STOCHASTICALLY REDUCING OVERFITTING IN DEEP NEURAL NETWORK USING DROPOUT

Nishtha Tripathi¹, Avani Jadeja²

¹M.E. Student, Computer Engineering, Hashmukh Goswami College of Engineering, GTU, India
²Assistant Professor, Computer Engineering, Hashmukh Goswami College of Engineering, GTU, India

Abstract

Deep neural networks are trained on the large number of parameters which are likely to co-adapt and overfit. Overfitting is a challenging problem in the deep neural network. Dropout training has shown a significant effect in improving deep neural network. The aim of this dissertation to study dropout and other which are built on dropout regularization methods. In real world data is noisy with i.e. missing features, unlabeled, unstructured. We will study method to distort data prior training to act as a regularizer. This will create data having a correlation with real world data. Restricted Boltzmann Machine probabilistic energy based graphical model with no interconnection between hidden to hidden units and visible to visible units. It would be stacked and Deep RBM will be formed for training.

Keywords: — Deep neural networks, Regularization, Overfitting, Distorted Distribution.

1. Introduction and Motivation

Neural networks are powerful computational models that are being used extensively for solving problems in vision, speech, natural language processing and many other areas. Fascinating and complex computational model called Deep Neural Network are inspired by human Visual Cortex. The main aim of the machine learning tasks is to build models which generalize well on unseen data. A wide difference between test and train error is known as Overfitting.

The proficiency of model (i.e. Neural Networks) is tested when it works on data which it has not been tested. Training model on the data is the most important phase for deciding the different parameters. The model should be trained considering problems related to dataset. Here we performed experiments by a method distorted distribution which distorts training and test data both to provide variety to train the model and then tested on the unseen data. It is a simple and effective method which has significant improvement in the training time as well as error reduction.

2. Deep Restricted Boltzmann Machine

![Figure 1 RBM](image)

- **Energy Function**

\[
E(v, h) = -\sum_i a_i v_i - \sum_j b_i h_i - \sum_{i,j} w_{ij} v_i h_i
\]

- **Probability of Hidden and Visible vectors**

\[
p(v, h) = \frac{1}{Z} e^{-E(v, h)}
\]

where \( Z = \sum_{v, h} e^{-E(v, h)} \)

\[\Rightarrow\] Visible to Hidden probablities

\[
p(h_j = 1 \mid v) = a_f \left(b_j + \sum_i v_i w_{ij}\right)
\]

where \( a_f = \text{activation function} \)

\[\Rightarrow\] Hidden to Visible probabilities

\[
p(v_i = 1 \mid h) = a_f \left(a_i + \sum_j h_j w_{ij}\right)
\]
Restricted Boltzmann Machine is undirected energy based graphical model. In the energy based models the energy functions distribution of the data over the hidden and visible units. The energy function is as described in the equation.

RBM can be stacked to form DRBM which can train using CD-k for a faster learning. DRBM can used various machine learning tasks like computer vision, Natural Language Programming, handwritten recognition.

3. Contrastive Divergence K-steps

Input : RBM \((V_1, \ldots, V_m, H_1, \ldots, H_n)\), training batch \(S\)

Output : Gradient approximation \(\Delta w_{ij}, \Delta b_j\) and \(\Delta c_i\)

for \(i=1, \ldots, n, j=1, \ldots, m\)

init \(\Delta w_{ij} = \Delta b_j = \Delta c_i = 0\) for \(i=1, \ldots, n, j=1, \ldots, m\)

for all the \(v \in S\) do

\(v^{(0)} \leftarrow v\)

for \(t=0, \ldots, k-1\) do

for \(i=1, \ldots, n\) do

\(\text{sample } h_i^{(t+1)} : p(h_i | v^{(t)})\)

for \(j=1, \ldots, m\) do

\(\text{sample } v_j^{(t+1)} : p(v_j | h^{(t)})\)

for \(i=1, \ldots, n, j=1, \ldots, m\) do

\(\Delta w_{ij} \leftarrow \Delta w_{ij} + p(H_i = 1 | v^{(0)}) \cdot v_j^{(0)} - p(H_i = 1 | v^{(k)}) \cdot v_j^{(k)}\)

for \(j=1, \ldots, m\) do

\(\Delta b_j \leftarrow \Delta b_j + v_j^{(0)} - v_j^{(k)}\)

for \(i=1, \ldots, n\) do

\(\Delta c_i \leftarrow \Delta c_i + p(H_i = 1 | v^{(0)}) - p(H_i = 1 | v^{(k)})\)

