Assignment # 5 (Task 3) Technical report on the sparse auto encoders

Waqar Kaleem Khan
Enrolment # 01-249191-013
Waqar.musa777@gmail.com
Department of Computer Science Bahria University – Islamabad

Abstract – This report addresses sparse auto encoder. Autoencoders are an unsupervised learning technique in deep learning. In the sparse autoencoders we don't reduce the number of neuron or hidden layer but it provide us a different technique for the bottleneck. We should preferably will build our lost function like that we penalize activations within layer. In this article we will also discuss how to use a sparse autoencoders and will also discuss about other autoencoder like undercomplete that how they work and which one is best to use in the above two autoencoders.

Keywords **–** Principle Component Analysis (PCA), Machine Learning (ML), Deep Learning (DL)

I. INTRODUCTION

In this introduction part we will first discuss little about that what auutoencoders is then we will discuss about the sparse autoencoders. Auto encoders are an unsupervised learning technique in the deep learning in which we grasp a network for the task of representation learning. Individually we will design a NN architecture and will impose a bottleneck in the network which will force a flatten knowledge representation of the original input of the network. If the given feature is independent the task would be difficult to subsequently reconstruct. And if there is some structure in the data like some relation between the features which is input so this structured can learn and grasps consequently when the input is forced through the bottlenecked network.

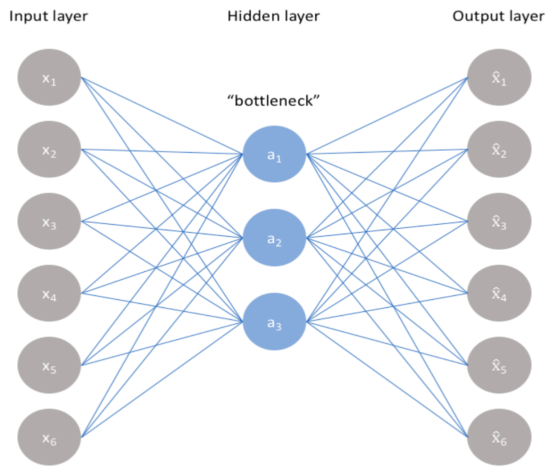

Fig. 01

(unlabeled dataset frame with supervised learning problem with bottleneck)

In the Fig 01 we can see that we can take unlabeled dataset and frame that as a problem of supervised learning which give an output of x hate after the reconstruction of the input x. The network can be trained when we minimize the reconstruction of the error L(x,x^), this measure the differences between the reconstruction and the original input. In our network the bottleneck is a key attribute our network can learn easily to simple memorize the input values to pass these input values through the network if there is no bottleneck in our network as we can see in the below Fig 02

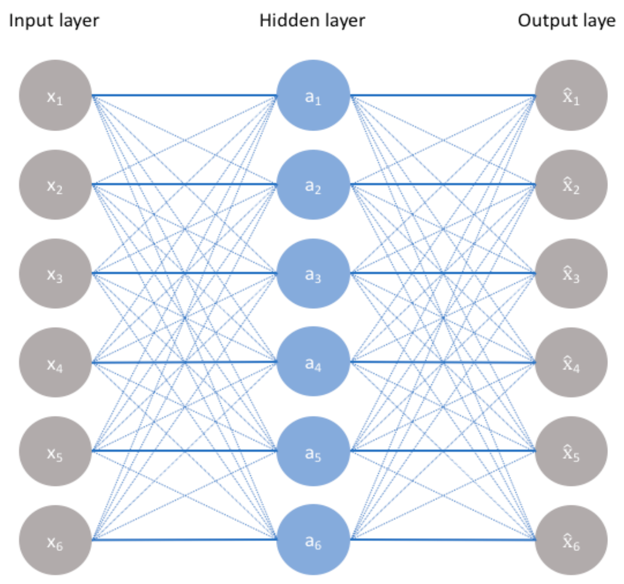

Fig. 02

(fully connected network)

The amount of information is restricted by a bottleneck which can traverse the full network, forced a learned compression of the input data.

The following model can be balances by any ideal autoencoder:

1) Enough sensitive to inputs to build a reconstruction accurately.
2) Enough intensive to inputs to doesn't overfit the training data

The model has forced by this trade-off to maintain the variations only in the required data to reconstruct the input without holding on the redundancies within the input. A loss function constructing involves in the most cases where our model encourages terms to be sensitive to the inputs and discourages second term to overfitting, L(x,x^)+regularizer.

