DATA REDUNDANCY REDUCTION IN LARGE DIMENSIONAL DATASETS USING DEEP LEARNING

M. Nagavignesh

Department of Information Technology, Vels Institute of Science, Technology and Advanced Studies, India

Abstract

In this paper, the DBN pretraining procedure is not the only one that allows effective initialization of DNNs. An alternative unsupervised approach that performs equally well is to pretrain DNNs layer by layer by considering each pair of layers as a de-noising auto-encoder regularized by setting a random subset of the inputs to zero. Another alternative is to use contractive autoencoders for the same purpose by favoring models that is less sensitive to the input variations, i.e., penalizing the gradient of the activities of the hidden units with respect to the inputs. Further, a developed the Sparse Encoding Symmetric Machine (SESM), which has a very similar architecture to RBMs as building blocks of a DBN. In principle, SESM may also be used to effectively initialize the DNN training. Besides unsupervised pretraining, the supervised pretraining, or sometimes called discriminative pretraining, has also been shown to be effective and in cases where labeled training data are abundant performs better than the unsupervised pretraining techniques. The idea of the discriminative pretraining is to start from a one-hidden-layer MLP trained with the BP algorithm. Every time when we want to add a new hidden layer we replace the output layer with a randomly initialized new hidden and output layer and train the whole new MLP (or DNN) using the BP algorithm. Different from the unsupervised pretraining techniques, the discriminative pretraining technique requires labels.

Keywords:

Deep Networks, Data Reduction, Redundancy, High Dimensional Datasets

1. INTRODUCTION

Until recently, most machine learning and signal processing techniques had exploited shallow-structured architectures. These architectures typically contain at most one or two layers of nonlinear feature transformations. Examples of the shallow architectures are Gaussian mixture models (GMMs), linear or nonlinear dynamical systems, conditional random fields (CRFs), maximum entropy (MaxEnt) models, support vector machines (SVMs), logistic regression, kernel regression, multi-layer perceptrons (MLPs) with a single hidden layer, and extreme learning machines (ELMs). For instance, SVMs use a shallow linear pattern separation model with one or zero feature transformation layer when kernel trick is used or otherwise.

Notable exceptions are the recent kernel methods that have been inspired by and integrated with deep learning [1]-[3]. Shallow architectures have been shown effective in solving many simple or well-constrained problems, but their limited modeling and representational power can cause difficulties when dealing with more complicated real-world applications involving natural signals such as human speech, natural sound and language, and natural image and visual scenes.

Human information processing mechanisms (e.g., vision and audition), however, suggest the need of deep architectures for extracting complex structure and building internal representation from rich sensory inputs. For example, human speech production and perception systems are both equipped with clearly layered hierarchical structures in transforming the information from the waveform level to the linguistic level [4]. In a similar vein, human visual system is also hierarchical in nature, most in the perception side but interestingly also in the "generation" side [5]. It is natural to believe that the state-of-the-art can be advanced in processing these types of natural signals if efficient and effective deep learning algorithms can be developed.

Historically, the concept of deep learning was originated from artificial neural network research. (Hence, one may occasionally hear the discussion of "new-generation neural networks".) Feed-forward neural networks or MLPs with many hidden layers, which are often referred to as deep neural networks (DNNs), are good examples of the models with a deep architecture. Back-propagation (BP), popularized in 1980's, has been a well-known algorithm for learning the parameters of these networks. Unfortunately, back-propagation alone did not work well in practice then for learning networks with more than a small number of hidden layers (see a review and analysis in [6]).

The pervasive presence of local optima in the non-convex objective function of the deep networks is the main source of difficulties in the learning. Back-propagation is based on local gradient descent, and starts usually at some random initial points. It often gets trapped in poor local optima when the batch-mode BP algorithm is used, and the severity increases significantly as the depth of the networks increases. This difficulty is partially responsible for steering away most of the machine learning and signal processing research from neural networks to shallow models that have convex loss functions (e.g., SVMs, CRFs, and MaxEnt models), for which global optimum can be efficiently obtained at the cost of less modeling power.

The optimization difficulty associated with the deep models was empirically alleviated using three techniques: a larger number of hidden units, better learning algorithms, and better parameter initialization techniques.

