Artificial Neural Networks and Overfitting

B.J Lascurain1, K.P. Bush2, M. Zuberi3, B. Parker4, and N. Taylor5

University of Louisville, Computer Information Systems, Louisville, KY 40292

\*Contact: bjlasc01@gmail.com, phone +1-352-293-6571

Abstract— Artificial intelligence comes in many different forms. One popular form is mimicking the design of natural neural networks to create an Artificial Neural Network (ANN). ANNs processes normalized inputs and output the category with a confidence level expressed as a percentage. ANNs usually are tailored for specific tasks such as predicting customer behavior or sorting materials at a recycling plant. Often ANNs have a predetermined structure. This structure has inputs within an input layer, some number of middle hidden layers, and an output layer that aggregates the final middle layer’s node’s outputs into a decision of categorization. The basis for ANNs that use supervised machine learning is to have a training dataset, a test dataset, and an optional validation dataset. These datasets are fed into the ANN through a flat file or database. The importance of the quality of the data you train a neural network with cannot be overstated. All the decisions made by the ANN will be based on the representativeness of the data to the actual cases that will be encountered. When creating these training and testing datasets care should be given not to exclude data that may be thought unimportant because it could have a hidden importance in classification. This research will analyze overfitting and the ways an artificial neural network can avoid or overcome it.

Keywords—artificial neural network, artificial intelligence, overfitting, regression, training datasets

1. Introduction

Overfitting can be described as a modeling error when a function is too closely fit to a limited set of data points. This can cause problems because it can fail to fit additional data or predict future observations reliably. In order to train an ANN system, we ideally input a model of sufficient datasets so that it can begin to recognize patterns in these datasets in hopes that it will be able to make an accurate prediction. After we train ANN on a training dataset, we then introduce a test dataset. If trained effectively, when shown our training dataset the ANN will categorize the inputs correctly with high accuracy. When introduced to a test dataset the network may produce a significantly lower output accuracy assuming our neural net was not properly generalizing between the training dataset and the test dataset; this is known as overfitting.

When working with supervised AI, a signal is an underlying pattern that we want our ANN to learn from a dataset, while noise is meaningless data and outliers. A well-created algorithm allows the AI to separate the noise from the signal creating accurate outputs. If the algorithm has too many input features or is not properly regularized it will result in memorizing the noise instead of finding the correct pattern or the signal. This causes the model to make predictions based on the noise, in training data the AI performs extremely well when tested against its training dataset, but will perform inaccurately when presented with unseen data. This happens if the model is too complex or too simple meaning it has a high bias or high variance. A key to avoiding overfitting data models is to find the balance of sufficient complexity and simplicity.

Overfitting is more likely with non-parametric and nonlinear models- many of these models also include parameters or techniques to limit and constrain how much detail the model learns. With multiple hidden layers in ANNs, they are capable of learning very complicated relationships between their inputs and outputs with limited training data. Noise in training results in the model learning anomalous patterns that detract from its ability to generalize its inputs.

Overfitted models are less portable, if at all, and don’t have the ability to be reused or applied in different situations or scenarios. Their ability to generalize is diminished.

1. Methods of Reducing Overfitting Datasets

There are lots of issues that we face when it comes to overfitting. Fortunately, there are methods we can use to help reduce the degree of overfitting. These methods include cross-validation, early stopping, ensembling, removing features, and training with more data.

“With cross-validation, you double-check how accurate your model is on multiple different subsets of data, making sure it will generalize well to data you collect in the future” [7]. Normal validation is very useful because it determines how much noise and bias is being picked up on. It uses error estimation of the training data to determine training error, which tells the reader how much error is within the data. The concept of cross-validation is to generate multiple smaller tests out of the initial training data. These smaller tests are called folds, which set the number of iterations. Each iteration will train the algorithm on the respective fold except the final fold will be for the holdout data. Cross-validation will allow the test set to remain unseen by allowing you to tune the original training hyperparameters.