For the adjustment of the trade-off between 2 objectives in front of the regularization term we will add a scaling parameter.

II. UNDERCOMPLETE AUTOENCODER

For autoencoder constructing the simple architecture is to constrain the hidden layer nodes of the network to limit the amount of information that will flow through the network. Our model can learn the important attributes of the input data by penalizing the network according to the reconstruction error. The describe latent attributes will be learn by this encoding.

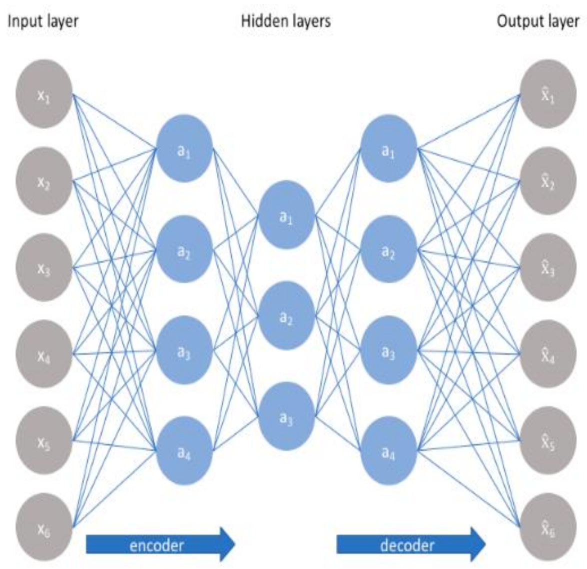

Fig. 03

(Fully connected network with encoder and decoder)

Because nonlinear relationship can be learn by neural networks this generalization can be though as more powerful generalization of PCA. A lower dimensions hyperplane is dicover by PCA which describe original data, where learning nonlinear manifolds can also be discover by the autoencoders.

The difference between these 2 approaches are shown below in Fig 04

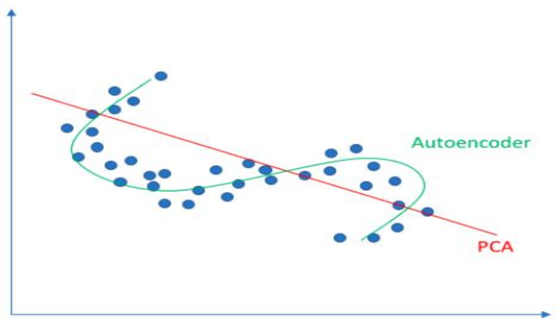

Fig. 04

(Linear vs. nonlinear dimensionality reduction)

Autoencoders are capable of learning a higer dimensional complex representation of data. Which can be used to correspondingly decode into the original input space and describe the observation in the lower dimensionality.

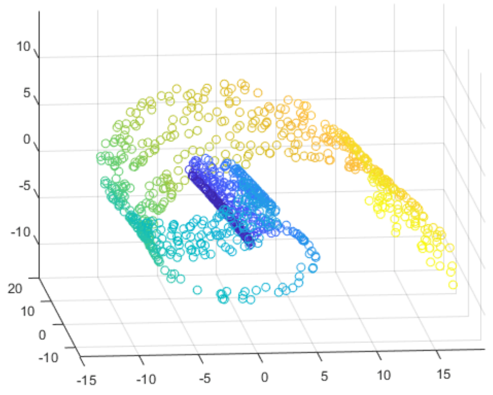

Fig. 05
(3D Nonlinear Swiss roll)

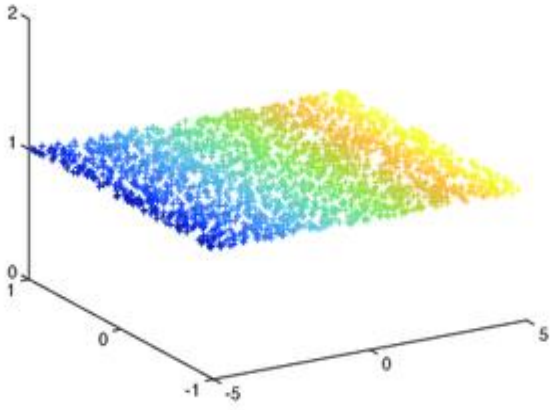

Fig. 06
(2D Linear manifold in 3Dl)

There is no explicit regularization term for and undercomplete autoencoder according to the reconstruction loss we simply train our model. To ensure that the model is not memorizing the data which have been given as an input to ensure that we have restricted the number of nodes sufficiently in the hidden layers.

