A Comparative Analysis of Various Regularization Techniques to Solve Overfitting Problem in Artificial Neural Network

Shrikant Gupta[1](✉), Rajat Gupta[1], Muneendra Ojha[1], and K. P. Singh[2]

[1] Dr. SPM International Institute of Information Technology Naya Raipur, Naya Raipur, India
{shrikant15101,rajat15101,muneendra}@iiitnr.edu.in
[2] Indian Institute of Information Technology Allahabad, Allahabad, India
kpsingh@iiita.ac.in

Abstract. Neural networks having a large number of parameters are considered as very effective machine learning tool. But as the number of parameters becomes large, the network becomes slow to use and the problem of overfitting arises. Various ways to prevent overfitting of model are further discussed here and a comparative study has been done for the same. The effects of various regularization methods on the performance of neural net models are observed.

Keywords: Dropout · L1 regularization · L2 regularization · Neural network Overfitting

1 Introduction

Artificial neural network contains multiple non-linear hidden layers which makes it possible for them to learn very complicated relationships between inputs and outputs. One of the problems that occur during neural network training is overfitting which causes them to perform very poorly. In case of overfitting it is observed that the error at the time of training the datasets reaches to a very insignificant value, but when new data is presented to the network the error produced is large. This is so because the network has learned the training examples, but it has not learned to generalize to new situations. Regularization is a way to reduce the problem of overfitting. A similar work is also done previously with respect to dropout by Srivastava et al. [1]. They have improvement in performance of models using dropout and obtained state-of-the-art results on many supervised learning tasks.

Performance of different regularization techniques on ImageNet is compared by Evgeny A. Smirnov et al. in the paper [2]. They have shown empirically that performance of Dropout on ImageNet Dataset is better than DropConnect. Implementation of neural network on Airfoil Noise is also shown by Lau et al. in his paper [3]. DropConnect is introduced in paper [4] which is a generalization of Dropout. It is used for large fully-connected hidden layers of the neural network. DropConnect is evaluated in the paper [4] on a range of datasets and it is observed that for some datasets DropConnect outperforms Dropout.

A brief introduction to various regularization techniques is deliberated in Sect. 2 of this paper. Section 3 contains the features of the overall model that is used here to train the datasets. Section 4 of this paper contains the observations and results after training the datasets using the discussed model. A final conclusion based on observed results has been drawn in Sect. 5 of this paper.

Artificial Neural Network (ANN) is an inspiration from biological neural networks of the human brain. ANN can be applied to wide range of field such as pattern recognition, image processing, financial data, medicine etc. As brain neurons are connected through synapses similarly nodes in neural network model are connected to each other. The ANN model contains several layers both hidden and visible. ANN models can generate a nonlinear function with the help of activation functions. There are different kinds of neural network based on network topologies and the one used here is the feed-forward neural network. The other types include Recurrent Neural Network, Convolutional neural network etc. Hypothesis in a neural network model having weight w, input x and bias b can be defined as Eq. 1.

$$h_\theta(x) = \sum_i w_i x_i + b \tag{1}$$

2 Regularization Techniques

There are several ways of controlling the problem of overfitting in a neural network. Some of them are discussed here.

2.1 L2 Regularization

L2 Regularization is one of the most general forms of regularization. It can be implemented by adding the squared magnitude of all the parameters i.e. weights and biases in the cost/loss function. This total cost function is further minimized using optimizer. That is for every weight *w* we add a term ½ λw^2 where λ is the regularization parameter. A factor of ½ is multiplied to simplify the gradient term. Using the L2 regularization ultimately means that every weight is decayed linearly towards zero.

$$L2\colon \frac{\lambda}{2} \sum_{i=1}^{m} w_i^2 \tag{2}$$

2.2 L1 Regularization

L1 is another common form of regularization, where instead of adding the square of the weights we just add the absolute value of the weights to the cost function.

$$L1: \lambda \sum_{i=1}^{m} |w_i| \tag{3}$$

Sometimes both L1 regularization and L2 regularization are applied together (it is called Elastic-net regularization). L2 regularization generally gives better performance than L1 regularization. Comparison between L1 regularization and L2 regularization is done by Ng [5]. In his paper, he compared L1 and L2 regularization and showed how L1 regularization of parameters grows logarithmically while L2 regularization with Neural Networks and SVM worst case sample grows linearly in a number of irrelevant features.

2.3 Dropout

Dropout is a recently introduced regularization technique by Srivastava et al. [1]. Dropout is a simple but extremely effective regularization technique. It prevents complex co-adaptations on training data in a neural net. The term "dropout" refers to dropping out hidden units and visible units in a model i.e. both hidden layer and input layer. The idea behind dropout neural network can be understood by Fig. 1.

