ISSN 2281-4299

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA UNIVERSITÀ DI ROMA

Global Optimization issues in Supervised Learning. An overview

Laura Palagi

Technical Report n. 11, 2017

Global Optimization issues in Supervised Learning

An overview

Laura Palagi

July 2017

Abstract The paper presents an overview of global issues in optimization methods for Supervised Learning (SL). We focus on Feedforward Neural Networks with the aim of reviewing global methods specifically devised for the class of continuous unconstrained optimization problems arising both in Multi Layer Perceptron/Deep Networks and in Radial Basis Networks. We first recall the learning optimization paradigm for FNN and we briefly discuss global scheme for the joined choice of the network topologies and of the network parameters. The main part of the paper focus on the core subproblem which is the unconstrained regularized weight optimization problem. We review some recent results on the existence of local-non global solutions of the unconstrained nonlinear problem and the role of determining a global solution in a Machine Learning paradigm. Local algorithms that are widespread used to solve the continuous unconstrained problems are addressed with focus on possible improvements to exploit the global properties. Hybrid global methods specifically devised for SL optimization problems which embed local algorithms are discussed at the end.

Keywords Supervised Learning · Feedforward Neural Networks · Global Optimization · Weights Optimization · Hybrid algorithms

Dip. di Ingegneria informatica automatica e gestionale A. Ruberti Sapienza - University of Rome
Via Ariosto 25 - 00185 Roma
Tel.: +39-06-77274081
E-mail: laura.palagi@uniroma1.it

Contents

1 Introduction

Mathematical optimization plays a pillar role in Machine Learning (ML). Indeed learning from available data means that parameters of a chosen system must be computed by solving to optimality a learning problem. Depending on the type of learning problem we get different classes of optimization problems. In this paper we aim to review the role of Global Optimization (GO) within the field of Supervised Learning (SL). Hence Unsupervised Learning (UL), such as clustering, is not reviewed and the interested reader can check e.g. [6,120]. We will mainly focus on continuous optimization formulations of SL problems. Actually, continuous optimization methods have been widely used in statistics and have had a significant impact in the last 30 years, that has led to an enormous amount of papers on the topic (see e.g. the recent survey [100]). Only recently some integer optimization formulations have been proposed [16] exploiting the significant advances in integer optimization that make it possible to solve large scale problems, such those arising in ML, within practical times.

SL paradigm consists in determining a prediction function for labeling unseen data using a set of labeled training data. In the context of SL we can distinguish two main classical approaches which are Feedforward Neural Networks (FNN) and Support Vector Machines (SVM) [19]. FNN and SVM represent a different way of exploiting the tradeoff between complexity and generalization performance of the machine according to the Vapnik-Chervonenkis (VC) theory [124], as it will be briefly reviewed in Section 2.

FNN training problem turns out to be an unconstrained non convex nonlinear optimization problem. On the other hand, training a SVM reduces to the solution of a convex problem that does not pose any difficulties from the global optimization point of view and hence it is not addressed in this review. Actually recent developments on Semi-Supervised SVM lead to a non-convex formulation that can be solved exactly by a combinatorial approach or approximately using a continuous relaxation. We refer to [37] for a survey on semi-supervised methods which are not covered in this paper and to [32] and references therein for a complete survey on mathematical optimization for SVM.

In this paper we mainly focus on the class of continuous optimization problem arising in FeedForward Neural Network (FNN).

A good learning process requires both the optimal selection of the appropriate network, which includes number of neurons and corresponding activation functions together with other hyper-parameters, and the choice of the parameters of the network (weights). From the optimization point of view the arising optimization problem is very difficult in the fact that it is nonlinear, non convex and in some cases also not differentiable. The relationships among parameters defining the learning machine and its ability to predict the output on future unknown data may be not explicitly known so that the problem is in effect a black-box optimization problem. Indeed, as explained in Section 2, the definitive objective in SL learning is not well defined and often different objectives that measure conflicting performance drive the final choice of the network.

However the core of FNN training process is the solution of an unconstrained nonconvex problem. Due to the multimodality of the objective function, standard unconstrained local optimization algorithms, such as gradient based methods, Newton type methods etc., may fail in finding the optimal value of the parameters. Depending on the structure analyzed this may cause more or less severe problems. We discuss in the paper the main issues arising in improving the local solutions.

The paper is organized as follows. In Section 2 we report some basics about the statement of SL problem; in Section 3 we enter more in details of the FNN architectures, in particular Multi Layer Perceptron/Deep Networks (MLP/DN) and in Radial Basis (RBF) Networks. In Section 4 we discuss the general paradigm of the global problem addressed in FNN. In Section 5 we introduce the core subproblem in the global optimization scheme which is the Weights Optimization (WO) problem. In Section 5.1 we discuss cases when a global solution, or a good approximation, is easy to be found so that applying a global procedure may be not worthwhile. In Section 5.2 we review local methods with emphasis on the possible hidden global properties; in particular we refer to sample-wise decomposition methods in Section 5.2.1 and block coordinate-wise decomposition in Section 5.2.2. In Section 5.3 a review of hybrid global scheme specifically designed for FNN are finally reported.

Abbreviations used throughout the paper can be found at the end of the paper in Table 6.

Notation. Throughout the paper, all vectors are considered column vectors. If not differently specified the vector/matrix norm is Euclidean, denoted by $\|\cdot\|$. Given a $n \times m$ matrix A, we denote by a_{ij} the elements and by $A_j \in \mathbf{R}^n$ the $j-$th column. An apex ^{T} denotes transposition of a vector/matrix. Given a vector $v \in \mathbf{R}^q$ and a subset $I \subset \{1,\dots,q\}$, we denote by v_I the subvector of dimension $R^{|I|}$ with component v_i for $i \in I$.

Table 1 Notation for data

P	number of training samples
$x^p \in \mathbf{R}^n$	p-th input sample
$X \in \mathbf{R}^n \times \mathbf{R}^P$	$n \times P$ input training matrix with columns $x^p \in \mathbf{R}^n$, $p = 1, \dots P$
$y^p \in \mathbf{R}^m$	p-th output sample
$Y \in \mathbf{R}^m \times \mathbf{R}^P$	$m \times P$ output training matrix with columns $y^p \in \mathbf{R}^m$, $p = 1, \dots P$
$\mathcal{P}$	unknown probability distribution
$\mathcal{T}$	training set $\{(x^p, y^p),\ p = 1, \dots, P\}$

2 Statement of the Supervised Learning problem

In this section we briefly recall standard definitions in Supervised Learning. In particular we introduce the definition of Expected Risk and Empirical Risk and the relationship among the two measures and the key factors impacting on these functions.

In SL we are given a training set $\mathcal{T}$ made up of input-output pairs (x^p, y^p) for $p = 1, \dots, P$, where $x^p \in \mathcal{X} \subseteq \mathbf{R}^n$ and y^p is a m dimensional vector, m being the number of outputs, with components that can assume values in $\mathbf{R}$ (regression pb. $y \in \mathbf{R}^m$) or in a discrete set (classification pb, e.g. $y \in \{-1, 1\}^m$ for two class problems). It is assumed that input-output pairs (x, y) are sampled by an unknown probability distribution $\mathcal{P}(x, y)$. IN the rest of the paper we consider the regression case, namely $y \in \mathbf{R}^m$. For reader's convenience we report the data notation in Table 1.

The goal of a learning machine is to determine a function $f(\cdot, \alpha)$, parametrized in α within a given class of functions $\mathcal{F}$ such that for a given input $x \in \mathbf{R}^n$ the output $f(x; \alpha)$ is an accurate prediction of the true unknown output. Both the choice of the parametrized class of function $\mathcal{F}$ and the selection of a specific prediction function $f(\cdot; \alpha^*)$ in the class, chosen by fixing the parameter to the value α^* are the two main problems addressed by a learning procedure.

In order to define a criterion for the choice of $f(\cdot; \alpha^*)$ in $\mathcal{F}$, a measure of the "goodness" of a learning machine must be defined and a loss function $\mathcal{L}(v, y) \geq 0$ is considered which measures the difference between the value returned by the selected machine $v = f(x; \alpha^*)$ and the true output value y when the input is x. By definition, the loss is nonnegative and high positive values indicates bad performance. The "quality criterion" that drives the choice of the function f and of the parameter α is, in principle, the minimization of the *expected risk* R that is the expected value of the error with respect to the given measure $\mathcal{L}$

$$R(\alpha) = \mathbf{E}\{E(x, y; \alpha)\} = \int \mathcal{L}(y, f(x, \alpha)) d\mathcal{P}(x, y) \tag{1}$$

where E is the composition of the loss function $\mathcal{L}$ and the prediction function f. However $\mathcal{P}(x, y)$ is unknown so that the problem $\min_\alpha\ R(\alpha)$ is imponderable.

Actually the only information available for the training process are the data (x^p, y^p) for $p = 1, \dots, P$ which can be seen as a set of realizations (samples) from the unknown distribution $\mathcal{P}(x, y)$. For a given value of α, the loss incurred with respect to the i-th sample is $E_i(\alpha) = \mathcal{L}(y^i, f(x^i, \alpha))$. Hence a valuable measure of the error that does not depend anymore on $\mathcal{P}$ is the *empirical risk* R_{emp} defined as

$$R_{emp}(\alpha) = \frac{1}{P} \sum_{i=1}^{P} E_i(\alpha) \tag{2}$$

The relationships between R and R_{emp} have been subject of the Vapnik-Chervonenkis (VC) theory [124]. They proved a fundamental result which states that with probability $1 - \eta$, being $\eta > 0$, the difference between R and R_{emp} is bounded above

$$\sup_{f \in \mathcal{F}} |R(\alpha) - R_{emp}(\alpha)| \leq \mathcal{O}\left(c(h, P, \eta)\right) \tag{3}$$

where the term $c(h, P, \eta)$ is called VC-*confidence*. We do not enter into details of the expression of VC-confidence and we refer e.g. to [30] and to the recent survey [22] for more details. What really matters is that the VC-confidence depends on the number of training data P and on the non negative integer h called (VC)-*dimension* which represents a measure of the classification ability of the learning machine represented by functions in the class $\mathcal{F}$. The VC-confidence is a monotonic increasing function of h and for fixed P the gap can be widen by larger h (higher complexity). Further the VC-dimension h may depend on the dimension of the input x, e.g. for the class of linear functions $h = n + 1$. Hence reducing the dimension n, namely reducing the number of features of the training data, may indirectly decreases h and in turn limits the VC-confidence term. On the other hand reducing n may lead to an increase of the empirical risk, and hence *feature selection* (FS) is a hot topic in SL. Recently some Global Optimization tools have been proposed to tackle the FS problem that are discussed in Section 4.

As a result of (3), the two learning tasks, i.e. choosing the class $\mathcal{F}$ and choosing the parameters α^* to identify a particular function $f(\cdot, \alpha^*)$, can be performed by looking for the best value of the upper bound $R_{emp}(\alpha) + c(h, P, \eta)$, which is the basis of the Structural Risk Minimization (SRM) principle. Following the SRM performs a tradeoff between decreasing the generalization error and increasing the empirical risk.

Two main extreme approaches can be derived from SRM principle which are FNN and SVM. Roughly speaking, FNNs assume a prefixed level of the VC-confidence, by defining a specific architecture of the network, and perform an unconstrained minimization of the empirical risk R_{emp} with respect to the free parameters. The corresponding problem is in general nonlinear and non convex and thus poses serious difficulties from the global optimization point of view.

