Adopting some good practices to avoid overfitting in the use of Machine Learning

IMANOL BILBAO[1], JAVIER BILBAO[2], CRISTINA FENISER[3]
[1]Electrical Engineering Department
[2]Applied Mathematics Department
[3]Department of Management and Economic Engineering
[1, 2]University of the Basque Country, UPV/EHU
Engineering School, Alda, Urkijo, s/n, 48013 - Bilbao
SPAIN
[3]Faculty of Machine Building, Technical University of Cluj-Napoca
ROMANIA
imanol.bilbao@ehu.eus javier.bilbao@ehu.es cristina.feniser@mis.utcluj.ro

Abstract: - In Machine Learning, different techniques, methods and algorithms are applied in order to a better approach for the problem that is solving. Adaptive learning, self-organization of information, generalization, fault tolerance and real-time operation are some of the most used in this field. These systems are dynamic and they can learn from the data adapting to the nature of the information. But an excessive adaptation or improvement of the response to the training data can lead to a poor generalization in many cases. Excessive training with the same set of data will cause the classification curves to over-detail the formal variations of that set. To avoid this overfitting, certain preventions can be taken. One possible option is to use the regularization technique keeping all the variables. This technique works well when we have many input parameters and each contributes "a little" in the prediction. We can conclude that the number of input features compared with the number of training samples, is really important to avoid overfitting.

Key-Words: - Machine Learning, overfitting, underfitting, regularization.

1 Introduction

Definition of Machine Learning has not been reached an agreement but the different techniques, methods and algorithms that make up the so-called Machine Learning share a series of common characteristics. Among them, we can name the most relevant: adaptive learning, self-organization of information, generalization, fault tolerance and real-time operation [1].

Those kind of systems learn from the data adapting to the nature of the information. They are dynamic self-adaptive systems that learn and change according to the data presented to them.

In the adaptation process, they are able to abstract essential characteristics of the input information depending on certain degree of freedom that is imposed in the design of the machine. They are able to organize the information.

Generalization is the ability to respond appropriately to data not previously presented in the training phase, to generalize the conclusions extracted from the learnt data to the real world data.

There are two different aspects regarding fault tolerance. First, the tolerance to failures in the input information (incomplete, noisy or distorted entries). Fault tolerance in the data is also called tolerance. Second, the tolerance to failures of the whole machine, where if it destroys or alters part of it, it can continue working (although some degradation). This last type of tolerance is given thanks to internal distributed representation (information not localized), and the existence of a certain degree of redundancy that characterize some techniques [2].

Finally, one of the keys to their success is the possibility to implement them easily with existing technology getting real-time operational machines. Even in recent years, platforms that specifically implement the different techniques and architectures have appeared, out of the von Neumann architecture.

2 The generalization-adaptation dilemma

An excessive adaptation or improvement of the response to the training data can lead to a poor generalization in many cases. Excessive training with the same set of data will cause the

classification curves to over-detail the formal variations of that set. That is the well-known problem of overfitting.

How to measure generalization? We can only have an estimate of it, never a security over its value. After being trained, the system must present the best measures (accuracy, classification error, precision, recall, F1 score, etc.) against the real data set. We only can try with the largest amount of real data available and take different measures of success. If the system presents sufficiently good ratings for all the markers and with a real and varied set (in time and variety) of data, we will accept that it generalizes correctly [3].

To measure the generalization of the system we should always use real data. Data without bias, as varied as possible. Data used must offer a representative diversity of the diversity of the set of data with which the machine must work during its normal operation. The use of real data is of vital importance, without 'cooking' or selecting the data, without introducing a tendency.

In our case, we use Artificial Neural Networks. There are many neural network types, although the most widely used is the single hidden layer back-propagation network, or single layer perceptron.

This type of network is applied normally both to regression or classification. For regression, typically K = 1 (dimension) and there is only one output unit Y_1. But this kind of networks can manage multiple quantitative responses in a continuous trend. Therefore, in general, for a classification of K dimension, there are K units as output, with the i^{th} unit representing the probability of the class *i*. So, there are *i* target values Y_i, where $i = 1,\ldots, K$, each of them being coded as a 0 to 1 variable for the i^{th} class.