Using hidden layers with many neurons in a DNN significantly improves the modeling power of the DNN and creates many closely optimal configurations. Even if parameter learning is trapped into a local optimum, the resulting DNN can still perform quite well since the chance of having a poor local optimum is lower than when a small number of neurons are used in the network. Using deep and wide neural networks, however, would cast great demand to the computational power during the training process and this is one of the reasons why it is not until recent years that researchers have started exploring both deep and wide neural networks in a serious manner.

Better learning algorithms also contributed to the success of DNNs. For example, stochastic BP algorithms are in place of the batch-mode BP algorithms for training DNNs nowadays. This is partly because the stochastic gradient descend (SGD) algorithm is

the most efficient algorithm when training is carried out on a single machine and the training set is large. But more importantly the SGD algorithm can often jump out of the local optimum due to the noisy gradients estimated from a single or a small batch of samples. Other learning algorithms such as Hessian free [7] or Krylov subspace methods [8] have shown a similar ability.

For the highly non-convex optimization problem of DNN learning, it is obvious that better parameter initialization techniques will lead to better models since optimization starts from these initial models. What is not obvious, however, is how to efficiently and effectively initialize DNN parameters until more recently [9]-[10].

The DNN parameter initialization technique that attracted the most attention is the unsupervised pretraining technique proposed in [11]. In these papers a class of deep Bayesian probabilistic generative models, called deep belief network (DBN), was introduced. To learn the parameters in the DBN, a greedy, layer-by-layer learning algorithm was developed by treating each pair of layers in the DBN as a Restricted Boltzmann Machine (RBM) (which we will discuss later). This allows for optimizing DBN parameters with computational complexity linear in the depth of the network.

It was later found out that the DBN parameters can be directly used as the initial parameters of an MLP or DNN and result in a better MLP or DNN than those randomly initialized after the supervised BP training when the training set is small. As such, DNNs learned with unsupervised DBN pre-training followed by back-propagation fine-tuning is sometimes also called DBNs in the literature. More recently, researchers have been more careful in distinguishing DNNs from DBNs, and when DBN is used to initialize the parameters of a DNN, the resulting network is called DBN-DNN.

The DBN pretraining procedure is not the only one that allows effective initialization of DNNs. An alternative unsupervised approach that performs equally well is to pretrain DNNs layer by layer by considering each pair of layers as a de-noising auto-encoder regularized by setting a random subset of the inputs to zero. Another alternative is to use contractive autoencoders for the same purpose by favoring models that is less sensitive to the input variations, i.e., penalizing the gradient of the activities of the hidden units with respect to the inputs.

Further, a developed the Sparse Encoding Symmetric Machine (SESM), which has a very similar architecture to RBMs as building blocks of a DBN. In principle, SESM may also be used to effectively initialize the DNN training. Besides unsupervised pretraining, the supervised pretraining, or sometimes called discriminative pretraining, has also been shown to be effective and in cases where labeled training data are abundant performs better than the unsupervised pretraining techniques.

The idea of the discriminative pretraining is to start from a one-hidden-layer MLP trained with the BP algorithm. Every time when we want to add a new hidden layer we replace the output layer with a randomly initialized new hidden and output layer and train the whole new MLP (or DNN) using the BP algorithm. Different from the unsupervised pretraining techniques, the discriminative pretraining technique requires labels.

2. MATERIAL AND METHODS

The Deep learning refers to a rather wide class of machine learning techniques and architectures, with the hallmark of using many layers of non-linear information processing that are hierarchical in nature. Depending on how the architectures and techniques are intended for use, e.g., synthesis/generation or recognition/classification, one can broadly categorize most of the work in this area into three classes:

- **Generative deep architectures**, which are intended to capture high-order correlation of the observed or visible data for pattern analysis or synthesis purposes, and/or characterize the joint statistical distributions of the visible data and their associated classes. In the latter case, the use of Bayes rule can turn this type of architecture into a discriminative one.
- **Discriminative deep architectures**, which are intended to directly provide discriminative power for pattern classification purposes, often by characterizing the posterior distributions of classes conditioned on the visible data; and
- **Hybrid deep architectures**, where the goal is discrimination which is assisted (often in a significant way) with the outcomes of generative architectures via better optimization or/and regularization, or where discriminative criteria are used to learn the parameters in any of the deep generative models.