On the other hand, SVMs pose constraints on the value of the allowed empirical risk and minimize the VC-confidence term by acting on the value h. SVM training problem turns to be a constrained and convex optimization problem. In this sense SVM training problem does not present global difficulties in its solution. Actually global issues are hidden in the SVM training process due to the need of choosing some hyper-parameters that play a fundamental role in the generalization performance of the SVM network. However in the SVM case, the choice of the hyper-parameters is almost always tackled in a heuristic way by a trial-and-error procedure (as explained in Section 4) before performing the optimization of the network parameters. Recently a Mixed integer Linear Programming problem for the joined choice of both parameters and hyper-parameters has been proposed in [53] with the aim of reducing the trial-and-error outer-loop on hyper-parameters.

Although the problem is NP-hard, it has been shown in [53] that it can be solved heuristically in a satisfactory way.

Recently in [24] the bound derived by the VC theory has been modified to tackle the large scale setting when the number of parameters is large, as it happens in Deep Networks, and the computing time represents the demanding resource in solving min R_{emp}. Indeed in large scale setting, computational time may be a severe restriction that does not allow the optimization procedure to find an accurate solution. Hence in [24] an *optimization error* E_{opt} is introduced, which reflects the fact that algorithms return an approximate solution within a given tolerance. The authors showed that the generalization properties of large-scale learning systems depend both on the statistical properties of the estimation procedure and on the computational properties of the optimization algorithm, so that the error E_{opt} cannot be neglected and the choice of the optimization procedure impacts on the performance of the training procedure. In contrast, in small-scale learning systems the optimization error E_{opt} can be reduced to insignificant levels. This may explain different performance of the same algorithm when applied to networks with large or small number of parameters.

The overall training procedure of a FNN is a hard task and it is not well posed in the fact that we need to manage a tradeoff among the generalization capability of the network measured by R given by the imponderable expression (1) and the available measure of the error which is R_{emp} [57]. The lonely use of R_{emp} as a measure of performance may lead to overfitting phenomena that cause ba beahviour of the network when used in the predictive phase. Hence, although the pursued goal in FNN for the choice of the parameters is the minimization of R_{emp}, the definitive measure of performance of the network is measured by a different function rather than the one used in the optimization process. These two conflicting aspects are often taken into account implicitly in the definition of the optimization procedure for minimizing (2) and this may often cause confusion on the definitive measure of performance of the solution as discussed in Section 4.

For preventing overtraining [98,18], a regularization term $\lambda\Omega(\alpha)$ where $\lambda \geq 0$ is often added to the error function R_{emp} given by (2).

3 FNN topologies: MLP/DN and RBF networks

In this section we consider the two main different architectures of FNN, namely we characterize the class of functions $\mathcal{F}$ and the corresponding parameters α.

FNN consists of several processing units (neurons) arranged in layers connected in a feed-forward way with weights. A FNN with n inputs and m outputs is an acyclic oriented network as schematically illustrated in Figure 1.

Each connection arch among neurons i, j in two successive layers is weighted by a scalar w_{ji} and each neuron j in layer ℓ is characterized by an activation function g_j^ℓ which usually depends on some of the parameters under choice. The choice of the activation functions g_j^ℓ leads to different classes of FNN. We analyse methods for *Multilayer Perceptron* (MLP) or *Deep Networks* (DN) [77,108] and *Radial Basis Function* (RBF) networks [98]. We use the name MLP to denote a *shallow* FFN, namely the case when there is only one hidden layer ($L = 2$), and we use *Deep Network* (DN) when the hidden layers are more than one. RBF networks are usually shallow networks that present some specific features of the training problem that can be exploited in the definition of algorithms converging to a solution. Although networks where RBF units coexist with MLP units [44] or/and RBF having deep structure [36] have been proposed, they have not been used in practice.

In order to express the output of a FNN network, we introduce the following notation summarized in Table 2. We denote by L the number of layers, and by N^ℓ the number of units in layer $\ell = 0, \dots, L$ being $\ell = 0$ the input layer with $N^0 = n$ and $\ell = L$ the output one with $N^L = m$; W^ℓ is the $N^\ell \times N^{\ell-1}$ matrix of weights from layer $\ell - 1$ to layer ℓ; $z^\ell \in \mathbf{R}^{N^\ell}$ is the output vector of the neurons of layer ℓ, being $z^0 = x$ and $z^L = y$. The activation function of neuron j at layer ℓ is denoted by g_j^ℓ and can be parametrized by additional (hyper)-parameters α_j^ℓ so that $z^\ell = g^\ell(W^\ell, \alpha^\ell; z^{\ell-1})$ for $\ell = 1, \dots, L$. We have not explicitly stated the presence of a bias at neuron j that w.l.g. can be included within the weights vector by adding a fictitious input. For sake of simplicity and w.l.g. from now on we will refer to a network with a linear unit in the output layer $g^L = W^L z^{L-1}$.

With this assumption and using the above notation, the output of a FNN is expressed as

$$y(\alpha; x) = W^L g^{L-1}\left(W^{L-1}, \alpha^{L-1}; g^{L-2}(\dots; g^1(W^1, \alpha^1; x))) \dots\right) \tag{4}$$

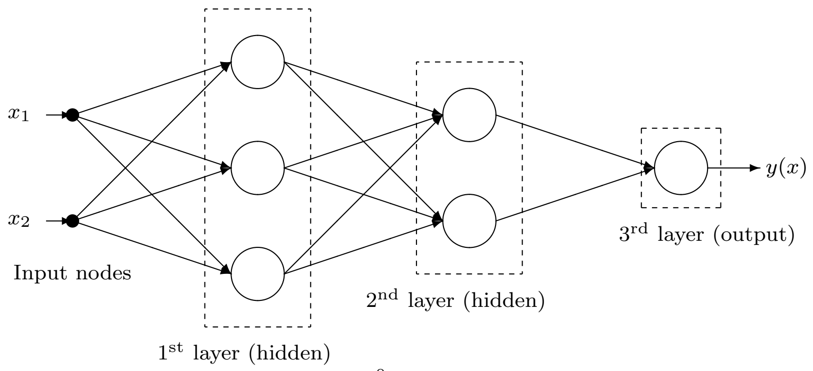

Fig. 1 Deep Network with two inputs $N^0 = n = 2$, two hidden layers $L = 3$ and a single output $N^3 = m = 1$

To complete the description we need to specify the type of activation function g_j^ℓ that may be different for every neuron j at layer ℓ of the network. Depending on g we distinguish among MLP and RBF network.

In MLP/DN networks the activation function acts as an on-off trigger on a weighted combination of the outputs of the neurons in the preceding layer; the activation function is in most cases a sigmoidal function[1] applied on scalar products. Usually $g_j^\ell = g$ for all j and $\ell = 1, \dots, L-1$. The activation function g is usually parameterized by the hyper-parameter $\sigma \in \mathbf{R}$ that define the slope of the function so that $z^\ell = g(W^\ell, \sigma; z^{\ell-1}) = g(W^\ell z^{\ell-1}, \sigma)$. Examples of some of the most used

[1] A sigmoid function σ is a monotonically increasing function that asymptotes to some finite value as $\pm\infty$ is approaching.

Table 2 Notation for FNN

L	number of layers
N^ℓ	number of units in the layer ℓ
$n = N^0$	number of inputs units in the zero layer
$m = N^L$	number of outputs units in the zero layer
$W^\ell \in \mathbf{R}^{N^{\ell-1}\times N^\ell}$	matrix of weights from layer $\ell - 1$ to layer ℓ
W	vector of all weights
z^ℓ, $\ell = 0, \dots, L$	output of neurons of layer ℓ
$z^0 = x$ and $z^L = y$	output of first and last layers respectively
g^ℓ	activation function vector of layer ℓ
$g^L = W^L z^{L-1}$	linear activation function of the output layer

MLP/DN activation functions are in Table 3. For a shallow FNN ($L = 2$) with linear output neuron we get the output

$$y(\alpha; x) = W^2 g\left(W^1 x; \sigma\right) \tag{5}$$

Thus in training MLP/DN we need to choose the architecture, namely L and N^ℓ for all $\ell = 1, \dots, L-1$, the activation function g and its hyper-parameter σ and the parameters $W = \{W^1, \dots, W^L\}$. These choices are part of the training procedure that aims at minimizing the function (2).

In RBF networks the activation function of the hidden layer g_j is a radial basis function which depends on the distance r between the input $z^{\ell-1}$ and some vectors called *centers* $c^j \in \mathbf{R}^n$, $j = 1, \dots, N^\ell$. The RBF can be also parametrized by $\sigma_j > 0$ (*radius*). Thus the outputs of units at layer ℓ is the vector $z^\ell = g(W^\ell, C, \sigma; z^{\ell-1})$ where $C = \{c^1 \dots, c^N\}$. Examples of most used RBFs are reported in Table 4.

RBF networks are usually shallow structure ($L = 2$) in which the input-to-hidden weights W^1 are all set to one so that the output of a shallow RBF network can be written as

$$y(\alpha; x) = W^2 g\left(C, \sigma; x\right) \tag{6}$$

where g is the N dimensional vector defined componentwise by $g_j(\|x - c^j\|; \sigma_j)$. Usually a uniform function $g_j(\|x - c^j\|; \sigma_j) = g(\|x - c^j\|; \sigma)$ is used for all j.

The number of hidden units N, the position of the centers C and the radius σ is not known and must be chosen by the learning procedure. Usually $N << P^2$.

For shortness in the following we may refer to the centers as weights at the hidden node. The different role played by the weights on the arcs and by the centers can be exploited in the optimization procedure.

The choice of the transfer function g plays an important role in the fact that forms decision boundaries of different shapes and it may have a strong impact on the complexity and performance of the neural model. We refer to [3, 50, 49] for surveys on the role of transfer functions in the training process. In the following we assume that in the training problem of minimizing (2) the transfer function g

[2] For $N = P$ the centers are set to $c^i = x^i$, $i = 1, \dots, P$ (i.e. $C = X$). The corresponding network is called *regularized RBF network* and the corresponding training problem (2) for the computation of the weights W reduces to the convex linear least square problem (11). Over fitting phenomena can frequently occur with this choice of N, leading to a bad performance in terms of generalization ability.

Table 3 Activation function $g(z)$

z	Linear
$\max\{z, 0\}$	ReLU: Rectified Linear Unit
$\dfrac{1}{1+e^{-\sigma z}}$	Logistic with $\sigma \in \mathbf{R}_+$
$\tanh(\sigma z) = \dfrac{1-e^{-2\sigma z}}{1+e^{-2\sigma z}}$	Hyperbolic tangent with $\sigma \in \mathbf{R}_+$

Table 4 Radial basis functions $g(r)$ with $r = \|x - c\|$

$e^{-(r/\sigma)^2}$	Gaussian
$(r^2 + \sigma^2)^{1/2}$	multiquadric
$(r^2 + \sigma^2)^{-1/2}$	inverse multiquadric
r	linear spline
r^3	cubic spline
$r^2 \log r$	thin plate spline.

is given and it is not subject to the optimization process. The hyper-parameter σ that enters its definition may instead be chosen by an optimization procedure.

4 The optimization paradigm in FNN training

In this section the optimization paradigm underlying FFN training problem is reviewed. In particular

– we define the unconstrained optimization problem,
– we discuss the two phase training paradigm for the choice of the network parameters (hyper-parameters and weights),
– we report the main approaches for the choice of the hyper-parameters in connection with GO procedures,
– we discuss the role of feature selection within this paradigm.

The unconstrained training problem. We consider the unconstrained minimization problem of the empirical error when $y(\alpha; x)$ is the output of a FNN network as given by (4).

In particular we focus on two phase approaches that splits the solution into

– the choice of hyper-parameters tied to the topology of the network
– the choice of the weights on arcs and/or on units of the network.