There are many advantages to using cross-validation.  When using cross-validation it allows the use of all the data for both training and testing. When the data is split in a typical 80-20% split, then only 20% is used to test your data and 80% is used to train your data. Depending on the number of classes involved there can be an even greater reduction in the amount of data of which tests may be performed. With cross-validation 100% is used for both training and testing among multiple iterations. Using all the data in the process will lead to a more efficient outcome.

If only one test is done with normal validation there is nothing to compare the result against. This is because there is only one produced result. With cross validation there are multiple tests completed, increasing confidence in the networks ability to accurately process inputs. If 5 tests are run with similar results there is high confidence that the predictions are accurate. However, in the event that the 5 tests resulted in completely different results that could show that the algorithm or data needs to be adjusted.

Some examples of methods to perform cross-validation are the holdout method, K-Fold cross-validation, and Stratified K-Fold validation. The holdout method is better than simple validation because it requires data to be taken out of the training data and used testing data. This method does not require any overhead to use but unfortunately results in high variance most of the time. K-Fold cross-validation solves for the variance issue if there is enough data. Each specified amount of data is put into a grouping and each group is used as a test set while all other groups are used as a training set together. This method reduces variance as most of the data is still used in the training set rather than being removed, similar to the holdout method. Stratified K-Fold validation is like the K-Fold cross-validation method but is used when there is a large difference in response variables. It creates groupings, where each fold uses the same percentage of samples from a target class.

Another method is early stopping. “When you’re training a learning algorithm iteratively, you can measure how well each iteration of the model performs.” [6]. Each new iteration will generate improvement up to a certain point, at which each new iteration has a negative effect. Early stopping is the practice of stopping upon reaching the point of negative effects. This can lead to stopping before reaching the maximum efficiency point, but it will allow the avoidance of extensive negative effects.

To be able to determine the early stopping point the data must be broken into 2 sets, the training set, and a validation set. Similar to a graph with accuracy on the vertical axis and iterations on the horizontal axis. As the iterations increase there should be an increase in both the training set and the validation set. As you reach the point of negative effects the training set will continue to increase in accuracy with each iteration whereas the validation set will begin to decrease in accuracy. Currently there is no way of perfectly locating the early stopping point. However, the graph described above gives a good idea of when the early stopping point may have been reached and can be implemented in future generations.

The next method is ensembling. “Ensembles are machine learning methods for combining predictions from multiple separate models” [7]. Bagging and boosting are two opposite methods of many methods of ensembling. Bagging uses “strong” learners which it trains and combines in order to level out the predictions, which should help to reduce the chance of overfitting. Boosting uses “weak” learners that learn for the one before it and then they are combined into a “strong” learner, which should help increase the prediction capabilities of simple models.

Another ensembling method worth mentioning is called stacking. The concept of stacking is using information from previously generated predictive models to generate a new model.  The new model’s performance will be more efficient than all the models combined to make it. Each of the previously generated models used will be reviewed to see which aspects of each model performed the best and which aspects performed the worse. The fewer similarities among the previous models, the better the new model will perform.

Another way to counter overfitting is by using larger datasets. This method is based on the concept of providing more training data to increase the accuracy of the model. If a model has more training data, then it has more factors to apply and is forced to generalize its selections. It is important to note that the data must be of good quality, simple adding irrelevant data for the sake of adding data will end up not be much of a help.

Removing variables is another reliable method.  Sometimes a variable will do a good job while making predictions but doesn’t carry any real-life value. Sometimes a variable will be biased toward a group in the set that natural has a dominating aspect of that variable. In these occasions the variable is good, but in the particular datasets, it causes a result in conflicting data. Removing conflicting or irrelevant variables will decrease the chance of overfitting and increase accuracy.

The methods for combating overfitting that have been described thus far are only a few examples. Combating overfitting is very important to get accurate results in your predictions. As more methods are discovered the performance of the prediction will increase substantially giving way to a much more powerful artificial intelligence agent.