In this process, the derived characteristics Z_j are usually originated from linear combinations of the inputs, and then the target Y_i is calculated as a function of linear combinations of the Z_j:

$$Z_j = \sigma\left(\alpha_{0j} + \alpha_j^T X\right) \qquad j = 1,\ldots, J \tag{1}$$

$$Y_i = \beta_{0i} + \beta_i^T Z \qquad i = 1,\ldots, K \tag{2}$$

$$f_i(X) = g_i(Y) \qquad i = 1,\ldots, K \tag{3}$$

where $Z = (Z_1, Z_2, \ldots, Z_J)$, and $Y = (Y_1, Y_2, \ldots, Y_K)$.

Other notations represents as follows:

α_{jk}: connection weight between input unit *k* and hidden unit *j*

β_{ij}: connection weight between hidden unit *j* and output unit *i*

α_{0j}, β_{0i}: bias terms

σ: hidden unit activation function

g_i: output activation functions

3 The overfitting problem

We can have hypotheses that fit perfectly or very well to the training data but that do not reflect the tendency of the model well, or fail to generalize to new examples. This usually happens when we have a high number of input features (p) compared with the number of training samples (n), which gives rise to very complicated functions with many unnecessary curves and angles.

Overfitting, therefore, is characterized by high accuracy for a classifier when evaluated on the training set but, at the same time, low accuracy when the evaluation is on a separate test set. In some references, overfitting has been recognized as a problem in $p \gg n$ settings [4]. In high-dimensional low sample size settings, case that is usual in some fields such as Medicine, different authors remark that the apparent (training set, re-substitution) accuracy of a classifier is highly optimistically biased and, therefore, if we want a good classification, we do not have to take it into account. Accuracy should be calculated based on the evaluation of the classifier on separate test sets. It should also be possible to complete re-sampling in which the model is re-developed for each re-sampling [4, 5, 6].

In the case of a typical example with several data of two different types (in the next figures, blue circles data and red circle data), we could happen to the following with a high number of parameters.
We could obtain a model like the Fig. 1, when in fact we would prefer a model more similar to one of the Fig. 2.

The learning curves are the curves that show the accumulative cost function value for the test (or validation) and training set with respect to the size of the training set (number of training examples) [7].

We can use them to know if getting more examples of training could help improve the system.

If underfitting exists, as the number of examples grows the cost functions will maintain in high and

similar values. Getting more examples will not help us.

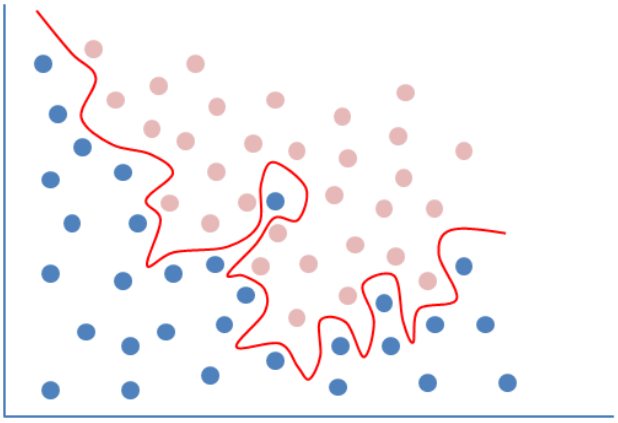

Fig. 1. Curve that overfits the boundary of the two types of data.

