An Overview of Overfitting and its Solutions

An Overview of Overfitting and its Solutions

Xue Ying

Building 1, Huizhong Tower, NO.1 Shangdi Seven Street, Haidian District Beijing 10 0085 China

Email: xuee4259@gmail.com

Abstract. Overfitting is a fundamental issue in supervised machine learning which prevents us from perfectly generalizing the models to well fit observed data on training data, as well as unseen data on testing set. Because of the presence of noise, the limited size of training set, and the complexity of classifiers, overfitting happens. This paper is going to talk about overfitting from the perspectives of causes and solutions. To reduce the effects of overfitting, various strategies are proposed to address to these causes: 1) "early-stopping" strategy is introduced to prevent overfitting by stopping training before the performance stops optimize; 2) "network-reduction" strategy is used to exclude the noises in training set; 3) "data-expansion" strategy is proposed for complicated models to fine-tune the hyper-parameters sets with a great amount of data; and 4) "regularization" strategy is proposed to guarantee models performance to a great extent while dealing with real world issues by feature-selection, and by distinguishing more useful and less useful features.

1. Introduction

In supervised machine learning, there's an un-detouring issue. Model does not generalize well from observed data to unseen data, which is called overfitting [1]. Because of existence of overfitting, the model performs perfectly on training set, while fitting poorly on testing set. This is due to that over-fitted model has difficulty coping with pieces of the information in the testing set, which may be different from those in the training set. On the other hand, over-fitted models tend to memorize all the data, including unavoidable noise on the training set, instead of learning the discipline hidden behind the data.

The causes of this phenomenon might be complicated. Generally, we can categorize them into three kinds: 1) noise learning on the training set: when the training set is too small in size, or has less representative data or too many noises. This situation makes the noises have great chances to be learned, and later act as a basis of predictions. So, a well-functioning algorithm should be able to distinguish representative data from noises [2]; 2) hypothesis complexity: the trade-off in complexity, a key concept in statistic and machining learning, is a compromise between Variance and Bias. It refers to a balance between accuracy and consistency. When the algorithms have too many hypothesis (too many inputs), the model becomes more accurate on average with lower consistency [2]. This situation means that the models can be drastically different on different datasets; and 3) multiple comparisons procedures which are ubiquitous in induction algorithms, as well as in other Artificial Intelligence (AI) algorithms [3]. During these processes, we always compare multiple items based on scores from an evaluation function and select the item with the maximum score. However, this process will probably choose some items which will not improve, or even reduce classification accuracy.

In order to reduce the effect of overfitting, multiple solutions based on different strategies are

proposed to inhibit the different triggers. Nevertheless, most of them perform poorly when dealing with real-world issues, because of the great amount of hypothesis. However, none of the hypothesis sets can cover all the application fields. The contribution of this paper is to give some general guidelines to choose solutions according to the application fields, from four perspectives.

2. Early-stopping

This strategy is used to avoid the phenomenon "learning speed slow-down". This issue means that the accuracy of algorithms stops improving after some point, or even getting worse because of noise-learning. The idea has a fairly long history which can be dating back to the 1970s in the context of the Landweber iteration [4]. Also, it is widely used in iterative algorithms, especially in neural networks starting from the 1990s.

As shown in Figure 1, where the horizontal axis is epoch, and the vertical axis is error, the blue line shows the training error and the red line shows the validation error.

If the model continues learning after the point, the validation error will increase while the training error will continue decreasing.

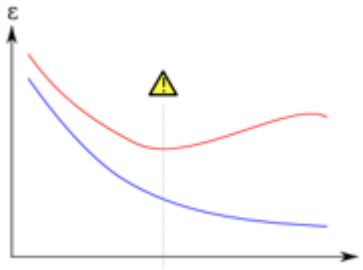

Figure 1. Validation error vs testing error

If we stop learning before the point, it's under-fitting. If we stop after the point, we get over-fitting. So the aim is to find the exact point to stop training. In this way, we got a perfect fit between under-fitting and over-fitting. For artificial neural nets, the learning process is to find a perfect set of weights and bias. The neurons learn at a rate determined by the partial derivatives of the cost-function: $\frac{\partial C}{\partial W}$ and $\frac{\partial C}{\partial b}$. So the speed of learning depends on the values of those two partial derivatives, which can be respectively described with the following formulas [5], where W_j is the j^{th} weight, b is the bias, C is the cost, X_j is the j^{th} input, y is the output, and a is the output when inputs are 1.

$$\frac{\partial C}{\partial W_j} = \frac{1}{n}\sum_x X_j(\sigma(z) - y) \tag{1}$$

$$\frac{\partial C}{\partial b} = \frac{1}{n}\sum_x (\sigma(z) - y) \tag{2}$$

Practically, to find out the point to stop learning, the obvious way is to keep track of accuracy on the test data as our network trains. In another word, we compute the accuracy at the end of each epoch and stop training when the accuracy on test data stops improving.