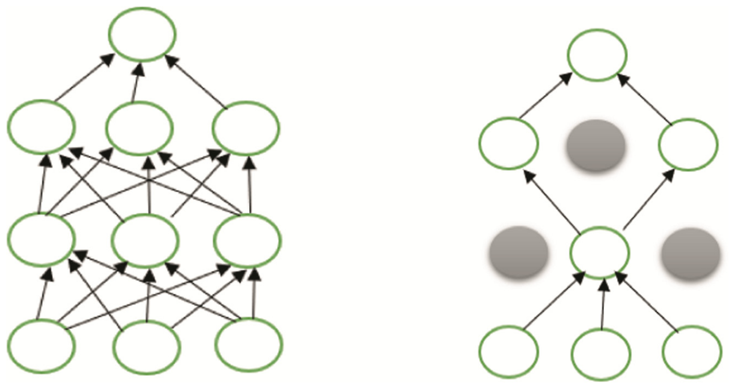

Fig. 1. Dropout neural net model (the black circle represents dropped units)

Dropping a unit means temporarily removing the unit along with all its connections from the network, as shown in Fig. 1. The decision on which unit to drop or remove is totally random. Generally, each unit in the model is kept with a probability p which is fixed and independent of other units in the model. The value of p can simply be set to 0.5 while for input units it is usually kept closer to 1 than to 0.5 (nearly 0.8).

At training time, the unit is present with probability p and is connected with weight w to the units in the next layer while at test time, the units are always present but the weights are multiplied with probability p. Therefore, the output at test time is same as expected output at the time of training.

3 Setting Up of Overall Model

Before running the model on dataset, the model and data needs to be properly processed to yield best result. In this section, we have discussed about the overall model after applying regularization. Following subheading gives a brief info about the additional design choices regarding data preprocessing, weight initialization, and loss functions.

3.1 Data Preprocessing

The two-common techniques of data preprocessing used here are Mean subtraction and Normalization. Preprocessing.scale() function of TensorFlow library does both mean subtraction and normalization.

Mean Subtraction. Mean Subtraction involves subtracting the mean (μ) across every individual feature i.e. replacing x_i with $x_i - \mu_i$. This in turn centralizes the data around origin and makes the features have approximately zero mean.

$$x_i \rightarrow x_i - \mu_i \tag{4}$$

Normalization. Normalization refers to normalizing the data dimensions to bring the data on the same scale. One way to achieve normalization is by dividing each dimension by its standard deviation. The scaled data will have zero mean and unit variance.

3.2 Weight Initialization

Before training the model initialization of the parameters is required. It is important to select appropriate initial values for the weights and biases. A good guess is to initialize all the parameters to zero but there will be no asymmetry between the neurons as they will compute same output and same gradient during backpropagation. As a result, they will go same parameter update and all the neurons will become identical. Therefore, a good solution is to initialize the parameters with small random numbers close to zero. This will make all the neurons random and unique.

3.3 Activation Function

Activation function defines the output of a node when an input or set of input is given to that node. They provide a different type of nonlinearities in the model. There are many known activation functions like linear, tanh, softmax, sigmoid, relu etc. but the activation function used in Sect. 3 for observations are relu and softmax.

Relu stands for Rectified Linear Unit. It returns the maximum of input and zero i.e. if the input of a unit is negative it will output zero else it will pass the input. Relu is used here in the regression problem (Airfoil dataset) and in the hidden layers of the model built for MNIST dataset.

$$Relu(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{5}$$

Softmax activation function is used in the output layer of MNIST dataset. It is used to assign probabilities to the units of output layer in classification problem as they give us a list of values that are between 0 and 1 and sum up to 1. Softmax is the final layer in our model for MNIST dataset.

$$softmax(x)_i = \frac{e^{(x_i)}}{\sum_j e^{(x_j)}} \tag{6}$$

3.4 Regularization

In Sect. 2 various regularization techniques have been discussed. Here we have used regularization functions present in Tensorflow library for implementing regularization. Regularization is applied to the loss/cost function to reduce overfitting.

3.5 Loss Functions

Mean Squared Error (MSE). To calculate loss, the sum of squares of the difference between predicted value and the actual value is taken for all the training example and it is further reduced by taking mean. Squaring the difference makes the error positive as we do not want to add negative error in our cost function. Another benefit is that it results in a function which has a single local minimum i.e. its global minima is same has local minima.