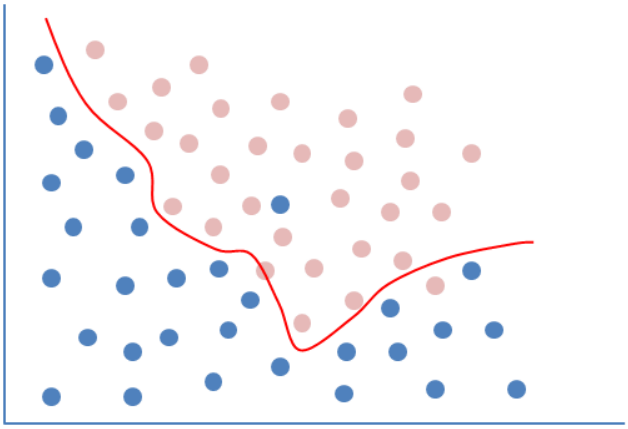

Fig. 2. Curve that fits well the boundary of the two types of data.

If there is overfitting, as the number of examples grows the test cost function will go down and training cost will stay at a low value. Between the cost functions there will be a big difference.

According to Hardt and Lin, the process of learning where high-capacity models involved and with long stretched training time on powerful devices could guide to a possible risk of overfitting [8, 9].

4 Solutions for overfitting

To avoid this, certain preventions can be taken. Among them, we can mention:

a) the re-adaptation to new training sets (online training if we can ensure a variable nature of the data in a natural way),
b) the limitation of the number of iterations carried out with the same data set,
c) the use of learning curves,
d) minimization of the values of the weights (also looking for their simplest relationships),
e) the use of regularization (parameter λ), etc.

If we have an overfitting problem, then we can:

- Get more examples for training.
- Test increasing the regularization parameter.
- Test with smaller input features sets (input variables). See if any of the variables is redundant or useless. (Maybe with the number of variables we have chosen we need more examples, as we are not going to get them, we chose to reduce the variables, so they will need fewer examples).

Avrutskiy mentions a contradiction between the necessity to increase network size and decrease in minimum number of points that prevent overfitting leads to his study [10]. According to the key principle in artificial neural networks that says precision is not the goal and a reasonable approximation is enough as a good result, the process of altering cost functions is omitted and instead the one with maximum number of available derivatives is used.

5 Regularization

Different techniques has used during the last years to address overfitting. For example, we can mention weight decay, early stopping and weight elimination in order to control overfitting [11]. One of the most used techniques has been the called early stopping and there are several references about it [12, 13, 14, 15]. Nevertheless, in some areas of research, such as time series of complex systems' behavior, this technique stops the training process too early. Other cons is that the chance to detect some relations decreases (for example, for meaningful relations between the network outputs and actual behavior of the complex system [16]).

Other authors, particularly for predicting time series, use the Elman networks, which is a positive feedback neural network with the typical three main levels of layers: input, hidden and output. Elman [17] describes it as the following:

"Both the input units and context units activate the hidden units; and then the hidden units feed forward to activate the output units. The hidden units also feed back to activate the context units. This constitutes the forward activation. Depending on the task, there may or may not be a learning phase in this time cycle. If so, the output is compared with a teacher input and backpropagation of error is used to incrementally adjust connection

strengths. Recurrent connections are fixed at 1.0 and are not subject to adjustment. At the next time step t+1 the above sequence is repeated. This time the context units contain values which are exactly the hidden unit values at time t. These context units thus provide the network with memory."

In this case, the focus is on a technique based on ensemble neural networks [18, 19, 20] to achieve the generalization. For this purpose, training data have a sequence of the previous behavior of the worked system.

In the method, a network that has the best generalization on the new training set is selected. At the same time, the number of neurons in the hidden layers of the network varies in this step, following these equations:

$$n_{11} = x \times \frac{n_{10}}{n_{20}} \tag{4}$$

$$n_{21} = x + n_{20} \tag{5}$$

$$x = \left\{ \frac{IN+1}{2} \times \mathrm{mod}\left[(IN-1), 2 \right] \right\} - \left\{ \left\{ \frac{IN+1}{2} \times \frac{1+sign\left(n_{20} - \frac{IN+1}{2} \right)}{2} + \left[\left(2 \times n_{20} - \frac{IN+1}{2} \right) - 1 \right] \times \left[\frac{1-sign\left(n_{20} - \frac{IN+1}{2} \right)}{2} + 0.5 \right] \right\} \times \mathrm{mod}(IN, 2) \right\} \tag{6}$$

where n_{10} and n_{20} are the initial number of neurons in the first and second hidden layers of the first set of networks and n_{11} and n_{21} are the new values. IN is the iteration number and x is a value used to calculate the new numbers of neurons.