In order to complete the definition of R_{emp}, the loss function $\mathcal{L}$ needs to be defined. Examples of well known loss functions are reported in Table 5. A standard and validated choice in FNN when $y \in \mathbf{R}^m$ is the squared loss, that gives the Mean Squared Error (MSE) expression of R_{emp}:

$$R_{emp}(\alpha) = \frac{1}{P}\sum_{i=1}^{P} \|y^i - f(x^i, \alpha)\|^2.$$

Table 5 Loss function $\mathcal{L}(v,y)$

$\|v-y\|^2$	L_2 norm Mean Square loss
$\|v-y\|_p^p$	L_p norm
$\sigma(z)=\frac{1}{1+e^{-\sigma z}}$	sigmoidal function
$-\sigma(v)\log\sigma(y)-(1-\sigma(v))\log(1-\sigma(y))$	cross entropy where $\sigma(\cdot)$ is a sigmoidal function
$\log(1+\exp\{-v^Ty\})$	log loss
$\max\{0,1-v^Ty\}$	hinge loss

In this cases, the regularization term takes often the form $\lambda\,\|\omega\|^2$ and it can be viewed as a form of scalarization of the two conflicting objectives described by the bound (3) undergoing the learning procedure [75]. Of course the choice of the correct value of the parameter λ is crucial and indeed λ is often included among the hyper-parameters π to be chosen during the learning process. We split parameters α into the hyper-parameters $\pi\in\mathit{\Pi}$ and the parameters $\omega\in\mathbf{R}^q$ (weights), so that $\alpha=(\pi,\ \omega)$. The reason for this distinction will be clearer in the following.

Thus the unconstrained minimization problem of the regularized empirical error for a FNN is given by

$$\min_{\omega,\pi} E(\omega;\pi)=\frac{1}{P}\sum_{p=1}^{P}\|y(\omega,\pi;x^p)-y^p\|^2+\lambda\|\omega\|^2 \tag{7}$$

where ω represents the weights on arcs and/or neurons (for MLP/DN $\omega=W$ whereas for RBF $\omega=(W,\ C)$). The hyper-parameters π depend on the architectural choices on the network and in particular on the parameter $\sigma\in\mathbf{R}$ appearing in the activation function, the number of layers L, the number of neurons N^ℓ and the regularization parameter λ.

Two phase optimization paradigm. In principle we look for a global solution (ω^*,π^*) of problem (7) which is a setting of the parameters such that

$$E(\omega^*;\pi^*)\le E(\omega;\pi)\quad \forall\ \omega\in\mathbf{R}^q \text{ and all possible settings } \pi\in\mathit{\Pi}$$

We observe however that the two blocks of parameters ω,π play different roles in the definition of the network. In particular, the choice of the hyper-parameters π is mostly tied to the topology of the network and to the generalization ability. On the other hand the choice of ω corresponds to the selection of a specific network with the given architecture. Hence in most training schemes the choices of ω and π are addressed separately and a two-phase procedure is implemented which consists in choosing respectively either ω or π. A possible two-phase learning scheme is reported below.

```
Two-phase training procedure
Given a training set $\mathcal{T}=\{(x^p,y^p):\ x^p\in\mathbf{R}^n;\ y^p\in\mathbf{R}\}_{p=1,\dots,P}$

 Repeat
    1. [Hyper-parameter selection]
       Choose the hyper-parameters $\pi$;
    2. [Weights selection]
       Choose the parameters $\omega$;
 Until a stopping criterion is satisfied
```

In principle both the two phases can be solved by means of an optimization process. If this is the case, the scheme above can be viewed as a two-block decomposition procedure [14] in which the alternate optimization w.r.t. π and ω is performed being the other parameter fixed. In this framework, starting from an initial guess π^0, ω^0, the scheme generates an iterate π^t, ω^t, $t = 1, \dots$ alternating the solution of the subproblems

$$\pi^{t+1} = \arg\min_{\pi} E(\pi, \omega^t)$$

and

$$\omega^{t+1} = \arg\min_{\omega} E(\pi^{t+1}, \omega)$$

until a satisfactory solution is found.

Thus the solution of problem (7) is split into the solution of smaller problems. However both the two subproblems correspond in general to hard non convex problem that cannot be solved to global optimality. But even in the case that a global solution could be found in each of the two phases, the two block procedure is not guaranteed to reach the global solution $(\omega^*; \pi^*)$ of problem (7) (see section 5.2.2 for convergence results on block decomposition methods).

Further we also have to consider that E depends differently on ω and π. Indeed, when using continuously differentiable activation functions g, $E(\omega, \pi)$ is continuously differentiable in ω. However the dependance on some architectural parameters, such as the value L, N^ℓ is discontinuous and the behaviour of E as a function of the hyper-parameters π is not well understood and can be regarded in some sense as a black-box optimization problem which is very difficult to solve (see e.g. [94]). Hence, the hyper-parameters problem is often solved heuristically by pursuing the definition of satisfactory solution rather than the exact solution of (7). As we already discussed in preceding sections, the definition of "satisfactory solution" may not coincide neither with the global optimal solution (ω^*, π^*) of problem (7) nor with its approximation. Indeed the definitive choice of the parameters $(\widehat{\omega}, \widehat{\pi})$ is often performed by looking also to the hidden task of minimizing the risk (1), namely on the basis of the generalization capability of the network. Although the regularization term is added to the empirical risk to improve generalization, other tricks such as the use of different measures of performance of the network, e.g. the *validation error* R_{val}, are used. The validation error is the error measured on a set of data, called *validation set* $\mathcal{V}$, not used to construct the training model (7). The MSE on validation set is:

$$R_{val} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \|y(\omega, \pi; x^i) - y^i\|^2.$$

The validation error[3] mimics the expected error in the sense that offers a measure of the error on samples unavailable to the optimization process. It is usually considered a more advisable measure of the generalization capability of the network and hence of the goodness of the overall training process rather than the value of R_{emp}. In this sense the learning problem is inherently a multiobjective optimization problem as explained e.g. in [75] which is addressed using a single objective or multiple objectives tackled in separated phases.

[3] Usually the data set is divided into K subsets and the average validation error across all K trials is computed (K-fold cross validation).

Hyper-parameters optimization. In a two-phase scheme, hyper-parameters are often chosen according to the value of the validation error by either an exhaustive search over a finite subset of parameter values Π using a grid or manual search (trial-and-error) or by an optimization procedure or by a combination of these two methods.

The use of a grid or manual search is particularly suited for all those hyper-parameters which naturally assume discrete values, such as the values L, N^ℓ. Hence frequently a trial-and-error approach is used for these parameters; the network architecture is fixed and a training process is started. Neurons and/or connections are removed or inserted and the network trained further. This process is iterated until a satisfactory solution is found. We note that including the hyper-parameters L, N^ℓ in the optimization process underlies the assumption that the quality of the network, measured by either the training or the validation error, is a smooth function of the topology of the network. More recently pruning methods which eliminate nodes during the training algorithm that chooses the weights have been proposed as e.g. the `DROPOUT` algorithm for DN in [9,116]. Indeed in DN the over fitting phenomena is important due to the large number of parameters involved so that reducing dimension is a key factor of success. The key idea of the `DROPOUT` algorithm is to randomly drop units (along with their connections) from the neural network during training giving raise to an exponentially large number of thinned network. The procedure prevents any neuron from relying excessively on the activation of other individual units, forcing it instead to rely on the population behavior of its inputs. Actually the dropout procedure can be applied also to the input layer and turn out to be a method for feature selection. In [9] a general mathematical framework that enables the understanding of several aspects of dropout has been developed. In particular it has been shown that `DROPOUT` possess inherently ensemble averaging properties, as well as its regularization properties.

For other hyper-parameters problem it has been proposed since the eighties the use of meta-heuristics. In particular standard global optimization (GO) methods [85] can be applied both deterministic [54,71] and stochastic [131,69] ones. Good surveys on GO methods for solving the SL problems are [51] and the more recent [90]. However it is worth to mention that most of local and global algorithms for solving the SL problem may require the choice of algorithmic parameters too. Often these parameters enter the learning procedure itself [90,1]. This would leave unsolved the question whether the learning model performs poorly due to the appropriateness of the given architectural choices or rather to the inability of the learning method to reach a good solution of (7).

However grid search and manual search, that consists in an exhaustive search over a finite subset Π of the possible values taken by the parameters, are the most widely used strategies for hyper-parameter optimization [78]. Clearly they are not very efficient, but are deterministic and reliable. When using an exhaustive search a subset of parameter values $\Pi = \{\pi^1, \dots, \pi^T\}$ is chosen. For each fixed setting of the hyper-parameters $\pi^t \in \Pi$, problem (7) is solved with respect to the only weights ω obtaining a solution ω^t (which depends on π^t). If a global minimizer can be found, ω^t satisfies

$$E(\omega^t; \pi^t) \le E(\omega; \pi^t) \qquad \forall\, \omega \in \mathbf{R}^q$$

Finally, the pair of parameters

$$(\widehat{\omega};\widehat{\pi}) = \arg\min_{\pi^t \in \Pi} R_{val}(\omega^t;\pi^t)$$

is chosen. The solution $(\widehat{\omega};\widehat{\pi})$ can be far from the true global one $(\omega^*;\pi^*)$ of problem (7) both depending on how fine the grid is, namely how large is the set Π, on the algorithm for solving the WO problem and on how reliable is the training set. Of course the larger the set Π the more the computational effort. This grid search procedure is outline below.

```
Grid-Search Two-phase training procedure
Given a training set $\mathcal{T} = \{(x^p, y^p) : x^p \in \mathbf{R}^n;\ y^p \in \mathbf{R}\ p = 1, \dots, P\}$ .
[Parameters selection]
Let $\Pi = \{\pi^1, \dots, \pi^T\}$.
 For $t = 1 \dots, T$
   [Weights Optimization]
   Find $\omega^t = \arg\min_{\omega} E(\omega, \pi^t)$
 End For
[Solution selection]
Select $(\widehat{\omega}, \widehat{\pi}) = \arg\min_{t=1,\dots,T} R_{val}(\omega^t, \pi^t)$
Return $(\widehat{\omega}, \widehat{\pi})$
```

The GS for the Hyper-parameter choice does not pose any difficulties from the optimization point of view rather than being computationally costly. The critical step is to choose the set of trials Π. Recently in [11] Bertstra at al. proposed an alternative strategy for producing a trial set by random search which is independent drawn from a uniform density from the same configuration space. It has been shown that randomly chosen trials for hyper-parameter optimization produces better generalization results than trials on a grid. The difficulties in the two phase GS procedure above is moved to the Weights Optimization phase which requires the solution of a non convex continuously differentiable unconstrained problem. Depend both on the algorithm for the solution of the subproblems and on the class of machine learning problems addressed, different heuristics for the solution of problem (7) are derived from the two-phase. Of course the scheme above do not encompass all possible decomposition schemes proposed in the literature and depending on the FNN architecture other schemes can be devised. We discuss some special schemes in Section 5.2.2.