1. Testing Prediction Accuracy

Additionally, the quality of the data is important when determining datasets. If the data is not of the best quality, bias can be expected. In the past, there have been many examples of machine learning algorithms using datasets that may have given them unintentional bias. One such example is Google's job searching algorithm showing more prestigious jobs to men over women.

Data testing is one of the most important parts of determining what data to use in the datasets. This is because it makes sure that the accuracy of the data is predictable. Data scientists often test the prediction accuracy of a group of data with cross-validation. The goal of cross-validation is used to make sure that the majority of data will be correct when incorporating real-world data, rather than just training datasets. Validation is very useful because it determines how much noise and bias is being picked up on. It uses error estimation of the training data to determine the training error, which tells the reader how much error is within the data. The way to determine the predicted error when looking at unseen data uses cross-validation methods, such as the holdout method, K-Fold cross-validation and stratified K-Fold validation. The holdout method is better than simple validation because it requires data to be taken out of the training data and used to train the rest of the data. This method does not require any overhead to use but unfortunately results in high variance most of the time. K-Fold cross-validation solves for the variance issue if there is enough data. Each specified amount of data is put into a grouping and each group is used as a test set while all the other groups are used as a training set together. This method reduces variance as most of the data is still used in the training set rather than being removed like the holdout method. Stratified K-Fold validation is similar to the K-Fold cross-validation method but is used when there is a large difference in response variables. It creates groupings, where each target class uses the same percentage of samples from a target class. These validation types are called non-exhaustive cross-validation methods.

1. Accuracy With Only Part of the Picture

Artificial neural networks are known to be accurate, but not to the extent of traditional analyzation techniques. One reason this viewpoint is commonly held is that attributes within artificial neural networks are sometimes omitted or simply missing data from the network. The most common way of dealing with this to improve accuracy by creating constant values in place of omitted values. Although constant values are the simplest option, others that are more complex might utilize regression or attribute prediction.

The opposite concept of omitted data, overfitting is also a common issue involved in artificial neural networks. The idea of cramming too many attributes into a neural network is more common than dealing with missing data. Overfitting is significant to neural networks because it negatively impacts the accuracy of a network. Smaller issues such as data redundancy or potentially inaccurate data, stem from a network that is overly trained resulting in overfitting.

There are multiple methods of combating overfitting in neural networks. “One method of combating overfitting is called regularization. Regularization modifies the objective function that we minimize by adding additional terms that penalize large weights.” [3] There are multiple types of regularization within artificial neural networks, L1 and L2.

“The most common type of regularization is L2 regularization. It can be implemented by augmenting the error function with the squared magnitude of all weights in the neural network. Another common type of regularization is L1 regularization. Here, we add the term λ|w| for every weight win the neural network. The L1 regularization has the intriguing property that it leads the weight vectors to become sparse during optimization” [3].

L1 regularization is completed by adding the term λ|w| for every weight in a neural network. According to Buduma, neurons with L1 regularization end up using only a small subset of their most important inputs and become quite resistant to noise in the inputs. On the other hand, weight vectors from L2 regularizations are usually small numbers. With that being said, each type is more useful than the other in specific situations. "L1regularization is very useful when you want to understand exactly which features are contributing to a decision. If this level of feature analysis isn't necessary, we prefer to use L2 regularization because it empirically performs better.” [3]

Penalizing the large weights of attributes is what is important to take away from the concept of regularization with artificial neural networks. The specific methods (L1 and L2) are much more detailed concepts for combating overfitting rather than the concept of regularization itself.

Aside from regularization and its different types, there is also 'Max norm constraints' and 'Dropout' as two other methods of combating overfitting. "Max norm constraints are a much more direct approach to restrict 'θ' from becoming too large. Max norm constraints enforce an absolute upper bound on the magnitude of the incoming weight vector for every neuron and use projected gradient descent to enforce the constraint. In other words, anytime a gradient descent step moved the incoming weight vector such that ||w||2 >c, we project the vector back onto the ball (centered at the origin) with radius c” [3]. With that being said, it is a benefit of Max norm constraints that the parameter vector cannot grow out of control, even if the learning rates are too high for the network to support because the updates to the weights are always bounded.