More generally, we can track the accuracy on validation set instead of test set in order to determine when to stop training. In another word, we use validation set to figure out a perfect set of values for the hyper-parameters, and later use the test set to do the final evaluation of accuracy. In this way, a higher generalization level can be guaranteed, compared to directly using test data to find out hyper-parameters' values. This strategy ensures that, at each step of an iterative algorithm, it will reduce bias but increase variance. So finally, the variance of the estimator will not be too high. Besides, it has a lower computational complexity. However, there also are some problems: 1) Strictly speaking, this is not a necessary sign of overfitting. It might be that accuracy of both the test data and the training data stops improving at the same time; 2) We cannot detect the stopping point immediately. Generally, we stop later to make sure that we found the exact point to stop.

There are some interesting usages of early-stopping. For instance, in the work of Rich Caruana,

Steve Lawrence, and Lee Giles [6], they combined early stopping with back-propagation. With the algorithm they proposed, we can train large nets without significant overfitting in an effective way. In the work of Hao WU, Jonathan L. Shapiro [7], they used early stopping to prevent overfitting and improve performance in Estimation of Distribution Algorithms (EADs, a class of evolutionary algorithms that use machine learning techniques to solve optimization problems).

3. Network-reduction

As we know, noise learning is one important cause of overfitting. So logically, noise reduction becomes one researching direction for overfitting inhibition. Based on this thinking, pruning is proposed to reduce the size of finial classifiers in relational learning, especially in decision tree learning. Pruning is a significant theory used to reduce classification complexity by eliminating less meaningful, or irrelevant data, and finally to prevent overfitting and to improve the classification accuracy.

Pre-pruning and post-pruning are two standard approaches used to dealing with noise:

Pre-pruning algorithms function during the learning process. Commonly, they use stopping criteria to determine when to stop adding conditions to a rule, or adding rule to a model description, such as encoding length restriction based on the evaluation of encoding cost; significance testing is based on significant differences between the distribution of positive and negative examples; cutoff stopping criterion based on a predefined threshold [8].

Post-pruning splits the training set into two subsets: growing set and pruning set. Compared to pre-learning, post-pruning algorithms ignore overfitting problems during the learning process on growing set. Instead, they prevent overfitting through deleting conditions and rules from the model generated during learning. This approach is much more accurate, and however, less efficient [8].

Lots of previous works have proven the effectiveness of this strategy in inducing decision tree learning, such as CN2 [9], FOIL [10], and Fossil [11], Reduced Error Pruning (REP) [12], GROW [13] and J-pruning based on information-theoretic J-Measure [14]. Furthermore, knowing that pre-pruning is more efficient and post-pruning is much more accurate, researchers proposed some variants combining or integrating pre-pruning and post-pruning are introduced to improve the accuracy and efficiency of classification, like Top-Down Pruning (TDP) and Incremental (I-REP) [15].

4. Expansion of the training data

In machine learning, algorithm is not the only key affecting the final classification accuracy. Its performance can be significantly affected by the quantity and quality of training dataset in many cases, especially in the area of supervised learning.

Model training is actually a process of tuning its hyper-parameter. Well-tuned parameters make a good balance between training accuracy and regularity, and then inhibit the effect of overfitting, as well as that of under-fitting. To tune these parameters, the model needs sufficient samples for learning. The amount of samples is proportional to the number of parameters. And the more complicated model you have, the more parameters need to be tuned. In another word, an expanded dataset can improve prediction accuracy to a great extent, especially in complicated models. That is why data augmentation is widely used and proved to be effective as a general strategy to improve models' generalization performance in many application areas, such as pattern recognition and image processing. However, the enormous size of data will definitely increase training time. Moreover, training data can be expensive or difficult to acquire because it needs lots of human intervention, like labeling. To address this issue, we can manipulate the existing data to generate some new data. In short, there are mainly four approaches [16-18] to expand the training set:

1) Acquire more training data
2) Add some random noise to existing dataset
3) Re-acquire some data from existing data set through some processing
4) Produce some new data based on the distribution of existing data set

5. Regularization

Generally, the output of a model can be affected by multiple features. When the number of features increases, the model becomes complicated. An overfitting model tends to take all the features into consideration, even though some of them have very limited effect on the final output. Or even worse, some of them are noises which are meaningless to the output.