Another possible option if this happens is to use the regularization technique keeping all the variables. This technique works well when we have many input parameters and each contributes "a little" in the prediction.

If we have an overfitting problem in our hypothesis function and we know which parameters should be less important, we can try to reduce the influence of those terms. Generalizing this idea, making the different influences small or close to zero, helps to have simpler hypotheses with less angulations and therefore less conducive to overfitting.

This is especially true if the contribution to the prediction of the different input parameters is similar.

It is impossible to know in advance which parameters may be more relevant in our hypothesis and in what proportion, so in regularization we will treat all parameters equally trying to get the smallest possible influence values and thus reduce the influence of its associated parameters. Since the free design terms are not associated with any parameter, they are not usually regularized.

An excess in the regularization can lead to hypotheses that are too soft, and even of constant value in the last extreme, which will end up incurring in underfitting, so a certain care is necessary when choosing the regularization parameter λ.

5.1 Feature reduction

In the case of facing an overfitting problem we can reduce the number of parameters manually, analyzing which ones are more important and we will conserve, and which ones seem secondary and we can eliminate them. We can also try to do this automatically through some technique like PCA. Unfortunately, in this way we will always be losing information.

Remember that it is not recommended to use PCA to prevent overfitting, because although in some cases it could be achieved, equal or better results can be obtained with the regularization, this being a technique that does not eliminate information from the data that could be valuable. It is advisable to try always the learning algorithm without applying PCA.

PCA is a method that seeks to minimize the projection error by reducing the data to a smaller number of dimensions. The method tries to find a vector space to project the data so that the projection error is minimal.

The error introduced in the simplification will be less if the features of the data are more linearly related, but we will always lose some information when applying it.

Remember that any new example obtained subsequently will be converted using the same reduction matrix used in PCA, which was originally calculated only with the training cases. Therefore, the validation and test data will not influence the calculation of the matrix.

6 Features-Training samples ratio compromise

As we have said before, the number of input features compared with the number of training samples, is really important to avoid overfitting. If we have a lot of features and very few samples the conclusions extracted from the training data are weak and may not be generalized to global real data. That could be known as the 'looking at clouds' effect, where we are imagining forms, for example a sheep, in a cloud and concluding that all of them look as a sheep.

Different authors offer different ratios, but to be able to conclude a statistically significant generalization, even if we cannot be completely sure about that, a ratio of 100:1 samples to features is recommended.

That ratio has always to be tested observing the learning curves to know if getting more samples would improve the final result with real data.

7 Conclusion

Not only in the case of appearing overfitting problem, but in order to use Machine Learning with all its potential, and to generalize the results of our machine for real data, we must try to adopt good practices:

- Use real data for training, avoiding data that are out of range.
- Keep a good proportion of features to training examples.
- Analyze the learning curves to see if overfitting occurs.
- Use the regularization within the chosen technique.
- Reduce the number of features as a last alternative.

References:

[1] Jagannath Aghav, Poorwa Hirve, Mayura Nene, Deep learning for Real Time Collision Detection and Avoidance, Proceedings of International Conference on Communication Computing and Networking (ICCCN-2017), Chandigarh, March 2017.

[2] A. Y. Ng, Preventing "Overfitting" of Cross-Validation Data. Proceeding ICML '97 Proceedings of the Fourteenth International Conference on Machine Learning, pp. 245-253, 1997. Available in July 2018 at: http://www.andrewng.org/portfolio/preventing-overfitting-of-cross-validation-data/

[3] James L. McClelland and David E. Rumelhart, *Parallel Distributed Processing, 2-vol. set, Explorations in the Microstructure of Cognition*, MIT Press Cambridge, MA, USA, 1986. Available in July 2018 at: https://mitpress.mit.edu/books/parallel-distributed-processing-2-vol-set.