Deep autoencoder is a special type of DNN whose output has the same dimension as the input, and is used for learning efficient encoding or representation of the original data at hidden layers. Note that autoencoder is a nonlinear feature extraction method without using class labels. As such the feature extracted aims at conserving information instead of performing classification tasks, although sometimes these two goals are correlated.

An autoencoder typically has an input layer which represents the original data or feature (e.g., pixels in image or spectra in speech), one or more hidden layers that represent the transformed feature, and an output layer which matches the input layer for reconstruction. When the number of hidden layers is greater than one, the autoencoder is considered to be deep. The dimension of the hidden layers can be either smaller (when the goal is feature compression) or larger (when the goal is mapping the feature to a higher-dimensional space) than the input dimension.

An auto-encoder is often trained using one of the many backpropagation variants (e.g., conjugate gradient method, steepest descent, etc.). Though often reasonably effective, there are fundamental problems when using back-propagation to train networks with many hidden layers. Once the errors get back-propagated to the first few layers, they become minuscule, and training becomes quite ineffective. Though more advanced backpropagation methods (e.g., the conjugate gradient method) help with this to some degree, it still results in very slow learning and poor solutions. As mentioned in the previous chapters this problem can be alleviated by using parameters initialized with some unsupervised pretraining technique such as the DBN pretraining algorithm. This strategy has been applied to construct a deep autoencoder to map images to short binary code for fast, content-based image retrieval, to encode documents (called

semantic hashing), and to encode spectrogram-like speech features which we review below.

In this method, features are formed using the distance-based connectivity. These hierarchal classification techniques can be categorized as top-down and bottom-up models based on how the connections are made with split and merge to form the group of objects or features in a tree form. Top-down approach is also known as divisive method in which the original features are split recursively one move down to form the hierarchy based on their similarity with a linkage method. The bottom-up approach is also known as agglomerative method. In this method, each feature initiates in its own group and the pair of groups are merged toward upward direction to form the hierarchical cluster using the similarity measure and linkage function. The linkage function can form the hierarchical cluster using the distance measures [10]. The distance measures are used to find the similarity between the features based on their distance.

3. EXPERIMENTAL SETUP

The experiments are conducted using MATLAB12b with the system configuration of Intel® Core™ 2 CPU T5300 @ 1.73 GHz processor, 4 GB memory (RAM) and 32-bit Windows vista Home Premium Operating system. The performance of the classification methods is tested on various high-dimensional datasets listed in Table.1. Further, the performance of the classification methods DD, GD and HD is tested in terms average intra-cluster redundancy rate and runtime.

The experiment is conducted with the following procedure: Initially, the dataset is given to the classification method with the number of features to be formed DL. Then the *K* numbers of features are formed. The corresponding runtime is noted and the intra-cluster redundancy rate is calculated for all *K* numbers of features for each dataset. The average intra-cluster redundancy rate is calculated by averaging intra-cluster redundancy rates from *K* numbers of features. For this experiment, the average intra-cluster redundancy rate and runtime are obtained by varying the number of features *K* from 2 to 10.

Table.1. Dataset

Dataset	Features	Instances	Classes
ORL10P[a]	10,304	100	10
PIX10P[a]	10,000	100	10
PIE10P[a]	2420	210	10
AR10P[a]	2400	130	10
SRBCT[b]	2308	83	4
ORL_32 × 32[c]	1024	400	40
Yale_64 × 64[c]	4096	165	15
COIL20[c]	1024	1440	20
DBWorld e-mails[d]	4702	64	2

Table.2. Runtime

Datasets	Runtime		
	Generative Deep	Discriminative Deep	Hybrid Deep
200	0.522	12.63	1.1
300	1.874	40.15	4.9
400	10.502	77.90	7.0
500	5.556	107.34	17.4
600	3.775	169.68	19.6
700	7.816	209.71	29.1
800	6.889	258.05	28.5
900	5.775	185.37	15.7
1000	24.168	136.45	16.5