III. SPARSE AUTOENCODER

In the sparse autoencoder we don't need to reduce the numbers of node in the hidden layers but sparse autoencoder provide us an alternate for the introducing an information bottleneck. We will construct our loss function such that with in layer penalizing activation functions. Activating a small number of numbers our network will learn an encoding and decoding we normally regularize the weights of network.

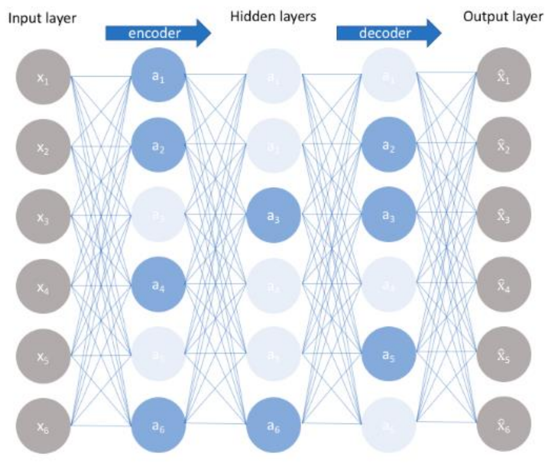

Fig. 07
(Generic Sparse autoencode)

As we can see above in Fig 07 in which a generic sparse autoencoder is visualize where transparency of a node correlate with the level of activation. The nodes of the trained model are data dependent which are activated. We allow the network to sensibilise isolate hidden layer nodes towards specific attributes of the data which is input. As we discus above the undercomplete autoencoder use the entire network for every observation and in the sparse autoencoder we force to separately activate nodes or regions of the network which depends on the input data. As a result the network capacity is decreases from memorizing the data without decreasing the capability of the network to extract features from the data. This allow us to consider separately the regularization and the latent state representation of the network. In accordance we can choose a latent state representation with what makes sense gien the context of the data while imposing regularization by scarcity constraint.
There are two main solution on which the sparsity constraint can be imposed both

solutions measuring the hidden layer activations for each and every training batch and add some values or terms to the loss function.
These solutions are give below

IV. *L1 Regularization*

To our loss function we can add a term that the absolute value of the vector is penalizes

$$\mathcal{L}\left(x, \hat{x}\right) + \lambda \sum_{i} \left|a_i^{(h)}\right|$$

V. *WHY L1 REGULARIZATION*

In ML and DL the L1 and L2 regularizations are broadly use. Absolute value of magnitude is add by L1 regularization as penalty tem and on the other hand the L2 regularization adds “squared magnitude”. Although both of the L1 and L2 regularization can be used the main difference between these two regularizations technique is that L1 shrink the penalty confident to zero and L2 will make a coefficient to move to zero but neither of them will reach to zero so L1 is use as a method of feature extraction the question arise here is that why the L1 lead us to the sparsity?.

Let us consider that we have L1 and L2 two loss function.

$$L_1 = \|w\| \, , L_2 = w^2$$

If we plot a graph of the above two loss function and also their derivatives it will look like the graph given below.

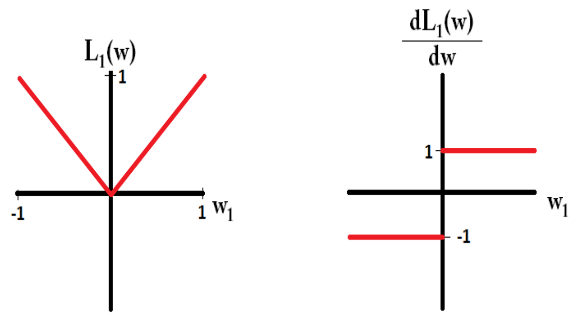

Fig. 09
(L1 regularization and its derivative)

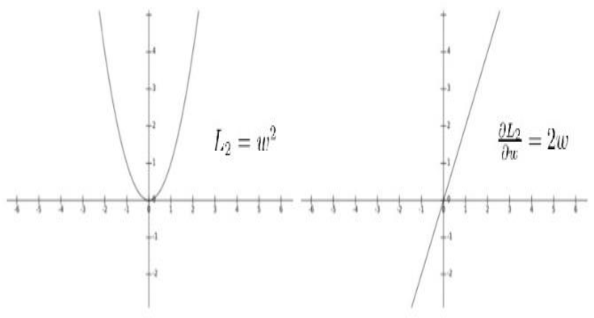

Fig. 09
(L2 regularization and its derivative)