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 \tag{7}$$

Where,

$$h_\theta\left(x^{(i)}\right) - \text{Predicted output}$$

$$y^{(i)} - Actual\ value$$

Cross-Entropy Error Function. During the process of learning, the neural network model goes through stages in which the reduction of error may become extremely slow. This stagnation period can impact learning time. In order to resolve this problem a better way is to replace the mean squared error (MSE) by cross-entropy function. The bounds of the squared error and cross entropy are further compared by Pavel et al. [6]. They have presented an investigation on the properties of the cross-entropy (CE) and mean squared error (MSE) criteria for training models in Neural net.

$$J(\theta) = -\sum_i h_\theta\left(x^{(i)}\right)\log\left(y^{(i)}\right) \tag{8}$$

The optimizers are used to compute the gradients for loss and apply those gradients to the weights and biases. The optimizers used in this paper are Gradient Descent optimizer and Adam optimizer. Gradient descent optimizer applies the gradient descent algorithm which is a well-known algorithm in machine learning field. On the other hand, Adam optimizer implements Adam algorithm. Both optimizers require a learning rate α to train the data which should neither be too large or too small. Taking a small learning rate will make the algorithm very slow while a large learning rate will result into an algorithm which will never reach to minima. The mathematics behind Adam optimizer is introduced in the paper by Kingma et al. [7].

3.6 Model Evaluation

Error Analysis. Percentage relative error is used for the comparison between different models. The absolute error for each testing example i.e. the |predicted value – true value| is divided by true value and it is then converted it into percentage form. The overall mean of error for each testing example is obtained which gives us the final % error of the model.

$$\frac{|Predicted\ Value - Actual\ Value|}{|Actual\ Value|} \times 100 \tag{9}$$

Accuracy. To figure out that we predicted the label correctly argmax function of TensorFlow is used which gives the index of the highest entry in a tensor along some axis. This function is applied to both actual values and predicted values and the outputs of the function are compared using the equal function in TensorFlow. The Equal function generates a Boolean output which is further cast into float value and its mean is taken.

In Sect. 4 the Airfoil Dataset is a regression problem so the model is evaluated using percentage relative error while MNIST is a classification problem and is evaluated using the accuracy of the model.

4 Implementation on Datasets

4.1 Airfoil Self-noise Data Set

Airfoil Self Noise Dataset is taken from UCL Machine Learning Repository Donated by Thomas et al. [8].

In Air dataset initially preprocessing was done to bring the features on the same scale. Two hidden layers of 100 and 10 units were used respectively. Relu activation function is applied to the hidden layers while the output layer is linearly activated. The model is trained for 10000 epochs with a learning rate of 0.01. Optimizer used for optimizing the cost is Adam Optimizer and the error function used is Mean Squared error (MSE). For evaluating the model error analysis is done by comparing percentage relative error.

On applying various regularization technique in air dataset, it is observed that the performance of L2 Regularization is better when applied to both weights and biases. The percentage relative error of various regularization techniques is shown in Fig. 2. Dropout here yield maximum error percentage and therefore is not suitable for datasets with fewer attributes.

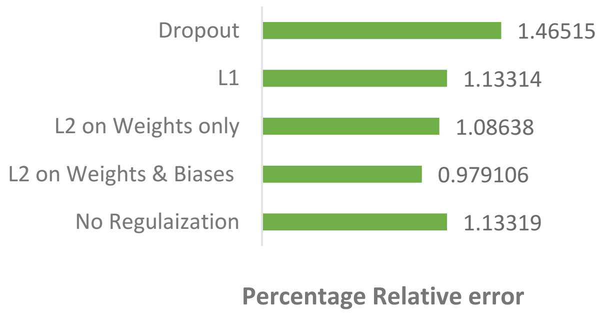

Fig. 2. Percentage relative error in different regularization technique

The performance of the model was further tested on L2 regularization by varying the regularization parameter lambda. It was observed that the best performance was shown by the model on $\lambda = 0.0001$ and on decreasing the value performance degrades (Fig. 3).

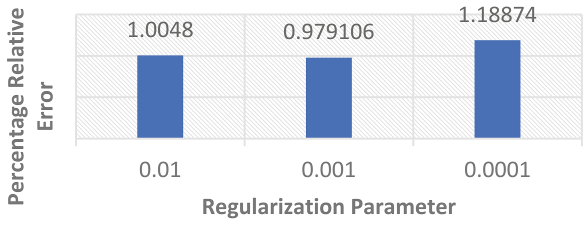

Fig. 3. Variation of error with regularization parameter in L2 regularization

4.2 MNIST Dataset

MNIST Dataset is an image dataset consisting the images of handwritten digits from 0–9. It has 28×28 matrix and 10 classes. That means the total number of features are 784 and labels are of shape 1×10 in one hot form. Dataset is taken from THE MNIST DATABASE of handwritten digits [9].