In order to limit these cases, we have two kinds of solutions:

1. Select only the useful features and remove the useless features from our model
2. Minimize the weights of the features which have little influence on the final classification

In another word, we need to limit the effect of those useless features. However, we do not always know which features are useless, so we try to limit them all by minimizing the cost function of our model. To do this, we add a "penalty term", called *regularizor*, to the cost function as shown in the following formula:

$$\tilde{J}(\omega; X, y) = J(\omega; X, y) + \alpha\Omega(\omega) \tag{3}$$

$$\tilde{J}(\omega; X, y) = \frac{1}{2m}||X_w - y||_2{}^2 + \alpha\Omega(\omega) \tag{4}$$

where $J(\omega; X, y)$ is the original cost function, ω is the weight, X is the training set, y is the labeled value (true value), m is the size of training set, α is the regularization coefficient, and $\alpha\Omega(\omega)$ is the penalty term.

Here we can use the "Gradient-Descent" method to find out the set of weights.

$$\omega^{(k+1)} = \omega^{(k)} - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(p\left(x^{(i)}\right) - y^{(i)}\right)x^{(i)} \quad - \lambda\frac{\omega^{(k)}}{m} \tag{5}$$

$$\omega^{(k+1)} = \omega^{(k)}\left(1 - \frac{\lambda}{m}\right) - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(p\left(x^{(i)}\right) - y^{(i)}\right)x^{(i)} \tag{6}$$

As shown in the formula (4), the bigger m is, the smaller $\lambda\frac{\omega^{(k)}}{m}$ will be. In another word, the bigger the training set is, the less risk of overfitting and regularization effect there will be.

In the following section, we will discuss in detail 3 general regularization methods: 1) L1 Regularization; 2) L2 Regularization; and 3) Dropout.

5.1 L1 regularization

To figure out the minimum of cost function, L1 regularization uses the Lasso Regression, one linear regression theory. In this approach, we use the so-called taxi-cab distance, sum of absolute values of all the weights as the penalty term.

$$\Omega(\omega) = ||w||_1 = \sum_i |w_i| \tag{7}$$

To minimize the cost function, we need to set the weights of some features to be zero. In another word, we remove some features from our model, and keep only those features more valuable. In this way, we can get a simpler model which are easier to interpret. However, at the same time, we lost some useful features which have lower influence on the final output.

5.2 L2 regularization (Weight Decay)

L2 regularization uses the "Ridge Regression" theory. In this approach, we use the Euclidean distance as the penalty term.

$$\Omega(\omega) = \left||w|\right|_2 = \sqrt{\sum_i w_i^2} \tag{8}$$

Compared to L1 regularization, this approach makes the networks prefers to learn features with small weights. Instead of rejecting those less valuable features, we give them lower weights. So, we get as much information as possible. Large weights can only be given to the features that considerably improve the initial cost function.

5.3 Dropout

Dropout is a popular and effective technique against overfitting in neural networks. The initial idea of dropout is to randomly drop units and relevant connections from the neural networks during training. This prevents units from co-adapting too much. The general process is as follow:

1) Drop randomly half of the hidden neurons to construct a new simpler network
2) Train the thinned network using stochastic gradient descent

The gradients for each parameter are averaged over the training cases in each mini-batch. Any training case which does not use a parameter contributes a gradient of zero for that parameter [19].

3) Restore the removed neurons
4) Randomly remove half of hidden neurons from the restored network to form a new thinned network
5) Repeat the process above until get an ideal set of parameters

By temporarily removing some units from neural nets, dropout approximates the effect of averaging the predictions of all these thinned networks. In this way, overfitting are inhibited to some extent while combining the predictions of many different large neural nets at test time. Besides, dropout reduces significantly the amount of computations. This makes it an effective choice for big or complicated networks which need lots of training data.

In the work of David Warde-Farley, Ian J. Goodfellow, Aaron Courville, and Yoshua Bengio [20], they investigated the efficacy of dropout, focusing on the specific case of the popular rectified linear nonlinearity for hidden units. And finally, they proved that dropout was an extremely effective ensemble learning method, paired with a clever approximate inference scheme that was remarkably accurate in the case of rectified linear networks.

6. Conclusion

Overfitting is general issue in supervised machine learning, which cannot be completely avoided. It happens because of either the limits of training data, which can have a limited size or include plenty of noises, or the constraints of algorithms, which are too complicated, and require too many parameters.

