19
Practical Recommendations for Gradient-Based Training of Deep Architectures

Yoshua Bengio

Université de Montréal

Abstract. Learning algorithms related to artificial neural networks and in particular for Deep Learning may seem to involve many bells and whistles, called hyper-parameters. This chapter is meant as a practical guide with recommendations for some of the most commonly used hyper-parameters, in particular in the context of learning algorithms based on back-propagated gradient and gradient-based optimization. It also discusses how to deal with the fact that more interesting results can be obtained when allowing one to adjust many hyper-parameters. Overall, it describes elements of the practice used to successfully and efficiently train and debug large-scale and often deep multi-layer neural networks. It closes with open questions about the training difficulties observed with deeper architectures.

19.1 Introduction

Following a decade of lower activity, research in artificial neural networks was revived after a 2006 breakthrough [61, 14, 95] in the area of *Deep Learning*, based on greedy layer-wise unsupervised pre-training of each layer of features. See [7] for a review. Many of the practical recommendations that justified the previous edition of this book are still valid, and new elements were added, while some survived longer by virtue of the practical advantages they provided. The panorama presented in this chapter regards some of these surviving or novel elements of practice, focusing on learning algorithms aiming at training deep neural networks, but leaving most of the material specific to the Boltzmann machine family to another chapter [60].

Although such recommendations come out of a living practice that emerged from years of experimentation and to some extent mathematical justification, they should be challenged. They constitute a good starting point for the experimenter and user of learning algorithms but very often have not been formally validated, leaving open many questions that can be answered either by theoretical analysis or by solid comparative experimental work (ideally by both). A good indication of the need for such validation is that different researchers and research groups do not always agree on the practice of training neural networks.

G. Montavon et al. (Eds.): NN: Tricks of the Trade, 2nd edn., LNCS 7700, pp. 437–478, 2012.

Several of the recommendations presented here can be found implemented in the *Deep Learning Tutorials*[1] and in the related *Pylearn2* library[2], all based on the `Theano` library (discussed below) written in the *Python* programming language.

The 2006 Deep Learning breakthrough [61, 14, 95] centered on the use of *unsupervised representation learning* to help learning *internal representations*[3] by providing a *local training signal* at each level of a hierarchy of features[4]. Unsupervised representation learning algorithms can be applied several times to learn different layers of a deep model. Several unsupervised representation learning algorithms have been proposed since then. Those covered in this chapter (such as auto-encoder variants) retain many of the properties of artificial multi-layer neural networks, relying on the back-propagation algorithm to estimate stochastic gradients. Deep Learning algorithms such as those based on the Boltzmann machine and those based on auto-encoder or sparse coding variants often include a supervised fine-tuning stage. This supervised fine-tuning as well as the gradient descent performed with auto-encoder variants also involves the back-propagation algorithm, just as like when training deterministic feedforward or recurrent artificial neural networks. Hence this chapter also includes recommendations for training ordinary supervised deterministic neural networks or more generally, most machine learning algorithms relying on iterative gradient-based optimization of a parametrized learner with respect to an explicit training criterion.

This chapter assumes that the reader already understands the standard algorithms for training supervised multi-layer neural networks, with the loss gradient computed thanks to the back-propagation algorithm [103]. It starts by explaining basic concepts behind Deep Learning and the greedy layer-wise pretraining strategy (Section 19.1.1), and recent unsupervised pre-training algorithms (denoising and contractive auto-encoders) that are closely related in the way they are trained to standard multi-layer neural networks (Section 19.1.2). It then reviews in Section 19.2 basic concepts in iterative gradient-based optimization and in particular the stochastic gradient method, gradient computation with a flow graph, automatic differentation. The main section of this chapter is Section 19.3, which explains hyper-parameters in general, their optimization, and specifically covers the main hyper-parameters of neural networks. Section 19.4 briefly describes simple ideas and methods to debug and visualize neural networks, while Section 19.5 covers parallelism, sparse high-dimensional inputs, symbolic inputs

[1] `http://deeplearning.net/tutorial/`

[2] `http://deeplearning.net/software/pylearn2`

[3] A neural network computes a sequence of data transformations, each step encoding the raw input into an intermediate or internal representation, in principle to make the prediction or modeling task of interest easier.

[4] In standard multi-layer neural networks trained using back-propagated gradients, the only signal that drives parameter updates is provided at the output of the network (and then propagated backwards). Some unsupervised learning algorithms provide a local source of guidance for the parameter update in each layer, based only on the inputs and outputs of that layer.