From above graph we can notice that for L1 that gradient is 1 or -1 if the value of w is not equal to zero which means that always the L1 will mover w with same step size of (1 or -1) towards zero. And no update will be make when the gradient is zero. However if we notice that L2 also moves w towards zero and the step size will become smaller but the value of w will never reach to zero.

VI. *LOSS FUNCTON*

As we discuss above about L1 and L2 regularization we now know why we use the L1 regularization in the sparse autoencoders now the loss function with the L1 regularization is below

$$Obj = L(x, \hat{x}) + regularization + \lambda \sum_i \left|a_i^{(h)}\right|$$

In the loss function we add a new term where the absolute value of vector will be penalize. And for controlling the effect on the whole loss function we use the hyperparameter. And by adding these values and new terms we build a sparse autoencoder.

KL- Divergence

The difference between two probability distribution measures is called KL-divergence. The ρ is a sparsity parameter which denotes over a collection of samples the neuron average activation. This expectation can be calculated as

$$\hat{\rho}_j = \frac{1}{m} \sum_i \left[a_i^{(h)} (x) \right]$$

Where the specific neuron denoted by j in the layer h, m is for summing the activation and x denoted the training observation. By over a collection of samples while constructing the average activation of a neuron we are encouraging neurons to only for a subset of observation fire. The ρ can be describe as the Bernoulli random variable distribution and the KL divergence can be leverage to compare the ideal distribution ρ to the ρ hat which is the observed distribution over all the nodes in the hidden layer.

$$\mathcal{L}(x, \hat{x}) + \sum_j KL(\rho || \hat{\rho}_j)$$

The probability distribution of a random variable is called Bernoulli distribution and how this distribution work is that it take the value 1 with p and take 0 with "q=1-p". Establishing the probability a neuron will fire.

Formula for KL divergence between two distributions of Bernoulli is given below

$$\sum_{j=1}^{l^{(h)}} \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j}$$

Below the visualized graph is for an ideal distribution where p = 0.2 with a penalty of approximately equal to zero at this point

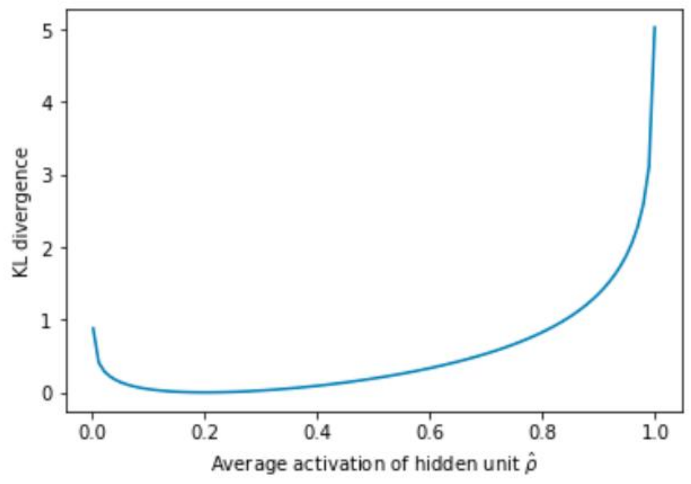

Fig 10
(KL divergence)

VII. CONCLUSION

Sparse autoencoders learn better representation because of the L1 regularization sparsity and the activations are more sparse which makes the sparse autoencoders perform better than the original autoencoders which are used without L1 regularizations.

REFERENCES

1)https://www.jeremyjordan.me/autoencoders/

2) https://medium.com/@syoya/what-happens-in-sparse-autencoder-b9a5a69da5c6

3) https://mccormickml.com/2014/05/30/deep-learning-tutorial-sparse-autoencoder/

4)http://ailab.chonbuk.ac.kr/seminar_board/pds1_files/sparseAutoencoder.pdf

5) https://towardsdatascience.com/visualising-relationships-between-loss-activation-functions-and-gradient-descent-312a3963c9a5

6) https://inteltrend.com/application-l1-l2-regularization-machine-learning/