Table.2. Average Error Rate

Datasets	Average error rate		
	Generative Deep	Discriminative Deep	Hybrid Deep
200	0.28	0.30	0.32
300	0.29	0.35	0.38
400	0.33	0.35	0.31
500	0.35	0.34	0.34
600	0.37	0.37	0.32
700	0.41	0.38	0.32
800	0.42	0.38	0.36
900	0.44	0.39	0.33
1000	0.46	0.41	0.45

Further, it is observed that the HD takes more time to form features due to the inherent computational complexity and it exhibits poor performance in terms of overall intra-cluster redundancy rate compared to GD and KC. Further, HD induces buffer overflow when the number of features is more (high-dimensional data) due to high space complexity. Therefore, HD is not a suitable choice for redundancy analysis in high-dimensional space. GD cluster has more computational complexity than DD and its overall performance in terms of intra-cluster redundancy rate is better than HC. Nevertheless, its performance is poor when compared to KC. DD classification technique performs better in redundancy analysis since it produces overall higher intra-cluster redundancy rate and takes less computational time compared to GD and HC. Therefore, it is concluded that DD classification technique can be the best choice for redundancy analysis for the high-dimension data.

4. CONCLUSION

This paper presented an empirical study on various classification techniques for redundancy analysis in feature selection for high-dimensional data classification. The performance of these classification approaches, is evaluated in terms of runtime and average intra-cluster redundancy rate. From the results, it is observed that the Hybrid deep classification is suitable for redundancy analysis in feature selection since it yields higher intra-cluster redundancy rate and takes less runtime for the high-dimensional space. Moreover, this work may be extended with different types of classification techniques and various statistical measures may be adopted for performance evaluation.

Furthermore, this redundancy analysis method can be combined with any one of the relevancy analysis methods for selecting the significant features from the high-dimensional data to achieve higher accuracy for the classification tasks for various applications.

REFERENCES

[1] S. Sowmyayani and P. Arockia Jansi Rani, "An Efficient Temporal Redundancy Transformation for Wavelet based Video Compression", International Journal of Image and Graphics, Vol. 16, No. 3, pp.1-6, 2016.

[2] S. Kallam, R. Patan and A.H. Gandomi, "Improved Salient Object Detection using Hybrid Convolution Recurrent Neural Network", Expert Systems with Applications, Vol. 166, pp. 1-23, 2020.

[3] N. Passalis and A. Tefas, "Learning Bag-of-EmbeddedWords Representations for Textual Information Retrieval", Pattern Recognition, Vol. 81, pp. 254-267, 2018.

[4] J. Zhang, C. Chen, Y. Xiang, W. Zhou and Y. Xiang, "Internet Traffic Classification by Aggregating Correlated Naive Bayes Predictions", IEEE Transactions on Information Forensics and Security, Vol. 8, No. 1, pp. 5-15, 2012.

[5] J. Kim, H. Choi and W. Lee, "Spoof Detection Method for Touchless Fingerprint Acquisition Apparatus", Korea Patent, Vol. 1, No. 54, pp. 314, 2011.

[6] S. Marcel, M.S. Nixon and S.Z. Li, "Handbook of Biometric Anti-Spoofing", Springer, 2014.

[7] G. Thimm and E Fiesler, "Neural Network Initialization", Proceedings of International Workshop on Artificial Neural Networks, pp. 535-542, 2005.

[8] M.P.S. Veenu Bhatia and P. Chandra, "Comparison of Sigmoidal FFANN Training Algorithms for Function Approximation Problems", Proceedings of International Conference on Computing for Sustainable Global Development, pp. 325-329, 2015.

[9] Simon Haykin, "Neural Networks and Learning Machines", 3rd Edition, PHI Learning Private Limited, 2011.

[10] Y. LeCun, L. Bottou, G. Orr and K. Muller, "Efficient BackProp", Springer, 1998.

[11] G.P. Drago and S. Ridella, "Statistically Controlled Activation Weight Initialization (SCAWI)", IEEE Transactions on Neural Networks, Vol. 3, No. 4, pp. 627-631, 1992.