Features selection. Recently some methods have been proposed that address the problem of selecting a subset of features, namely the training set $\mathcal{T}$ is not defined one for all but it enters the optimization procedure. The main goal of *Feature Selection* (FS) is to reduce the number n of features by eliminating redundant ones or by selecting the most informative features which capture the relevant properties of data. As we briefly explain in Section 5, the dimension n of the input impacts on the VC dimension which in turn impacts on generalization performance, so that FS can both improve the generalization capability and simplify the optimization problem reducing its dimension. Usually, FS involves combinatorial models and an exhaustive search can conceivably be performed, only if the number of variables is not too large. However, the search becomes quickly computationally intractable (see e.g. [35] and references therein for a survey on FS). Recently the problem of simultaneously selecting a subset of features and the hyper-parameters of the training problem has been addressed. In these cases FS is performed jointly with the hyper-parameters selection with a GO procedure and then weights optimization is performed. Hence this kind of FS scheme can be embedded into a two-phase scheme. As an example of such methods, we mention Simulated Annealing (SA) proposed in [82] in connection with a Back-Propagation Algorithm. The Cross-Entropy Method (CEM) proposed in [23] has been applied to find the smallest feature set that induce the lowest error rate in order to jointly minimize the misclassification rate and the generalization error. The problem has been modelled as stochastic global optimization problem. Numerical results using an SVM as learning machine show that using a CEM procedure seems to be more effective than GS but much more time consuming. In [29] the FS and the joined hyper-parameters and weights optimization for a RBF network are addressed simultaneously. They proposed to use Evolutionary Algorithm (EA) applied to a population of RBF networks (individuals) each represented by its genotype which is made up of a feature binary vector (which indicates the presence ("1") or absence ("0") of one of the possible features in the currently selected feature subset) and the number of centers. Selection for reproduction or for reinsertion is based on an individuals fitness which corresponds to the performance of the trained RBF network. For the training of each RBF in the population they use a three-phase optimization scheme which select the centers C, the radius σ, and the hidden-to-output weights W^2 as described in Section 5.2.2.

Quite recently in [26] a feature ranking method based on the solution of a smooth GO problem based on the available training data has been proposed. The approach is based both on a formal definition of relevant feature which involves the problem of minimum zero-norm inversion of a FNN and on transforming a zero-norm minimization problem into a smooth, concave optimization problem. Computational results on real data sets showed that the method is a valid alternative to existing FS methods in terms of effectiveness.

5 Weights optimization

In the following sections we consider the Weights optimization (WO) problem derived from (7) when the hyper-parameters π are fixed. In particular in this introductive section we discuss the main difficulties in solving the unconstrained problem, that are exploited in the next subsections, and the well known gradient method with focus on recent advances toward global aspects.

Statement and issues of the WO problem. The Weights optimization (WO) problem is a continuous unconstrained problem in the only variables ω

$$\min_{\omega \in \mathbf{R}^q} E(\omega) = \sum_{p=1}^{P} E_p(\omega) + \lambda \|\omega\|^2 \tag{8}$$

where $E_p = \|y(\omega; x^p) - y^p\|^2$, $\lambda > 0$ is assumed fixed.

Actually, when speaking about optimization methods for FNN training, one has often in mind the weights optimization problem rather then the full optimal setting of the all the parameters as in problem (7).

We remark that the existence of the overfitting phenomena had led several authors to suggest that inaccurate weights minimization can be better than good ones [78] which corresponds to affirm that the objective function in (8) is not correctly stated. Indeed most local methods for the solution of (8) use *early stopping* of the optimization procedure, that is the optimization procedure is stopped before getting to convergence, and reaching the global solution is not really pursued. To this aim the validation error is used as a merit/fitness function. This is actually the main reason why not too much effort has been devoted in studying specific GO methods for the WO problem. In the following we focus on strategies for the solution of problem (8) without accounting for generalization properties.

We first note that the problem (8) is well posed for any $\lambda > 0$ in the sense that it admits a solution. In fact the level sets

$$L = \{\omega \in \mathbf{R}^q : E(\omega) \leq E(\omega^0)\}$$

are compact for every $\omega^0 \in \mathbf{R}^q$ thanks to the fact that by definition $E_p(\omega) \geq 0$ and the regularization term gives coerciveness of the objective function. Hence the function E admits a global minimum point ω^* such that $E(\omega^*) \leq E(\omega)$ for all $\omega \in \mathbf{R}^q$. This property may not hold anymore if the regularization term is removed ($\lambda = 0$). We remark however that convergence of standard unconstrained methods strongly relies on compactness of the level sets L.

Assuming continuously differentiability of the transfer function g, any standard gradient based unconstrained method [12] can be used to get a solution of problem (8). A standard method for unconstrained minimization detects only critical/stationary points of $E(\omega)$ namely a point $\widehat{\omega}$ where the gradient vanishes

$$\nabla E(\widehat{\omega}) = 0.$$

The function in (8), in the general case, presents multiple local minima (as recently proved in [55]), hence local search techniques are unable to detect a global or a "good" local minimum of the function $E(\omega)$, and consequently the corresponding minimizer ω^* or its approximation, as they may stuck in "bad" local minima. Even the easiest problems are NP-hard [113, 20], so that the structure of the problem does not allow any simplification. By bad/poor local minimizer we mean a point where the training error is high [114, 76]. Indeed we remark that in principle training error can be lead to zero by using network with a large number q of parameters, as it happens in very deep networks. The knowledge of a bound to the optimal value $E(\omega^*)$ allows to bound the gap between the solution found and the global one and can be used to assess the quality of the solution. This aspect has been exploited e.g. in [123] as discussed in Section 5.3.3.

Research on problem (8) moves along different lines: study of the shape of the objective function of problem (8) with the aim of identifying easy cases; definition of efficient local method, suitable for large scale setting, for identifying "good" stationary points; global tricks for escaping from stationary points.

Most of the specific algorithms for the global solution of problem (8) can be derived by using hybrid schemes combining local methods with global strategy in order to alleviate the disadvantages of local search methods or in order to decrease the overall search time in the case of global search of the weight space. The cost of the global phase must be not too high comparing with the local search. Indeed it may be not worthwhile to spend time to slightly improve the solution. However in the literature much effort has been dedicated to develop local algorithms with improves performance in term of the quality of the solution rather than to the definition of specific global strategies accounting the special nature of the FNN training problem (8). Unless some exception discussed in Section 5.3, standard global heuristic strategies have been applied in connection with the use of efficient local methods rather than specific global heuristic tied on the problem.

Much more effort has been done in studying the properties of the highly nonconvex objective function. Indeed its shape and the behavior in a neighborhood of different type of critical points, as well as the reason why large- and small-size networks achieve different practical performance, are still very poorly understood. Among critical points we may distinguish global, local minimizers and saddle points. It is claimed that local minima and saddle point may not represent a problem for DN and that local algorithms, such as gradient methods, converges to local minimum points that are satisfactory from the generalization point of view even if it does not correspond to a global solution of (7). This is actually true in some cases as explained in Section 5.1.

Many papers has been devoted to identifying the role that negative curvature and saddle points play in the optimization of deep neural networks. In particular using second order information, when available, about the curvature $\nabla^2 E(\widehat{\omega})$, the following type of points [46], playing a major role in FNN, can be considered.

1. *Strict saddle point* [79]. A critical point $\widehat{\omega}$ where $\nabla^2 E(\widehat{\omega})$ is indefinite and non singular; $\widehat{\omega}$ is a saddle point with a $\min - \max$ structure.
2. *Degenerate stationary point.* A critical point $\widehat{\omega}$ where $\nabla^2 E(\widehat{\omega})$ is singular. the point $\widehat{\omega}$ can be a degenerate minimum, maximum or saddle point. When it is a saddle point, the function exhibits a *plateau* along the direction of singularity.

We discuss this topic in Section 5.1.

Gradient based methods for WO. As regard gradient based unconstrained methods for solving problem (8), such as Back Propagation (BP) algorithms [70], it is well known that they may be trapped in the basin of attraction of bad local minima.

This is mainly due to the fact that they enforce monotonic decrease of the objective function thus following the possible narrow curves leading to local minimizers or being slow down along large plateau in the error surface. Allowing non monotonicity of the sequence generated by the unconstrained method, namely in SL allowing the error function values to increase at some epochs, may avoid the region of attraction of local minima [67,64]. Improvements exploiting the possibility of having nonmonotone iteration has been proposed in [97,92,93] proving effectiveness of non monotonicity on the quality of the solution found. Another way of allowing non monotonicity consists in adding a momentum term [99,89] to the updating rule of the gradient iteration with the aim of accelerating convergence. Indeed higher values of the momentum may not immediately result in a significant reduction in error but this may take the optimizers to new regions of the parameter space that may lead to higher-quality local minima (see e.g. [118]).

In [79] it is has been proved that the BP algorithm defined by the iteration $\omega^{k+1} = \omega^k - \eta \nabla E(\omega^k)$, being $\eta > 0$ sufficiently small, almost surely converges to a local minimizer so that saddle point are not a problem. The result holds under the standard Lipschitz assumption on the gradient and the further assumption that each critical point is either a minimum or a strict saddle point where $\nabla^2 E(\omega)$ has at least one negative eigenvalue. Using a local algorithm that enforces convergence to points satisfying the second order necessary conditions allows to avoid the region of attraction of such bad points. It is not clear how stringent the strict saddle point assumption is in the FNN setting.

Another global trick used to enhance effectiveness of local methods is to inject randomness in the network by adding random noise to the to the inputs, outputs, weight connections, and weight changes during BP iteration [5,117,63,21]). In these cases the aim is to improve robustness or regularize the network in an empirical way. Recently in [88] it has been proposed to add noise directly to the gradient in a BP iteration. In particular the gradient at iteration k is modified by a time-dependent Gaussian noise $\mathcal{G}(0, \sigma^2)$ with mean zero, and decaying variance according to the rule

$$\nabla_w E(w^{k+1}) = \nabla_w E(w^k) + \mathcal{G}\left(0, \frac{\theta}{(1+k)^\gamma}\right)$$

where they set $\theta \in \{0.01, 0.3, 1.0\}$ and $\gamma = 0,55$ in their experiments.

The presence of noise encourages active exploration of parameter space. Further higher gradient noise at the beginning of training forces the gradient away from zero in the early stages so to facilitate crossing the plateau in the early learning. Hence this technique gives the model more chances to escape local minima particularly in the case of complicated (deep) neural networks that may have many local minima. Experiments in [88] show that if the initialization is poor, optimization of very deep network can be difficult, and adding noise to the gradient is a good mechanism to overcome these difficulties.

However nowadays the main problem to be faced is the large dimension of the optimization problem to be solved. Indeed dimension enters the problem in two different forms, both in the number of training data P which enters the summation and in the number of parameters q which define the network architecture due to the developments of deep structures.

Hence much research has been devoted to develop local algorithms that exploit the structure of the unconstrained problem in order to get efficient and effective solution. We discuss improvements of local methods that can be viewed as heuristic to detect global solutions in Section 5.2.

5.1 When is global optimization really needed ?

An important question concerns the distribution of critical points, minimum and saddle points of problem (8). Of course we consider only the cases in which the choice of loss and activation functions do not produce trivially a convex problem. For a a list of convex loss functions which together with linear activation functions lead to convex unconstrained problems see e.g. [121].

Much papers have been devoted to study the general shape of the error function when the quadratic loss (MSE) is used, namely:

$$\min_{\omega \in \mathbf{R}^q} E(\omega) = \sum_{p=1}^{P} \|y(\omega; x^p) - y^p\|^2$$

when y is the output of a MLP/DN networks with given hyper-parameters π.

It is claimed that the error surface of an MLP network has in general many local minima. In numerical experiments the behaviour of a typical learning curve shows a *plateau*, namely a slow decrease of the training error for a long time before a sudden improvement. Such a plateau can be easily misunderstood as a local minimum in practical problems. However even in the simple case of XOR function learned by a MLP [83,68,115] it has been showed that there are no local minimizers. Only recent a formal proof of the existence of local minimizers has been derived in [55].

The problem of local minima in MLP/DN has been studied since long time of [59,7,17,128] and it is still a hot topic of research. We report some results on this topic which identify cases when GO is/is not needed.

In [7] it has been proved, under rather strong assumptions on the data matrices, that the MSE function of a linear shallow autoencoder ($L = 2$, $g(z) = z$, $m = n$) has a block-wise convex structure in the sense that $E(\omega)$ is convex in each layer weights W^1 (or W^2) when the other W^2 (or W^1) is fixed and further every local minimum of $E(\omega)$ is a global minimum. All the additional critical points of E are saddle points.