The Dropout method of combating overfitting in artificial neural networks is described as "The Dropout method prevents overfitting by providing a way of approximately combining exponentially many different neural network architectures efficiently" [3]. But how does it do this? While training, dropout is implemented by only keeping a neuron active with some probability p (a hyperparameter) or setting it to zero otherwise. This forces the network to be more accurate even when missing certain information, making it even more relaible. By dropping nodes inner layer nodes at random we create a more resilient model by reducing dependencies on any one inner layer. The dropout method is regarded as one of the most effective methods for combating overfitting because of its simplicity and effectiveness.

The bottom line between both omitted attributes and overfitted networks is the importance of the entire networks and data’s accuracy. This is why the accuracy of the artificial neural network’s output is often brought into question.

1. Conclusions

The importance of having good training and testing datasets coupled with effective methods of countering overfitting cannot be overstated. The effectiveness of an artificial neural network will be determined entirely by its ability to generalize never-before- seen data. A network that fits test data with 100% accuracy, but fails to generalize unseen data is worthless in application. Each method of countering overfitting has clear drawbacks and benefits that must be considered when designing a particular neural network. There is no one-size-fits-all approach.

References

1. Brownlee, J. (2018). Overfitting and Underfitting With Machine Learning Algorithms. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/.
2. Babyak, Michael A. “What You See May Not Be What You Get: A Brief, Nontechnical Introduction to Overfitting in Regression-Type Models.” Psychosomatic Medicine, vol. 66, no. 3, 2004, pp. 411–421., doi:10.1097/00006842-200405000-00021.
3. Buduma, N. (2015). Data Science 101: Preventing Overfitting in Neural Networks. Retrieved November 6, 2018, from https://www.kdnuggets.com/2015/04/preventing-overfitting-neural-networks.html
4. Liang, Jianglin, and Ruifang Liu. “Stacked Denoising Autoencoder and Dropout Together to Prevent Overfitting in Deep Neural Network.” 2015
5. Hawkins, Douglas M. “The Problem of Overfitting.” Journal of Chemical Information and Computer Sciences, vol. 44, no. 1, 2004, pp. 1–12., doi:10.1021/ci0342472. (2002) The IEEE website. [Online]. Available: http://www.ieee.org/.
6. EliteDataScience. (2018). Overfitting in Machine Learning: What It Is and How to Prevent It. [online] Available at: https://elitedatascience.com/overfitting-in-machine-learning [Accessed 9 Nov. 2018]. *FLEXChip Signal Processor (MC68175/D)*, Motorola, 1996.
7. DataRobot. (2018). Cross-Validation | DataRobot Artificial Intelligence Wiki. [online] Available at: https://www.datarobot.com/wiki/cross-validation/ [Accessed 9 Nov. 2018].
8. Kdnuggets.com. (2018). Data Science 101: Preventing Overfitting in Neural Networks. [online] Available at: https://www.kdnuggets.com/2015/04/preventing-overfitting-neural-networks.html [Accessed 9 Nov. 2018].
9. “Datasets and Machine Learning,” Skymind. [Online]. Available: https://skymind.ai/wiki/datasets-ml. [Accessed: 03-Nov-2018].
10. “Preparing Your Dataset for Machine Learning: 8 Basic Techniques That Make Your Data Better,” AltexSoft. [Online]. Available: https://www.altexsoft.com/blog/datascience/preparing-your-dataset-for-machine-learning-8-basic-techniques-that-make-your-data-better/. [Accessed: 03-Nov-2018]
11. P. Gupta, “Cross-Validation in Machine Learning – Towards Data Science,” Towards Data Science, 05-Jun-2017. [Online]. Available: https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f. [Accessed: 09-Nov-2018].
12. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std. 802.11, 1997.