INCREMENTAL LEARNING OF CONVOLUTIONAL NEURAL NETWORKS

Dušan Medera
Fac. of Electrical Engineering and Informatics, Technical University of Košice, 042 00 Košice, Slovakia

Štefan Babinec
Fac. of Chemical and Food Technologies, Slovak University of Technology, 812 37 Bratislava, Slovakia

Keywords: Convolutional neural networks, Incremental learning, Handwritten numbers classification.

Abstract: Convolutional neural networks provide robust feature extraction with ability to learn complex, high-dimensional non-linear mappings from collection of examples. To accommodate new, previously unseen data, without the need of retraining the whole network architecture we introduce an algorithm for incremental learning. This algorithm was inspired by AdaBoost algorithm. It utilizes ensemble of modified convolutional neural networks as classifiers by generating multiple hypotheses. Furthermore, with this algorithm we can work with the confidence score of classification, which can play crucial importance in specific real world tasks. This approach was tested on handwritten numbers classification. The classification error achieved by this approach was highly comparable with non-incremental learning.

1 INTRODUCTION

Many claims have been made about the importance of neural networks in modeling nature and artificial intelligence problem domains. To this stage, the neural networks have been applied to a variety of areas. One of the most studied areas, where neural networks were successfully applied and where they still have a potential, is pattern recognition task. The performance of neural networks as classifiers relies heavily on the availability of a representative set of the training examples. In many practical applications, data acquisition and the training process is time consuming. It is not uncommon in applications, that the data are available in small batches over a period of time. In ideal case, the classifier should support an incremental fashion of accommodating new data without compromising old data classification performance. Learning new information without forgetting previously acquired knowledge raises so-called stability-plasticity dilemma (Grosberg, 1988).

Convolutional neural networks (CNN) are representative of classifiers, which require retraining of the classifier using all data that have been accumulated so far. Why there is an effort to accommodate convolutional neural networks to incremental fashion? They have been widely adopted in pattern recognition (Y. LeCun and Haffner, 1998) and image recognition areas (Delakis and Garcia, 2003). From their fundamental principles they lack the option to accommodate new unseen data without time consuming learning process, which can play a crucial role in many applications (e.g. face recognition).

In this paper we introduce incremental learning algorithm of convolutional neural networks, which was inspired by AdaBoost and Learn++ algorithm. This algorithm allows us to accommodate new previously unseen data without the need of retraining the whole network architecture. It utilizes ensemble of modified convolutional neural networks as classifiers by generating multiple hypotheses. Furthermore, with this algorithm we can work with the confidence score of classification, which can play crucial importance in specific real world tasks. Classification results of this incremental approach are better quality as the results gained from the non-incremental approach.

2 BACKGROUND

The ability of multi-layer neural networks trained with gradient descent to learn complex, high-

Medera D. and Babinec Š. (2009).
INCREMENTAL LEARNING OF CONVOLUTIONAL NEURAL NETWORKS.
In *Proceedings of the International Joint Conference on Computational Intelligence*, pages 547-550
DOI: 10.5220/0002316405470550

dimensional, non-linear mappings from collection of examples makes them candidates for image recognition tasks. In the traditional model of pattern recognition, a hand-designed feature extractor gathers relevant information from the input and eliminates irrelevant variabilities. A trainable classifier then categorizes the resulting feature vectors into classes. In this scheme standard fully connected multi-layer networks can be used as classifiers. A potentially more interesting scheme is relating on learning in the feature extractor itself as much as possible.

CNN combines three architectural ideas to ensure some degree of shift, scale and distortion invariance: local receptive fields, shared weights and temporal sub-sampling. The CNN architecture used in our experiments is inspired by Yann LeCun (Y. LeCun and Haffner, 1998). CNN output layer consists of Euclidean radial basis function (RBF) neurons. The reason for using RBF neurons is to connect distributed codes layer with classification classes in output layer. Classification confidence score is very important in the decision making support systems. We can determine it in the following way. Neurons in the output layer can be considered as centers of individual classes of clusters defined through values of the synaptic weights. We can use the following Gaussian function for our purpose:

$$\varphi(y_i) = e^{-\frac{y_i}{\sigma}}, \quad (1)$$

where y_i is the output of ith neuron in the output layer and σ is the radius of the cluster defined through distributed codes. It is clear that values of the $\varphi(y_i)$ function are from the interval $[0,1]$ and we can consider these values as a confidence score. For the output, which is close to the distributed code of the corresponding class, confidence will be close to 1 and vice versa.