More recent in [8] and later in [76] an extension of this result was given. In particular the next theorem presents an instance of MLP/DN that would be "easy" to train.

Proposition 1 (Loss surface of a MLP/DN linear network [76]) *Consider a MLP/DN linear network ($g(z) = z$) with $m \leq n$. Assume further that*

- *the matrices XX^T and XY^T are full rank;*
- *the matrix $YX^T(XX^T)^{-1}XY^T$ has m distinct eigenvalues.*

Then $E(\omega)$ has the following properties:

i) *it is non-convex non-concave;*
ii) *every local minimum is a global minimum;*
iii) *every critical point which is not a global minimum is a saddle point.*
iv) *If rank$(W^2 \dots W^L) = \min_{\ell=2,\dots,L-1} N^\ell$, then the Hessian at any saddle point has at least one (strictly) negative eigenvalue.*

The assumptions on the training data X, Y are considered realistic. Indeed from the proof of Proposition 1, the authors derived the following result on the effect of deepness on the type of critical points.

Corollary 1 (Loss surface of MLP/DN linear network [76]) *Consider a MLP/DN linear network ($g(z) = z$) with $m \leq n$. Under the same assumptions of Proposition 1 we have that:*

- *for $L = 2$ (shallow MLP) any saddle point $\widehat{\omega}$ is strict (the hessian $\nabla^2 E(\widehat{\omega})$ has at least one negative eigenvalue);*
- *for $L \geq 3$ (DN) there exists saddle points $\widehat{\omega}$ with hessian $\nabla^2 E(\widehat{\omega})$ positive semidefinite.*

Theorems above present cases when the training problem of MLP/DN network can be tractable. In particular, Corollary 1 states that for shallow Linear MLP network, any saddle point has the nice max-min structure so that it is either a global solution or a second order descent direction exists that allows moving from it. Indeed, when the hessian at a critical point has a negative eigenvalue, the negative curvature can be used by second order unconstrained methods [12] to escape from the region of attraction of these saddle points. If we use any of these second order method for a linear shallow network, convergence will be towards critical points $\widehat{\omega}$ with $\nabla^2 E(\widehat{\omega}) \succeq 0$ which cannot be saddle points, so that these are global minima.

For deep network ($L > 1$) instead, there may exists, even for linear network, degenerate saddle points, namely points satisfying the second order necessary conditions but that are not local minimizers. However convergence towards strict saddle points holds under the assumption on the weights matrices expressed by (iv) of Proposition 1. We note that this is an assumption on the sequence generated by the optimization algorithm itself so that it is "a posteriori" assumption on the behaviour of the sequence that cannot guarantee that to be satisfied.

Hence in the linear case, there are quite simple cases that can be solved by means of a local algorithm. When we pass to nonlinear activation function, the surface is much more complicated. In particular even for a ReLU activation function, in [41,42] it has been proved that the loss function of a DN have a combinatorially large number of saddle points. Further the number of local minima diminishes exponentially as the error value increases. The same surface has been studied in [76] where a stronger result was proved, under less unrealistic assumption than in [41,42], showing that poor local minima does not exist. It has been observed that most local minima are equivalent and yield similar performance from the generalization point of view. Furthermore the probability of finding a "bad" local minimum, namely one with large training error, decreases quickly with the network size. Their idea seems to be confirmed by the results proved in [27] on Gaussian fields. Actually they showed that at a critical point $\widehat{\omega}$ there is a strong correlation between the value of the training error $E(\widehat{\omega})$ versus the number of negative eigenvalues of the hessian $\nabla^2 E(\hat{\omega})$; in particular the larger the error, the greater the number of negative eigenvalues. This implies that critical points with much larger error than the global minimum are exponentially likely to be saddle points with a fraction of the negative curvature directions which increases as the error. Conversely all the local minimum are likely to have an error very close to that of the global minimum. Dauphin et al. in [46] prove empirically that the observations of Bray and Dean [27] hold for MLP network. Results above seem to suggest that in DN, namely increasing the size of the network, with ReLu activation function most local minima are equivalent and yields similar generalization performance. Hence for this class of network struggling for finding the global minimum is not worthwhile. However recently in [119] several concrete examples of datasets which cause the error surface to have a suboptimal local minimum are presented when either the sigmoid or the ReLu activation function is used. Thus showing that although difficult to prove, the global minimization of a DN with nonlinear function can be a challenging task.

The effect of increasing neurons in a shallow MLP has been studied in [55]. They analysed the hierarchical geometric structure of the error surface of a MLP shallow network ($L = 2$) with one output ($m = 1$) when the number N of hidden units is increased. The study holds for any sigmoidal activation function g (although $g(z) = \tanh(z)$ has been in the paper), for any monotone nonlinear output unit (although a linear output has been used) and for any loss function. In [55] it is shown that a critical point of the error surface for the MLP model with $N - 1$

hidden units gives a set of critical points in the parametric space of the MLP with N hidden units. More precisely, the critical points corresponding to the global minimum of the smaller network can be embedded in a one-dimensional affine space into the larger network two ways: saddle line segments and line segments of local minima, if any. The author gave an explicit second order condition when this occurs, which formally proves for the first time the existence of local minima in the MSE surface. The possible coexistence of lines of local minima and saddle points in one equivalent set of critical points can cause plateaus in learning of neural networks. Numerical examples that depict these cases are reported in [55]. In general symmetries with respect to the parameters in the objective function E for shallow MLP imply that any local minimizer gives rise to many more local minima (permutations of the weights yield the same learner).

From the results above it turns out that when nonlinear networks (shallow or deep) are used, it is not possible to trust on local algorithms for finding a global solution or a local minimizer close to its optimal value. Hence there is still room of improvements in using global strategies for the solution of the training problem. However the development of local algorithms which exploit the special structure of problem (8) have received much attention and are often confused with global improvements because they may share some global enhancement. These improvements over local methods are discussed in Section 5.2. Global strategies specially developed for the WO problem are not too many. Indeed most of the global problems that have been dealt with arose in the joined choice of hyper-parameters and weights rather than in the WO problem itself. However we discuss a few of them in Section 5.3.

5.2 Local methods: improvement toward better local solutions

In the literature many papers have been devoted to define local algorithms specifically suited to FNNs with improved performance in term of the quality of the solution reached w.r.t. standard local method. Indeed most popular local training algorithms tend to get stuck at the nearest critical point of the error surface which leads to a suboptimal network model.

Local algorithms developed for the Weights Optimization (WO) problem exploited the peculiar structure of the objective function of problem (8) that turn out to be useful to define effective and efficient local strategies suitable for the large scale setting. Indeed the objective function of problem (8) presents some special structure in the fact that i) it is the sum of pieces which takes all the same form $E_p(w)$; ii) thanks to the layered structure, $E(\omega)$ presents a natural block separable structure of the variables ω.

All the methods derived from these observations can guarantee convergence only to stationary points of problem (8) but they proved to be quite good in practice in locating good local solution that are satisfactory from the learning point of view and that often outperform standard local methods.

Roughly speaking most of the approaches for the minimization of the error function are decomposition methods

- w.r.t. the number of samples P, namely exploiting the additive structure of the error function in (8) (sample-wise decomposition);

– w.r.t. the parameters ω, namely exploiting the layered structure of the network (block coordinate-wise decomposition).

In the first case, the most famous and widely used class of algorithms that possess some implicit global properties are on-line methods, namely incremental or stochastic gradient methods. Using gradient on-line methods instead of the batch BP algorithm allows to speed up the optimization process and may produce some enhancement in the quality of the local solution found. We briefly discuss these aspects in Section 5.2.1.

On the other hand, exploiting the structure into layers of the FNN has produced different approaches for finding a critical point of (8). In particular this approach has been investigated for shallow network ($L = 2$) where a "natural" block decomposition of the variables can lead to the solution of convex subproblems, being solvable to global optimality at least for some of the parameters. Of course some questions remain open on how to choose the parameters of the non-convex part of the objective function. Global issues are indeed connected to these aspects and several improvements have been proposed. These approaches are reported in Section 5.2.2.

5.2.1 Sample-wise decomposition methods

In this class of local methods the idea is to exploit the additive structure of the objective function $E(\omega) = \sum_p E_p(\omega)$, thus splitting the problem to reduce complexity w.r.t to the number of samples P. Indeed on-line or incremental methods [14,105] consists in updating the weights' vector at iteration k by using a single term E_{p^k} of the function error, without evaluating the full function E. In particular the on-line Back Propagation (on-line BP) method is an *incremental gradient method* with the following iterative rule for updating the parameters ω

$$\omega^{k+1} = \omega^k - \eta^k \nabla E_{p^k}(\omega^k). \tag{9}$$

where η^k is a positive stepsize called learning rate and p^k is an index in $\{1, \dots, P\}$ selected by some deterministic or randomized rule. The stepsize η^k must be driven to zero to get convergence to a stationary point. Intuitively when the sequence $\{w^k\}$ converges to a solution $\widehat{\omega}$, the vectors $\eta^k \nabla E_{p^k}(w^k)$ must converge to zero, however the gradient of the single component needs not to be zero at a solution $\widehat{\omega}$, so that η^k must be driven to zero.

Often iteration (9) is referred to as stochastic gradient (SG) [103] because it can be seen as a method to minimize the Expected risk $\mathbf{E}\{E((x, y); \omega)\}$ in which $\nabla_w E((x^{p^k}, y^{p^k}); \omega^k)$ is evaluated in a sample (x^{p^k}, y^{p^k}) of the random variable (x, y). The SG is related to the on-line BP method (9) when randomization is used for selecting the component p^k. Indeed when the index selection is done uniformly random, i.e. p^k is chosen among the indexes $\{1, \dots, P\}$ with equal probability $1/P$ and independently of preceding choices, iteration (9) can be viewed as stochastic gradient as clearly reported in [15,14]. We refer to [13,22] and the many references therein for comprehensive surveys on stochastic gradient methods and modifications for computational performance improvements.

From the point of view of global issues, it is interesting to look to iteration (9) as a gradient method perturbed with a random noise term e^k [15,13]. Indeed it can be easily written as

$$w^{k+1} = w^k - \eta^k \nabla E(w^k) - \eta^k \left(\nabla E_{p^k}(w^k) - \nabla E(w^k) \right) = w^k - \eta^k \left(\nabla E(w^k) + e^k \right),$$

thus it is clear that using a single component E_{p^k} of the objective function implicitly introduces an error e^k on the BP iteration. Convergence to a stationary point of incremental method (9) can be proved under suitable assumptions on e^k that in turn depends on the choice of index p^k. The idea underlying the proof of convergence is that, under Lipschitz continuous gradient assumption, the error is proportional to the stepsize η^k, so that a diminishing stepsize drives the error to zero when e^k satisfied suitable properties (see e.g. [13] for convergence analysis). Thus using on-line methods compare with adding a random noise to the gradient, as e.g. in [88], and hence may explain the better performance both in train and generalization. Indeed on-line methods are shown to be more effective in terms of the quality of the solution found w.r.t. batch methods. As an example in [58] Goodfellow et al. presented a computational study designed to understand whether SG encounter obstacles during iterations. Their experiments on MLP/DN (with both ReLU and sigmoidal activation functions) showed that SG may slow down due to the presence of plateau or ravines but there is no evidence that it bumps into local minima or saddle points. Analogous evidence its has been shown for a large shallow MLP in [28] where it is observed that SG trained networks retain "memory" of the initial guess follow different paths for a large number of epochs but converging at the end at the same point. The author claimed this behaviour can be due to the fact that MLP training problem requires optimization of a large collection of largely independent and unrelated functions.