Responding to these causes, a variety of algorithms are introduced to reduce the effect of overfitting. On the one hand, to deal with noises in training set, algorithms based on the "early-stopping" strategy help us to stop training before learning noises; besides, algorithms based on the "reduce the size of network" strategy provide us an approach to reduce noises in the training set. On the other hand, "data-expansion" strategy is proposed for complicated models which require plentiful data to fine-tune their hyper-parameters. Besides, "Regularization"-based algorithms help us distinguish noises, meaning and meaningless features and assign different weights to them.

To deal with real-world problems, the majority of models are complicated, because generally, the final output can be affected by dozens of or even hundreds of factors. A well-generalized model will be prone to take into consideration of all the potential features instead of arbitrarily discarding the useless-like ones. The increase of parameters demands a great amount of training data to tune the hyper-parameters set, such as the weights. Thereby, data becomes a key point in machine learning, especially in supervised machine learning. In most cases, the more training data we use for training, the more accurate our final model is. However, until this moment, the acquisition of data is still a painful topic in machine learning. In many fields, data can be really expensive, because some data is difficult to obtain, and some data requires human labor for labeling. What's more, a perfect training is not great in size, but also includes a limited proportion of noise. Thus, data-acquisition and data-cleaning can be two interesting research directions for future works.

References

[1] https://elitedatascience.com/overfitting-in-machine-learning
[2] Paris G., Robilliard D., Fonlupt C. (2004) Exploring Overfitting in Genetic Programming. Artificial Evolution, International Conference, Evolution Artificielle, Ea 2003, Marseilles, France, October. DBLP, pp.267-277.
[3] Jensen D D, Cohen P R. (2000) Multiple Comparisons in Induction Algorithms. Machine Learning, 38(3):309-338.

[4] Raskutti G, Wainwright M J, Yu B. (2013) Early stopping for non-parametric regression: An optimal data-dependent stopping rule. Communication, Control, and Computing. IEEE, pp.1318-1325.
[5] Nielsen M A. (2015) Neural networks and deep learning. Determination Press.
[6] Caruana R, Lawrence S, Giles L. (2000) Overfitting in neural nets: backpropagation, conjugate gradient, and early stopping. International Conference on Neural Information Processing Systems. MIT Press, pp.381-387.
[7] Wu H, Shapiro J L. (2006) Does overfitting affect performance in estimation of distribution algorithms. Conference on Genetic and Evolutionary Computation. ACM, pp.433-434.
[8] Fürnkranz J. (1997) Pruning Algorithms for Rule Learning. Machine Learning, 27(2):139-172.
[9] Clark P, Niblett T. (1989) The CN2 Induction Algorithm. Machine Learning, 3(4):261-283.
[10] Quinlan J R, Cameron-Jones R M. (1993) FOIL: A midterm report. Machine Learning: ECML-93. Springer Berlin Heidelberg, pp.1-20.
[11] Fürnkranz J. (1994) FOSSIL: A robust relational learner. Machine Learning: ECML-94. Springer Berlin Heidelberg.
[12] Pazzani M J, Brunk C A. (1991) Detecting and correcting errors in rule-based expert systems: an integration of empirical and explanation-based learning. Academic Press Ltd.
[13] Cohen W W. (1993) Efficient Pruning Methods for Separate-and-Conquer Rule Learning Systems. International Joint Conference on Artificial Intelligence, pp.988-994.
[14] Bramer M. (2002) Using J -pruning to reduce overfitting in classification trees. Knowledge-Based Systems, 15(5–6):301-308.
[15] Furnkranz J. (1994) A Comparison of Pruning Methods for Relational Concept Learning, pp. 371--382.
[16] Sun Y., Wang X, Tang X. (2014) Deep Learning Face Representation from Predicting 10,000 Classes. Computer Vision and Pattern Recognition. IEEE, pp.891-1898.
[17] Karystinos G N, Pados D A. (2000) On overfitting, generalization, and randomly expanded training sets. IEEE Transactions on Neural Networks, 11(5):1050.
[18] Yip K Y, Gerstein M. (2009) Training set expansion: an approach to improving the reconstruction of biological networks from limited and uneven reliable interactions. State of the art of air pollution control techniques for industrial processes and power generation. Dept. of Civil Engineering, College of Engineering, University of Tennessee, pp.243-250.
[19] Srivastava N, Hinton G, Krizhevsky A, et al. (2014) Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1):1929-1958.
[20] Warde-Farley D, Goodfellow I J, Courville A, et al. (2013) An empirical analysis of dropout in piecewise linear networks. Computer Science.