Boosting

Boosting refers to a general and provably effective method of producing a very accurate prediction rule by combining rough and moderately inaccurate rules of thumb in a manner similar to that suggested above. Boosting has its roots in a theoretical framework for studying machine learning called the "Probably Approximately Correct (PAC)" learning model due to Valiant (Valiant, 1984). Valiant was first to pose the question of whatever a "weak" learning algorithm which performs just slightly better that random guessing in the PAC model can be boosted into an accurate "strong" learning algorithm. Our inspiration was AdaBoost algorithm introduced by Freund and Schapire in 1995 (Freund and Schapire, 1999). Practically, AdaBoost has many advantages. It requires no prior knowledge about the weak learner and so can be flexibly combined with any method for finding weak hypotheses. Finally, it comes with a set of theoretical guarantees given sufficient data and a weak learner that can reliably provide only moderately accurate weak hypotheses. On the other hand, the actual performance of boosting on a particular problem is clearly dependent on the data and the weak learner. Consistent with theory, boosting can fail to perform well given insufficient data, overly complex weak hypotheses or weak hypotheses which are too weak. Boosting seems to be especially susceptible to noise. Boosting approach was also used to improve classification performance on convolutional neural networks (Y. LeCun and Haffner, 1998).

Ensemble of Classifiers

The proposed incremental learning system using Convolutional Neural Networks described in this section was inspired by the AdaBoost algorithm (Freund and Schapire, 1999) and Learn++ algorithm (R. Polikar and Udpa, 2001). Learn++ algorithm was designed for incremental learning of supervised neural networks, such as multilayer perceptrons to accommodate new data without access to previously trained data in the learning phase. Algorithm generates an ensemble of weak classifiers, each trained using a different distribution of training samples. Outputs of these classifiers are then combined using Littlestone's majority-voting scheme to obtain the final classification rule. Ensemble of classifiers can be optimized for improving classifier accuracy or for incremental learning or new data (Polikar, 2007). Combining ensemble of classifiers is geared towards achieving incremental learning besides improving the overall classification performance according to the boosting. Proposed architecture is based on this intuition: each new classifier added to the ensemble is trained using a set of examples drawn according to a distribution, which ensures that examples that are misclassified by the current ensemble have a high probability of being sampled (examples with high error rates are precisely those that are unknown).

Incremental Learning

We can describe the learning algorithm inspired by (R. Polikar and Udpa, 2001) by assuming following inputs. Denote training data $S_k = [(x_1,y_1),\ldots,(x_m,y_m)]$, where x_i are training samples and y_i are corresponding correct labels for m samples randomly selected from the database Ω_k, where $k = 1,2,\ldots,K$. Algorithm calls weak learner repeatedly to generate multiple hypotheses using different

subsets of the training data S_k. Each hypothesis learns only a portion of input space X and is weighted according to the final hypothesis. The weight of distribution on training example i on round t is denoted $D_t(i)$. Those weights are initialized by rule $D_1(i) = w_1(i) = \frac{1}{m}$, unless there is a prior knowledge to select otherwise. Learning process is iterative and in each iteration $t = 1,2,\ldots,T_k$, where T_k is number of classifiers (weak learners used in current iteration), algorithm dichotomizes S_k into a training subset TR_t and test subset TE_t according to D_t. All classifiers are called and the hypothesis $h_t : X \rightarrow Y$ is generated. The error of h_t on S_k is defined as

$$\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i), \tag{2}$$

which is simply the sum of distribution weights of misclassified examples. If $\varepsilon_t > \frac{1}{2}$, h_t is discarded and we repeat this step. Otherwise we compute normalized error β_t as

$$\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}. \tag{3}$$

All hypotheses generated in the previous t iterations are then combined using weighted majority voting scheme to obtain composite hypothesis (hypothesis performs well on own training and testing data set by giving them larger voting powers)

$$H_t = arg\max_{y\in Y} \sum_{t:h_t(x)=y} log\frac{1}{\beta_t}. \tag{4}$$

The composite error made by hypothesis H_t is computed by following equation:

$$E_t = \sum_{t:H_t(x_i\neq y_i)} D_t(i). \tag{5}$$

If $E_t > \frac{1}{2}$, current h_t is discarded and a new TR_t and TE_t is selected to obtain new h_t. E_t can only exceed this threshold during the iteration after new database Ω_{k+1} is introduced. If $E_t < \frac{1}{2}$, composite normalized error is computed as

$$B_t = \frac{E_t}{1-E_t}. \tag{6}$$

After computing B_t, the weights $w_t(i)$ are adjusted in an incremental manner of the algorithm

$$\begin{array}{ll} w_{t+1}(i) = w_t(i)B_t & \text{if} \quad H_t(x_i) = y_i, \\ w_{t+1}(i) = w_t(i) & \text{otherwise.} \end{array} \tag{7}$$

In other words, if an example x_i is correctly classified by H_t, its weight is multiplied by B_t, otherwise the weight is kept unchanged. This rule reduces the probability of correctly classified examples being chosen to TR_{t+1}. Using of composite hypothesis makes incremental learning possible particularly in cases when examples from the new class are introduced.