Another possible explanations of the SG behaviour can be the effect of the inherent non monotonicity of the iteration (9), which may give more chances of avoiding the region of attraction of some local minima. Indeed at w^k the gradient of a single component E_{p^k} does not define a descent direction for the full error function E that may bounce up and down repeatedly due to rapid changes in direction of SG. This evolution allows to explore larger regions thus allowing the possibility of escaping regions of attraction of local minimizers.

Similar arguments have been reported in [64] where problem (8) has been formulated as an equivalent unconstrained problem in a larger space using auxiliary variables $z \in \mathbf{R}^{P(q+1)}$. The transformation is based on the definition of an equivalent constrained problem and on the use of an exact augmented Lagrangian to get a new unconstrained problem $\displaystyle\min_{z_p \in \mathbf{R}^{q+1}, \omega \in \mathbf{R}^q} \sum_{p=1}^{P} \Phi_p(z_p, \omega)$ which is block-wise separable w.r.t. the auxiliary variables $z_p \in \mathbf{R}^{q+1}$. The sequence generated is now moving in a larger space $(z, \omega) \in \mathbf{R}^{P(q+1)} \times \mathbf{R}^q$ which together with the natural block-wise decomposition arising in the new unconstrained formulation allow much freedom in the space exploration thus encouraging convergence towards improved local solutions.

Another way of causing non monotonicity in the iteration (9) is to modify it by adding a *momentum* term which add an extrapolation term along the direction $(\omega^k - \omega^{k-1})$ (see e.g. [99,89,13] for non incremental versions). Recently, Sutskever et al. in [118] showed that the use of momentum on Deep Network can produce significant advantages over the pure SG method. Indeed, the "transient phase" of convergence seems to matter a lot more for optimizing DN networks rather than shallow ones so that a slowly increasing update for the momentum parameter, tied with a well-designed random initialization, can work very well on deep autoencoder. Actually the experiments in [118] showed that using higher values of the momentum parameter allows a significantly non-monotonic behaviour of the sequence. Higher values of the momentum may not immediately result in a significant reduction in the error but this may take the optimizers to new regions of the parameter space that may lead to higher-quality local minima.

Overall online methods performs quite well and seems to be able to explore the space of variables and to avoid region of attraction of "bad" local minima. However there is no guarantee of convergence toward global solutions and also the reasons of the good performance are not completely clear, yet.

5.2.2 Block coordinate-wise decomposition methods

In this section methods that exlpoit the layered structure of FFN are reviewed. In particular we focus on

- Two phase methods (randomization algorithms)
- Two blocks decomposition methods (Generalized Gauss Seidel algorithms)

and on some variants of these two main classes.

The layered structure of the FNN allows to exploit the different role played by the parameters in the different layers of the network. Algorithms in this class consists in a sequence of main iterations, indexed by k, such that at each iteration, only a specific set of parameters of the network are updated. In many learning algorithms for FNN the block variable-wise decomposition has been exploited to derive simple learning schemes which possibly require only the solution of convex problems that do not pose any difficulties from the computational point of view and guarantee to reach global optimal solution. In particular this idea has been exploited for shallow FNN ($L = 2$) and well known algorithms devised for WO fit in a two-block decomposition scheme in which the variables ω are split into two blocks $\omega_{I_1}, \omega_{I_2}$, with $I_1 \cup I_2 = \{1, \dots, q\}$, $I_1 \cap I_2 = \emptyset$. Hence problem (8) reads

$$\min_{\substack{\omega_1 \in \mathbf{R}^{q_1} \\ \omega_2 \in \mathbf{R}^{q_2}}} \quad E(\omega_1, \omega_2) \tag{10}$$

where for sake of simplicity we denote $\omega_1 = \omega_{I_1}, \omega_2 = \omega_{I_2}$. Indeed taking into account the form of the outputs (5) and (6) for shallow networks the training optimization problem (10) can be written as

$$\min_{(\omega_1, \omega_2) \in \mathbf{R}^{q_1} \times \mathbf{R}^N} \sum_{p=1}^{P} \|\omega_2^T g\,(\omega_1, x^p) - y^p\|^2 + \lambda(\|\omega_1\|^2 + \|\omega_2\|^2)$$

where $\omega_2^T = W^2$ are the hidden-to-output weights and ω_1 are the parameters of the hidden layer which depend on the chosen architecture (either $\omega_1 = W^1 \in \mathbf{R}^N$ the weights from input-to-hidden layer for MLP or $\omega_1 = C \in \mathbf{R}^{nN}$ the centers in the hidden layer for RBF network). Observing problem above, it turns out that whatever the function g is, when ω_1 are fixed problem (10) reduces to a strictly convex linear least square problem (LLSP). Indeed by introducing the $P\times N$ hidden matrix $H = \{h_{pi}\}$ with elements $h_{pi} = g_i(\omega_1; x^p)$ $p = 1, \dots, P$, $i = 1, \dots, N$, we can re-write problem (10) as

$$\min_{\omega_2 \in \mathbf{R}^N} \|H\omega_2 - Y^T\|^2 + \lambda\|\omega_2\|^2 \tag{11}$$

that can be solve either exactly or approximately to global optimality, obtaining ω_2^*. Of course the solution found depends on the value of the parameters ω_1 which impacts on the matrix H. Hence for any choice of ω_1^t, we get ω_2^{t*} with the property that

$$E(\omega_1^t, \omega_2^{t*}) \le E(\omega_1^t, \omega_2) \quad \forall\ \omega_2 \in \mathbf{R}^N$$

Algorithms for solving the WO problem (10) differ in the way of selecting the parameters ω_1 and definitively the best pair $(\omega_1^t, \omega_2^{t*})$ in terms of the objective function. Of course in general this approach does not guarantee having a global minimizer of the optimization problem (10) in the variables (ω_1, ω_2).

We note that a similar two-block approach can be devised also for deep network. Indeed the choice of setting $\omega_2 = W^L$ and ω_1 as the other parameters in the layers $\ell = 1, \dots, \mathrm{L} - 1$ leads to a problem of the form (11) and preserves the possibility of obtaining a global minimizers with respect to ω_2.

The simplest learning schemes derived by the two-block decomposition of the variables is a simple *two-phase method* in which an initial value ω_1^0 is chosen and a single optimization of (11) is performed to get ω_2^{*0}.

Randomization algorithms for FNN [106, 127] fit in this class of two-phase methods. Indeed a randomized algorithm consists in generating the initial value ω_1^0 randomly according to any continuous probability distribution and in performing a single global (exact or approximate) minimization of problem (11) to get ω_2^{*0}. The returned solution $(\omega_1^0, \omega_2^{*0})$ is not guaranteed to be even a critical point of the error function E nor local solution. Methods in this class differentiate in how the initial guess ω_1^0 is chosen which plays of course a significant role in the performance of the network but never leads to a certified global minimizer of (10). However, this approach had a discrete success in SL due to the simplicity of the training procedure which requires only the solution of the simple problem (11) but also to the fact that such randomized models can reach sound performance compared to fully adaptable ones, as indicated by experimental results, with a number of favorable benefits. This may be due to the effect of randomization of the weights. See [106] and the special issue [127] for overviews of the different ways in which randomization can be applied to the design of shallow FFN and the statistical learning properties of the corresponding randomized learning scheme.

There are two well-known families of methods falling under this basic two block scheme, depending on the FNN architecture chosen (MLP or RBF). For MLP, $\omega^1 = W^1$, the randomization scheme described above corresponds to the well-known *Extreme Learning* (EL) machine [73, 72] which leads to remarkable efficiency compared to traditional gradient methods for MLP. In [129] ELM has been applied also to DN exploiting the two-block idea and applying random choices of the weights in the hidden layers.

On the other hand for RBF, $\omega^1 = C$ and the randomization procedure corresponds to the *unsupervised selection of centers* which consists in choosing the centers C without making use of the output training data. In this case centers C

can be selected either randomly, assigned without any dependence on the training set or from the training set, or using a clustering approach.

Assuming that the aim is to find a global solution problem (10) or at least to improve the quality of the local solution obtainable without accounting for generalization issues, the randomization approach above can be easily improved along different directions.

A first approach to improve the two-phase two-block procedure described above consists in performing a third-phase in which starting from the actual solution $(\omega_1^0, \omega_2^{*0})$ a full optimization of E w.r.t. to both (ω_1, ω_2) is performed using e.g. a BP gradient method to get a new solution $(\widehat{\omega}_1, \widehat{\omega}_2)$ which trivially satisfies

$$E(\widehat{\omega}_1, \widehat{\omega}_2) \leq E(\omega_1^0, \omega_2^{*0}) \qquad \nabla_{(\omega_1,\omega_2)} E(\widehat{\omega}_1, \widehat{\omega}_2) = 0$$

Three-phase procedures have been proposed mostly for RBF networks actually including in the optimization the radius σ too. Indeed the choice of the radii σ_j in RBF network is crucial for the final performance; since the error E is continuously differentiable w.r.t. to σ, it is easily included among the parameters ω_1 to be set by the optimization procedure.

In [109] a third training phase adapts the whole set of parameters C, σ, W^2 of a RBF simultaneously. First the choice of centers C and weights W^2 have performed separately by a two-phase procedure being the initial radius σ fixed. A gradient iteration using a variable stepsize obtained using an Armijo line search is finally performed w.r.t. all the parameters C, σ, W^2. In [43] a similar idea is proposed but they observe that the force which drives the radius to zero is stronger than other optimization forces so that they proposed to add a third phase optimization in C, σ, W^2 which includes in the objective function a term which penalizes small radii

$$E(w) + \sum_{k=1}^{N} \frac{1}{\sigma_k}.$$

Although these improved schemes are often referred to as global method, only convergence to a stationary point can be proved. Furthermore the final optimization phase is no more than the solution of (8) using a warm start from a careful starting guess, thus it has the same computational cost of problem (8) losing the advantages of both solving simpler problems and possibly avoiding the region of attraction of local minimizers retained by decomposition schemes.

A different way to improve the solution obtained by the simple two phase scheme above without losing the advantages of decomposition scheme consists in successive alternating minimization w.r.t. each of the two blocks of variable ω_i in Problem (10), being the other fixed. We will refer to this approach as *successive block minimization.* This corresponds to a two block Gauss-Seidel [14] scheme which starting with an initial guess $\omega_1^0 \in \mathbf{R}^{q_1}$ produces a sequence $\{\omega_1^k, \omega_2^k\}$, $k = 1, 2 \ldots$ according to the rule

$$\begin{aligned} \omega_2^k &= \arg\min_{\omega_2} E(\omega_1^{k-1}, \omega_2), \\ \omega_1^k &= \arg\min_{\omega_1} E(\omega_1, \omega_2^k). \end{aligned}$$

It can be applied to any FNN training problem (independently from the architecture MLP or RBF). Grippo and Sciandrone [66] proved that the two-block Gauss-

Seidel method converges to a stationary point $\nabla E(\omega) = 0$ without requiring any convexity assumption. In particular the following theorem holds.

Theorem 1 (Two-block Gauss-Seidel convergence [66]) *Suppose that the global minimization with respect to each block ω_1, ω_2 is well defined. Then, the two block Gauss-Seidel method generates an infinite sequence $\{(\omega_1^k, \omega_2^k)\}$ such that:*

(i) every limit point of $\{(\omega_1^k, \omega_2^k)\}$ is a stationary point of E;
(ii) if the level set L is compact we have

$$\lim_{k\to\infty} \nabla E(\omega_1^k, \omega_2^k) = 0$$

and there exist at least one limit point that is a stationary point of E.