MNIST Dataset does not need to be preprocessed as the features are nearly on the same scale. The model consists of three hidden layers with 500 units in each of the hidden layers. As MNIST is a computer vision dataset of images of handwritten digits so the possible outcome can be a number from 0–9. Therefore the output class consists of 10 units which output the probability of each number. Relu is applied to all the hidden layers while in the output layer softmax is applied as we want to the probabilities to add up to one. The Gradient descent optimizer is used here to optimize the cost and it is trained for 10 epochs. The accuracy of the model is measured using the inbuilt function of TensorFlow as discussed in Sect. 3 As the dataset is very large the inputs are given in batches (Fig. 4).

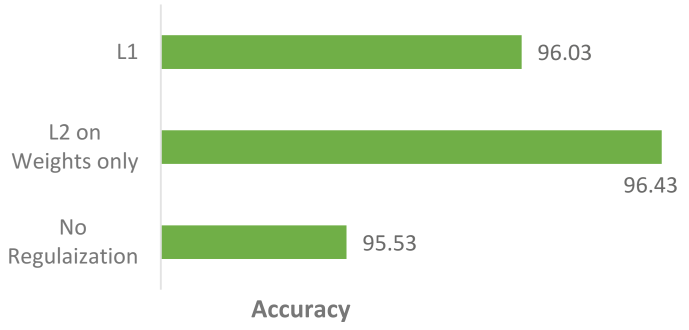

Fig. 4. Accuracy in different regularization techniques

On training the model on MNIST Dataset it was observed that performance of L2 Regularization is better than L1. The performance increased after applying L1 regularization as compared to a model with no regularization (Fig. 5).

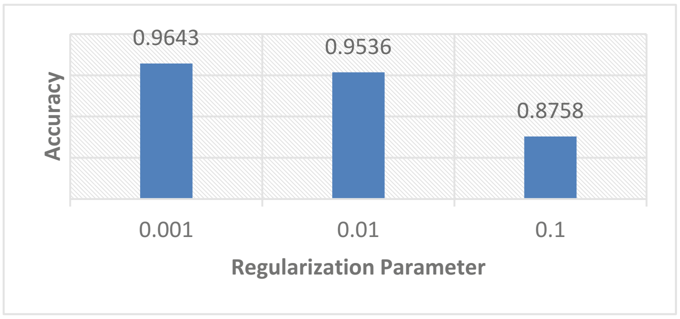

Fig. 5. Variation of accuracy with regularization parameter

On L2 Regularization the effect of regularization parameter λ was investigated and it was observed that it works best when λ is near 10^{-3}.

5 Conclusion

The paper described a detailed overview of a neural network model and its working. The problem of overfitting and its solutions were also discussed and further compared with respect to their performance. In this work, we find that models having less number of features perform comparatively better with L2 Regularization. Regularization parameter in L2 regularization also plays a significant role and yields the better result when its value is kept small. Dropout and DropConnect can be preferred when the numbers of features are high.

References

1. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
2. Smirnov, E.A., Timoshenko, D.M., Andrianov, S.N.: Comparison of regularization methods for imagenet classification with deep convolutional neural networks. AASRI Procedia **6**, 89–94 (2014)
3. Lau, K., López, R., Oñate, E.: A neural networks approach to aerofoil noise prediction. In: International Centre Numerical Methods Engineering, vol. CIMNE No-3 (2009)
4. Wan, L., Zeiler, M., Zhang, S., LeCun, Y., Fergus, R.: Regularization of neural networks using dropconnect. In: ICML, no. 1, pp. 109–111 (2013)
5. Ng, A.: Feature selection, L1 vs. L2 regularization, and rotational invariance. In: Twenty-First International Conference Machine Learning - ICML 2004, p. 78 (2004)
6. Golik, P., Doetsch, P., Ney, H.: Cross-entropy vs. squared error training: a theoretical and experimental comparison. In: Interspeech, vol. 13 (2013)
7. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv: 1412.6980 (2014)
8. Lopez, R., Brooks, T.F., Pope, D.S., Marcolini, M.A.: Airfoil self-noise data set. UCI Machine Learning Repository (2008)
9. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of IEEE, vol. 86, no. 11, pp. 2278–2324 (1998)