4. Distorted Distribution

The data will be distorted by different distributions. The distribution will define the characteristics of new features. The new distribution will in the form

\[ P(x' | x) = \prod_{n=1}^{N} P_D(x'_n | x_n ; \theta_n) \]

where,

\(x')\) are new parameter

\(\theta_n\) model parameters

\(x\) normal parameter

\(P_D\) is the type of distribution

\(P_D\) can take one of the following forms:

1. Dropout in which the \(n^{th}\) feature is randomly set to zero with probability \(p_n\);
2. Gaussian on the \(n^{th}\) feature with variance \(\sigma\)
3. Laplace \(n^{th}\) feature with variance \(\lambda\)
5. Experimental Results and Analysis

The MNIST database consists of 60,000 training and 10,000 test images of handwritten digits of size 28 by 28 pixels. For the data set, we used the training set to train RBM. The digits data used are taken from the MNIST data set [26, 31], which itself was constructed by modifying a subset of the much larger dataset produced by NIST (the National Institute of Standards and Technology).

It comprises a training set of 60,000 examples and a test set of 10,000 examples. Some of the data was collected from Census Bureau employees and the rest was collected from high-school children, and care was taken to ensure that the test examples were written by different individuals to the training examples.

We have trained 2-layer DRBM with 500-1000 hidden units and 784 visible units respectively. Data set was divided mini-batches of 100 for reducing training time. We trained model using CD-k steps. Logistic activation function used for calculations. Then the model was fine tuned using gradient calculation [39].

Evaluation Criteria:

Error %=(misclassified images * 100)/Total Images
Test Images: 10,000
Train Images: 60,000

Figure 4 Random images taken after applying distortions from left first is dropout, Lapalace and Gaussian distribution respectively.

Figure 5 Graph for evaluating Error for different layers no. of epochs as shown in Table I and with fine-tuning

<table>
<thead>
<tr>
<th>Model</th>
<th>100 Epochs</th>
<th>200 Epochs</th>
<th>DRBM 500 Epochs</th>
<th>Error DRBM %</th>
<th>Error DRBM %</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBM</td>
<td>28.31 min</td>
<td>3.23 hours</td>
<td>20.0 hours</td>
<td>0</td>
<td>1.3</td>
</tr>
<tr>
<td>DBM_Dist.</td>
<td>29 min</td>
<td>3.26 hours</td>
<td>20.4 hours</td>
<td>0</td>
<td>0.87</td>
</tr>
<tr>
<td>DBM_Dropout</td>
<td>28.21 min</td>
<td>3.27 hours</td>
<td>20.0 hours</td>
<td>0.11</td>
<td>1.03</td>
</tr>
</tbody>
</table>

Table I: Recorded time and error for running different layers and errors with fine-tuning

In the performed experiment for distorted distribution model the test and train are both distorted with the rate 0.2 % which has shown significant improvement in error rate and reduced training time also. Dropout takes the more time then the our method to reduce the error rate.

CONCLUSIONS

Deep Neural network a powerful, complex computational model for non-linear processing. The problem with DNN is that it deals with billions and millions of parameters which are likely to co-adapt. The randomized distorted methods for features and data can itself acts as regularizer. This suffices the need of adding model parameters so might possibly no extra computation added for regularizations, which in lieu can save time.

We had performed experiments using distorted distribution on the MINIST dataset. Distorted Distribution can be extended to use different types of distribution.
function. For training our model on different variety of dataset we increase dataset size by applying different distributions. Comparison with GPU performance will be a good comparison, it can explore the distribute aspect also. These methods can be extended to NLP, computer vision as well as multimodal learning.

The model of DRBM is implemented through stacking different RBM. It would be interesting to observe results by applying to different models like convolution neural net (CNN), Deep Belief Net (DBN).

References


[28] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng Reading Digits in Natural Images with Unsupervised Feature Learning NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011.

[29] Y. LeCun, F.J. Huang, L. Bottou, Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting. CVPR 2004

[31] C. M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.