We remark that even if one would be able to guarantee convergence to a global minimizer with respect to both ω_1 and ω_2, Theorem 1 does not guarantee convergence to a global solution ω^* of the function E. This is true only when further convexity assumption on the full objective function E are imposed. Indeed if E is pseudoconvex, every limit point is a global minimizer [66]. Nevertheless being able to find the global solution of E w.r.t. block ω_2 may help in escaping from irrelevant local minimizers that could not be true in optimization problem without decomposition. Further the use of any global method, either deterministic or stochastic [85], for the solution of the nonconvex problem in the variables ω_1 may lead to an improvement of the overall solution. Hence a Gauss-Seidel two-block decomposition method which alternates exact minimization wr.t. ω_2 and a global methods w.r.t. ω_1 can be considered an heuristic to find a good solution.

However, since in FNN training the subproblem in the variables ω_1 is in general nonlinear and non convex, the computational effort to apply a global method can be excessive compared with respect to the possible improvement of the quality of the solution. Actually the two-block Gauss-Seidel scheme can be generalized in different directions that can be exploited in SL training methods to improve computational performance whilst improving the quality of the solution w.r.t. standard local methods. Among possible modifications, it can be allowed that:

1. [66] the exact global minimization with respect to ω_1 is replaced by a local minimization or even less by the search for a point ω_1^k satisfying
$$E(\omega_1^k, \omega_2^k) \le E(\omega_1^{k-1}, \omega_2^k) \qquad \nabla_{\omega_1} E(\omega_1^k, \omega_2^k) = 0;$$
2. each block minimization w.r.t. ω_i can be performed (exact or approximate) only using a subset of the variables in each block, namely selecting subsets $I^k \subseteq \{1, \dots, q_1\}$ and/or $J^k \subseteq \{1, \dots, N\}$ and solving the subproblems w.r.t. $(\omega_1)_{I^k} \in \mathbf{R}^{|I^k|}$ and/or $(\omega_2)_{J^k} \in \mathbf{R}^{|J^k|}$ (see examples in [31,65]).

A possible decomposition scheme that encompass these generalizations is reported below (where $\overline{I^k}, \overline{J^k}$ are the complementary sets of I^k, J^k respectively) and which is derived by several papers by Grippo and Sciandrone & coauthors [66,31,65].

```
Generalized (G-S)$^2$ two-block algorithm for minimization of $E(\omega_1, \omega_2)$
  Choose starting guess $\omega_1^0 \in \mathbf{R}^{q_1}$. Set $k = 1$.
  Repeat
    1. Select $I^k \subseteq \{1, \ldots, q_1\}$ and $J^k \subseteq \{1, \ldots, N\}$.
    2. [Exact/Approximate solution w.r.t. $(\omega_2)_{J^k}$]
          $(\omega_2)^k_{J^k} = \arg\min_{(\omega_2)_{J^k}} E\left(\omega_1^{k-1}, (\omega_2^{k-1})_{\overline{J^k}}, (\omega_2)_{J^k}\right)$
          $(\omega_2)^k_{\overline{J^k}} = (\omega_2^{k-1})_{\overline{J^k}}$
    3. [Approximate solution w.r.t. $(\omega_1)_{I^k}$]
          $(\omega_1)^k_{I^k} = \arg\min_{(\omega_1)_{I^k}} E\left((\omega_1^{k-1})_{\overline{I^k}}, (\omega_1)_{I^k}, \omega_2^k\right)$
          $(\omega_1)^k_{\overline{I^k}} = (\omega_1^{k-1})_{\overline{I^k}}$
  Until (a stopping criterion satisfied)
  Return $(\widehat{\omega}_1, \widehat{\omega}_2) = (\omega_1^k, \omega_2^k)$.
```

At step 1, the subproblem is still a strictly convex LLSP of type (11) and the possibility of updating only some of the parameters ω_2, those in the index set J^k, is allowed. At Step 2, the subproblem is non convex and an approximate solution must be pursued. However the use of an index set I^k allows to update only some of the components of ω_1 thus decomposing even more the difficult subproblems exploiting the possible structure of E as a function of ω_1. A convergence result of the generalized scheme above analogous to Theorem 1 can be proved under suitable assumptions on the choice of the index sets I^k, J^k and on the properties of the minimization algorithms used at each step. Of course again only convergence to a stationary point can be proved. However, it is easy to see that at each iteration k of the decomposition scheme the solution us improved. Indeed assuming for sake of simplicity that $I^k = \{1, \ldots, q_1\}$ and $J^k = \{1, \ldots, N\}$, a two-phase block decomposition learning method is obtained by the scheme above stopping with $k = 1$; whereas applying the full scheme until convergence produce a sequence which satisfies

$$E(\omega_1^{k+1}, \omega_2^{k+1}) \leq E(\omega_1^k, \omega_2^k) \leq E(\omega_1^1, \omega_2^1)$$

thus improving the solution over standard learning two-phase methods whatever ω_1^0.

Thus, the decomposition scheme allows improving at each iteration over the value in the initial guess, being better of standard learning two-phase block decomposition methods. Block descent techniques can be even more advantageous when used in the context of an early stopping strategy. Indeed decomposition techniques are typically faster during the early stages of the minimization process, thus facilitating leaving prematurely optimization as soon as the error decreases enough.

Different methods using the generalized decomposition have been proposed exploiting the FNN architecture.

In particular, in [65] the generalized two-block decomposition has been applied to obtain an improvement over the basic ELM for a shallow MLP. The proposed decomposition method consider the possibility of updating weights ω_1 using in turn a decomposition method which at each iteration may update even a single component of the vector $(\omega_1)_i$ using a linesearch gradient iteration. Computation can be be efficiently organized in order to reduce the cost of evaluating the objective function E each time that a block component $(\omega_1)_i$ is changed thus requiring much less computational effort than that required in the full optimization approach. Further also the minimization with respect to ω_2 can be performed by further decomposition up to a single component and in this case analytic solution can be found. Numerical results on many instances highlight improvements over ELM both in terms of generalization performance, usually with a smaller number of hidden units, and of training error and computational time.

As regards RBF network the generalized two-block decomposition scheme which select both the centers $\omega^1 = C$ and the hidden-to-output weights $\omega_2 = W^2$ through an optimization procedure is known as *supervised selection of the centers*. In [31] a training algorithm for RBF network has been proposed that updates all the centers $C \in \mathbf{R}^{nN}$ using at Step 2 a gradient iteration with the stepsize chosen by a suitable line search procedure. Actually the separable structure of the centers into $c_j \in \mathbf{R}^n$, $j = 1, \dots, N$ allows to further decompose Step 2 into N small subproblems in the variables c_j, $j = 1, \dots, N$. Thus it turn to be a $N + 1$ block decomposition scheme whose convergence has been proved using suitable linesearch along the gradient direction. We note that that the model encompass also the possibility of alternating global minimization w.r.t. weights ω_2 and inexact minimization w.r.t. center c_j, so that weights are updated to a new global value whenever one single centre is moved. This can hekp even more in escaping by local solutios.

The numerical results reported in [31] shows that this further decomposition yields a reduction in training times w.r.t. standard methods.

The last approach to improve the two/three-phase block decomposition consists in embedding within a global strategy. Actually this case has been exploited mainly for RBF networks with the aim of including within the optimization process also the hyper-parameters σ of the hidden units and the regularization term λ. Most of the proposed method applying evolutionary or genetic methods. In 1999 a Genetic Algorithm (GA) is proposed in [38] for optimization of parameters σ, λ and weights W^2, being the centers fixed as $c_p = x^p$ for $p = 1, \dots, P$. At each iteration the GA acts on a population of RBF networks, each identified by (σ_i, λ_i) and by the weights W^{2i} obtained by solving the corresponding (11). The fitness function of each RBF network in the population is the inverse of the MSE on the validation data set. Random periodic initialization of the population is also considered which helps escaping local solutions. However RBF network with $N = P$ are known to be prone to overfitting and they are no more used.

Later in 2005, in [29] the three-phase optimization of the RBF parameters C, σ, W^2 is embedded within an Evolutionary Algorithm for selecting both the relevant features and the RBF parameters (as already described in Section 4). The algorithm generates a population of RBF networks (individuals) each represented by its genotype which is composed by the features vector and by the number of centers. The training procedure is split into three optimization phases respectively w.r.t. the centers C, the radius σ, and the hidden-to-output weights W^2. Selection among individuals is based on the performance of the RBF network.

Each RBF network produced during evolution describes a possible solution of the optimization problem.

In 2010, an hybrid Simulated Annealing (SA) method has been proposed in [80] which combines the exploration capabilities of the stochastic search by SA with a local method. A randomization algorithm defines the first guess $\omega_1^0 = (C^0, \sigma^0)$ and $\omega_2^0 = (W^2)^0$. New solutions ω_1^k are generated by first applying a random mutation and then applying few iterations of the Levemberg-Marquardt algorithm. Weights ω_2^k are adjusted by solving (11). Convergence in probability is proved and experiments seem to be promising.

5.3 Global strategies for FNN

As already mentioned the dualism in the objectives in SL, low empirical risk and high generalization ability, has lead researchers to concentrate more on the definition of efficient local algorithms rather than on GO strategies. This is mainly due to the fact that the effort paid to improve the solution of the WO problem does not yield in general too much better generalization performance of the network. Nevertheless many papers have been dedicated to GO for the WO problem.We refer to [51,95,96,110,111] for general reviews and comparison among GO methods applied to SL. As a matter of example we mention multistart methods [126,125] and [122] for the role of the starting configurations in DN, simulated annealing [4,102,80], evolutionary algorithms [25,91,86,56], genetic algorithms [47,107,74], particle swarm optimization (PSO) [130], dynamic tunneling in weight space [104]. The TRUST algorithm [33,10] is applied in connection with a back propagation gradient method to a shallow MLP with sigmoidal transfer function in [34] .

Mainly these methods are a straightforward application of well known GO methods to the unconstrained problem (8). However some nice ideas have been proposed that exploits the particular structure of the training problem (8) although they date back to the earlier nineties. We present in this section few classes of hybrid global method specifically developed for the training problem (8) in connection with the use of local algorithms.

5.3.1 The Expanded Range Approximation algorithm: an homotopy method

The Expanded Range Approximation (`ERA`) algorithm is an example for deterministic global optimization strategy specifically designed for MLP proposed in [62, 61,60]. For sake of simplicity we refer to a single output network and we set in (8) the regularized term to zero ($\lambda = 0$).

`ERA` defines an homotopy that works with deformation on the target values y^p, $p = 1, \dots, P$ of the training set. The homotopy is achieved by compressing the target vectors into their mean values, obtaining the *mean target vector*

$$\bar{y} = \frac{1}{P} \sum_{p=1}^{P} y^p$$

and expanding them back to their original value varying a scalar parameter β^t progressively from zero to one as

$$y^p(\beta^t) = (1 - \beta^t)\bar{y} + \beta^t y^p.$$

Correspondingly the MSE surface is parameterized in β as

$$E(\omega;\beta^t) = \sum_{p=1}^{P} \|y(\omega;x^p) - y^p(\beta^t)\|^2$$

At each iteration $t = 0, 1, \dots$ the `ERA` algorithm, starting from a point ω^{0t}, finds $\omega^{*t} = \arg\min_\omega E(\omega;\beta^t)$ and hence it generates a sequence $\{\omega^{*t}\}$. The basic underlying idea is to prove that $\{\omega^{*t}\} \to \omega^*$ as $\beta^t \to 1$ assuming the following:

(i) the first problem $\min_\omega E(\omega;0)$, corresponding to $\beta^0 = 0$, has a unique global minimizer ω^{*0} that can be found easily;
(ii) the parameter β can be increased by a small step ξ without exit from the basin of the attraction of the global minimum ω^{*0};
(iii) it is possible to progressively expand the range parameter $\beta^{t+1} = \beta^t + \xi$, e.g. by a uniform stepsize, without exiting the region of attraction of a global minimizer ω^{*t-1}.