Finally after T_k hypotheses are generated for each Ω_k, the final hypothesis is obtained by the weighted majority voting of all composite hypotheses:

$$H_{final} = arg\max_{y\in Y} \sum_{k=1}^{K} \sum_{H_t(x)=y} log\frac{1}{B_t}. \tag{8}$$

Incremental learning is achieved through generating additional classifiers and former knowledge is not lost since all classifiers are retained.

3 EXPERIMENTS

Our goal in this paper was to compare results achieved by non-incremental learning of CNN with our new approach. We have used standard benchmarking data set MNIST in this paper. This data set represents handwritten numbers samples from different people. Every sample from this data set is represented in grayscale with 28×28 dimension and is centered in the 32×32 grid.

We have created 4 independent training sets D1, D2, D3 and D4. Every training set was composed of 2500 samples and the testing set was composed of the next 5000 samples. We have used modified Levenberg-Marquadt backpropagation of error method (LeCun, 1998) as a learning algorithm for the individual classifiers. Regarding to smaller training set, the number of learning cycles was set to 20. The initial value of the global learning parameter γ in learning algorithm equals 5.10^{-5}. In the 4th cycle the value was decreased to 2.10^{-5} and in the 12th cycle to 1.10^{-5}. The value of the parameter μ was set to 0.02. Maximum number of classifiers per data set was constrained to 5 for every training data set.

To allow comparison with non-incremental approach we have also trained one CNN, which was still retrained for all gradually presented training data sets. We have retrained it 4 times, where one retraining consisted of 40 learning cycles.

We can see the results of experiments in the folowing Tables 1 and 2. Best results were achieved with the cluster radius parameter σ set to 1. As we can see, our incremental approach achieved better results during the whole gradual training process.

4 CONCLUSIONS

Incremental learning inspired by Learn++ algorithm is based on ensemble of Convolutional Neural Network classifiers. Algorithm's update rule is optimized for incremental learning of new data. It does not

Table 1: Results of experiments for standard non-incremental learning.

Non-incremental approach		
Prediction accuracy		
	Training set [%]	Testing set [%]
D1	92.13	88.07
D1 D2	91.42	87.76
D1 D2 D3	91.99	89.36
D1 D2 D3 D4	92.12	89.63

Table 2: Results of experiments for our new approach.

Our new approach			
Number of classifiers	Prediction accuracy		
		Training set [%]	Testing set [%]
1	D1	95.52	91.95
2		98.88	92.76
3		99.44	93.02
4		99.56	92.87
5		99.56	92.76
6	D2	95.36	93.80
7		98.04	94.14
8		99.32	94.12
9		99.40	94.00
10		99.56	93.98
11	D3	98.04	94.70
12		99.60	94.90
13		99.76	94.92
14		99.84	94.86
15		99.84	94.82
16	D4	97.32	95.18
17		99.20	95.42
18		99.68	95.62
19		99.96	95.62
20		99.96	95.59

require access to previously seen data during subsequent training process and it is able to retain previously acquired knowledge. We have chosen handwritten numbers set as a testing data for our classification experiments. Our aim was to find out if this approach can give better results in comparison with standard non-incremental algorithm and in the same time offer incremental form of learning. From the results shown in this paper, it is clear that this aim has been accomplished. Classification results of our incremental algorithm are better as the results gained from the non-incremental approach. In addition, our approach brings the possibility to work with the confidence score of classification.

ACKNOWLEDGEMENTS

This work was supported by Scientific Grant Agency Vega of Slovak Republic under grants 1/4053/07, 1/0804/08 and 1/0848/08.

REFERENCES

Delakis, M. and Garcia, C. (2003). Training convolutional filters for robust face detection. In *Proc. of the IEEE international Workshop of Neural Networks for Signal Processing*, pages 739–748.

Freund, Y. and Schapire, R. (1999). A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14:771–780.

Grosberg, S. (1988). Nonlinear neural networks principles, mechanisms and architectures. *Neural Networks*, 1(1):17–61.

LeCun, Y. (1998). Efficient backprop, neural networks tricks of the trade. *Lecture Notes in Computer Science*, 1524:9–53.

Polikar, R. (2007). Bootstrap inspired techniques in computational intelligence. *IEEE Signal Processing Magazine*, 24(4):56–72.

R. Polikar, L. U. and Udpa, S. (2001). Learn++, an incremental learning algorithm for supervised neural networks. *IEEE Trans. on Systems*, 31(4):497–508.

Valiant, L. (1984). A theory of the learnable. *Communications of the ACM*, 27:1134–1142.

Y. LeCun, L. Bottou, Y. B. and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. of IEEE*, 86(11):2278–2324.