[4] Richard Simon, Michael D. Radmacher, Kevin Dobbin, Lisa M. McShane, Pitfalls in the use of DNA microarray data for diagnostic and prognostic classification, Journal of the National Cancer Institute, vol. 95, Issue 1, pp. 14-18, 2003.

[5] Christophe Ambroise, Geoffrey J. McLachlan, Selection bias in gene extraction on the basis of microarray gene-expression data, Proceedings of the National Academy of Sciences of the United States of America, vol. 99, No. 10 (May 14, 2002), pp. 6562-6566, 2002.

[6] Jyothi Subramanian, Richard Simon, Overfitting in prediction models – Is it a problem only in high dimensions?, Contemporary Clinical Trials, vol. 36, pp. 636-641, 2013.

[7] Imanol Bilbao and Javier Bilbao, Solving problems for new results predictions in artificial neural networks, International Journal of Neural Networks and Advanced Applications, vol. 4, pp. 10-13, 2017.

[8] M. Hardt, B. Recht, and Y. Singer, Train faster, generalize better: Stability of stochastic gradient descent, Proceedings of the 33rd International Conference on Machine Learning, pp. 1225–1234, 20–22 Jun 2016.

[9] J. Lin, R. Camoriano, and L. Rosasco, Generalization properties and implicit regularization for multiple passes SGM, Proceedings of the 33rd International Conference on Machine Learning, pp. 2340–2348, 20–22 Jun 2016.

[10] V. I. Avrutskiy, Avoiding overfitting of multilayer perceptrons by training derivatives, https://arxiv.org/pdf/1802.10301, 2018.

[11] A. S. Weigend, B. A. Huberman, and D. E. Rumelhart, Predicting sunspots and exchange rates with connectionist networks. In M. Casdagli and S. Eubank, editors, Nonlinear Modeling and Forecasting, SFI Studies in the Sciences of Complexity, Proceedings vol. XII, pp. 395–432. Addison-Wesley, 1992.

[12] L. Prechelt, Early Stopping - But When?, In: Orr G.B., Müller KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol. 1524. Springer, Berlin, Heidelberg, 1998.

[13] J. Loughrey, P. Cunningham, Overfitting in Wrapper-Based Feature Subset Selection: The Harder You Try the Worse it Gets, In: Bramer M., Coenen F., Allen T. (eds) Research and Development in Intelligent Systems XXI. SGAI 2004. Springer, London, 2005.

[14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Journal of Machine Learning Research, vol. 15, pp. 1929-1958, 2014.

[15] Maren Mahsereci, Lukas Balles, Christoph Lassner, Philipp Hennig, Early Stopping without a Validation Set, arXiv preprint arXiv:1703.09580, 2017.

[16] Kaveh Mahdaviani, Helga Mazyar, Saeed Majidi and Mohammad H. Saraee, A Method to Resolve the Overfitting Problem in Recurrent Neural Networks for Prediction of Complex Systems' Behavior, 2008 International Joint Conference on Neural Networks (IJCNN 2008), pp. 3723-3728, 2008.

[17] Jeffrey L. Elman, Finding structure in time, Cognitive Science, vol. 14, pp. 179-211, 1990.

[18] S. Chiewchanwattana, C. Lursinsap, C. H. Chu, Time-series data prediction based on reconstruction of missing samples and selective ensembling of FIR neural networks, Proceedings of the 9th International Conference on Neural Information Processing, 2002.

[19] Harris Drucker, Corinna Cortes, L. D. Jackel, Yann LeCun and Vladimir Vapnik, Boosting and Other Ensemble Methods, Neural Computation, vol. 6, pp. 1289-1301. 1994.

[20] De-Wang Chen and Jun-Ping Zhang, Time series prediction based on ensemble ANFIS, 2005 International Conference on Machine Learning and Cybernetics, Guangzhou, China, pp. 3552-3556, vol. 6. 2005, doi: 10.1109/ICMLC.2005.1527557.