R. Guan, "Overfitting Reduction of Text Classification Based on AdaBELM", Entropy, 19, 330, 2017. doi:10.3390/e19070330

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, "Dropout: A SimpleWay to Prevent Neural Networks from Overfitting", J. Mach. Learn. Res., vol. 15, pp. 1929–1958, 2014.

[22] REE (Red Eléctrica de España), http://www.ree.es/en/activities/realtime-demand-and-generation, 2017.