Actually in [62] only a formal proof of (i) has been stated for a simple MLP learning that implements the XOR function. However intensive numerical experiments seems to confirm $E(\omega;0)$ having a unique global minima. The analysis for proving (ii) goes trough the first order expansion $E(\omega;\xi) = E(\omega;0) + \nabla_\beta E(\omega;0)\xi + O(\xi^2)$ with the aim of showing that for small ξ the first order term does not create any local minima but simply shift the location of the global one, but a formal proof of statement (ii) is not given. However the authors observed numerically that when a global minimizer ω^{*1} of the parametrized function $E(\omega;\beta^1)$ can be found, the subsequent `ERA` iterations never fail; whereas a failure at the first step gives subsequently unsuccessful iterations as β is driven to one. No rule for the size of the first and subsequent steps in order to ensure convergence have been suggested. Assuming that the first small step works, in order to ensure success of the subsequent iterations the starting point ω^{0t} of the $t-$th minimization must belong to the basin of attraction of the global minimum ω^{*t}. To this aim the authors suggest to use as warm start procedure, setting $\omega^{0t} = \omega^{*t-1}$. The main difficulty stays in the fact that, as the authors observed, bifurcations can appear at a certain value of β in the homotopy path which may lead far fromt he global solution.

An advantage of this approach is that it is independent from the local method used for solving the unconstrained problem and from the chosen architecture. Further it is very simple to implement. However a lot of questions remain still open and indeed it has not been used occasionally in practice [87].

5.3.2 Uniform space exploration: `NOVEL`*,* `TRUST-TECH`*,* `TP-ES-BP` *algorithms*

Local methods begin at some initial guess of the weights and deterministically lead to a nearby local minimum. Usually the quality of the final solution depends significantly on the initial set of parameters available and uniform space exploration is a global optimization techniques which tries to explore the entire solution space effectively in order to locate the more promising local solutions. The simplest examples being uniform random sampling over a bounded region, like multistart methods. However more efficient sampling have been proposed and applied to problem (8).

The NOVEL algorithm [112] is a trajectory-based method that explores the solution space and locates promising regions from which a local optimization can start. The global phase is based on a trajectory defined by Ordinary Differential Equations (ODE) as:

$$\dot{w}(t) = P\left(\nabla_w E(w(t))\right) + Q\left(\mathcal{T}(t), w(t)\right)$$

where P, Q are generic non linear function and $\mathcal{T}$ is called *trace* function which plays the role of covering the space uniformly. In the paper [112] the authors proposed to use a linear form for P, Q and, after experimentation, they set

$$\mathcal{T}_i(t) = \rho \sin\left(2\pi \left(\frac{t}{2}\right)^{1-(a+b(i-1))/q} + \frac{2\pi(i-1)}{q}\right) \qquad i = 1\ldots, q$$

For the local phase they can use any available unconstrained method starting with point chosen along the trajectory. Multistart along the trajectory is exploited and best solution is selected. Although proposed in the contest of FFN global optimization, NOVEL method does not exploit the special form of the objective function of FNN network training problem and it is rather suitable for any unconstrained nonlinear problem. It requires the solution of an ODE which is computationally expensive and works in batch mode. Author proposed to use a discretized version, but still facing numerical and computational difficulties. Hence Novel method is not suitable for large scale setting.

Another method that falls in the space exploration is the TRUST-TECH algorithm [39] (TRansformation Under STability-reTaining Equilibria CHaracterization). The basic idea TRUST-TECH is to move out of the local minimum in a deterministic way and systematically exploring multiple local minima on the error surface in a tier-by-tier manner in order to advance towards the global minimum. Once local minimizer has been detected, moving directions are obtained by using the reformulation of the original problem (8) as a nonlinear dynamical system $\frac{d\omega}{dt} = -R(\omega)\nabla E(\omega)$ where R is a positive definite matrix. The underlying principle is to perform small steps along multiple eigenvectors of the Jacobian of the dynamic system. Along these directions the objective function value first increases till it hits the stability boundary, after then it start decreases. Hence a local methods starting along the trajectory may move to the region of attraction of another minimizer. From all these solutions, the best one is chosen to be the desired global optimum.

A different approach that allows a wide exploration of the weights space in proposed in [45] A two-phased and Ensemble scheme integrated Backpropagation (TP-ES-BP) algorithm is proposed to detect a bunch of potential solutions. Indeed, during training, whenever the objective function is below a threshold value the network weights matrices are stored up and a random perturbation terms, generated with white Gaussian noise, is added to the network weights so to escape the region of attraction of the local minimum and learning procedure starts again. At the end of this phase, a bucket of several potential network solutions is achieved and a neural network ensemble (NNE) system [48] is trained. Authors show that their approach can overcome the limitation of individual networks performance.

5.3.3 Bounding the Search Space

As we discussed above, many deterministic and stochastic global search techniques perform exhaustive/random sampling of the weight space. However the domain of search space is not known in advance and so, in practice, global optimization procedures search for a global minimum in a prefixed box $\mathcal{D} \subset \mathbf{R}^q$ and the original unconstrained global optimization problem (8) is arbitrarily converted to an ad-hoc bound constrained optimization one. Usually the bounds $\mathcal{D}$ of the search region are intuitively defined and a common technique is to define the region of the search space to be "as large as possible". However, such a choice may lead to unsuccessful training or give a computationally expensive solution that is impractical to use in real life problems. Using interval analysis Adam et al. in [2] define a procedure for restricting the box for the range of weights in the case of MLP/DN. These bounds depend on the activation functions of hidden and output layers, the number of neurons in the input, hidden and output layers and the precision of the machine. In [2] the values of the bounds for a shallow MLP with a logistic or hyperbolic tangent activation function are reported. However a thorough performance analysis of global optimization procedures with and without using the proposed approach was not reported in [2]. Hence it is not possible to state clearly the impact in terms of computational complexity and cost of the proposed approach.

Another way of bounding the search space can be derived by exploiting the fact that a bound is known on the optimal value $E(\omega^*)$. In particular in [123] a monotonically decreasing transformation $v(\cdot)$ of $E(\omega)$ is defined, given by $v(\omega;\gamma) = e^{-\gamma E(\omega)}$ and it is observed that if a lower bound $\widehat{E}$ on the value of global minimum of $E(\omega)$ were known, then we could multiply $E(\omega)$ by $e^{\gamma\widehat{E}}$ for scaling purposes and get $v(\omega;\gamma) = e^{-\gamma(E(\omega)-\widehat{E})}$ This means that the best value of $v(\omega)$ can be normalized near the value 1 while all other local solutions can be "flattened" using a sufficiently high value of γ. In the case of FNN with LSE as loss function, the ultimate lower bound of $E(\omega^*)$ is zero. Indeed, let Ω be the set of all local minimizers of $E(\omega)$, Toh in [123] proved the following: let $\omega^* \in \Omega$, if it exists a value $\bar{\gamma} > 1$ such that for all $\gamma \geq \bar{\gamma}$ we have

$$e^{-\gamma(E(\omega)-E(\omega^*))} \leq \xi,\ \ \xi \in (0,1) \qquad \forall\ \ \omega \in \Omega \text{ with } E(\omega) \neq E(\omega^*),$$

then ω^* is a global minimizer of problem (8).

Hence, under these conditions, a level $\bar{\xi} \geq \xi$ can be found such that adding to problem (8) the constraint $\rho e^{-\gamma E(\omega)} \leq \bar{\xi}$ segregates the global minima from the other local minima. Since $E(\omega^*) \geq 0$ these cutting level can be set to be slightly less than 1 with sufficiently high value of γ and a global constrained minimization of E must be performed. Now the problem is moved to solve to global optimality a nonlinearly constrained problem. In the paper the author used a simple penalization approach that however suffers of numerical problems (due to the increasing value of the penalty parameter) and it cannot be guaranteed to reach a global solution.

5.3.4 Deformations of the objective function

Among hybrid global strategies, methods based on the modification of the objective function have been proposed with the aim of moving successively from one local minimum to another better one. Being $\widehat{\omega}$ the current solution, the basic idea is to get a new function $F(\omega;\widehat{\omega})$ by modification of E such that $\widehat{\omega}$ is no more a local solution of F and the other solution are preserved. Hence a local minimization of F starting from $\widehat{\omega}$ or a slight perturbation allow to escape from $\widehat{\omega}$. Methods in this class does not receive too much attention in FNN community. However some few examples have been proposed.

The basic idea outlined above dates back to the early nineties with the filled function proposed by Ge in [101] which aim to “fill” the basin of the current local solution in order to escape from it. Many papers have been published proposing new filled functions (see e.g. the very recent [84] and references therein) but up to our knowledge they have not been applied to FNN with the few exceptions [52, 81].

A different approach has been proposed in [95] where a deflection function $\delta(\omega;\widehat{\omega})$ is proposed to escape from current local minima $\widehat{\omega}$. In a local minimizer $\widehat{\omega}$ a deflection function $\delta(\omega;\widehat{\omega})$ for $E(\omega)$ is a multiplicative function such that the composition $F(\omega;\widehat{\omega}) = \delta(\omega;\widehat{\omega})E(\omega)$ does not present anymore a local minimizer in $\widehat{\omega}$ while all other local solutions remain unchanged. A function which possess this deflection property is $\delta = \tanh(\sigma\|w-\widehat{w}\|)$. Plagianakos et al. in [95] proposed to apply a deflection procedure to each minimizer $\widehat{\omega}^t$, $t = 1, 2\ldots,$ encountered during the minimization process using a gradient iteration to minimize function

$$F(\omega;\widehat{\omega}^1,\widehat{\omega}^2,\ldots) = \tanh(\sigma\|w-\widehat{w}^1\|)\tanh(\sigma\|w-\widehat{w}^1\|)\ldots E(\omega).$$

This algorithm (*Steepest Descent with Deflection* SDD) can be embedded within a stochastic GO procedure such as SA, GA or EA. Results on small problem (XOR and parity problems) showed that the SDD has better performance than BP algorithm.

An additive modification of E is proposed in [40]. In their approach using the LLSQ solution they eliminated the the hidden-to-output weights (using pseudo inverse) and they wrote the training problem in the reduced space of input-to-hidden weights. Whenever the local algorithm bumps into a critical point, a penalty term of Gaussian type on the weights is added which has the effect to force the iteration out of the current point. Numerical results on small scale problem (XOR and spiral problems) are reported.

6 Conclusion

We reviewed a few of the numerous global issues arising in Machine Learning. In particular we focused on Supervised Learning using Feedforward Neural Network. We review local methods with proved convergence properties and we focus on the global hidden properties tied to randomization, on the non monotonic behaviour of the sequence and on modified versions that allow to improve the quality of the solution. Any of these local algorithms can be embedded within specific global strategies devised for the training subproblem. We review deterministic hybrid algorithms specifically developed for the special structure of the FNN training problem. We only touched upon standard global methods used without devising ad hoc procedure for the problem. Some important problems have not been tackled at all, such as unsupervised classification (clustering) or sparse supervised learning.

Table 6 Abbreviations in Alphabetical Order

CEM	Cross-Entropy Method
DN	Deep Network
FNN	Feedforward Neural Network
FS	Feature Selection
GA	Genetic Algorithm
GO	Global Optimization
GS	Grid-Search
ML	Machine Learing
MLP	(Shallow) Multilayer Perceptron
MSE	Mean Square Error
PSO	Particle Swarm Optimization
RBF	Radial Basis function
SA	Simulated Annealing
SL	Supervised Learning
SVM	Support Vector Machines
SRM	Structural Risk Minimization
UL	Unsupervised Learning
WO	Weights optimization

References

1. Abraham, A.: Meta learning evolutionary artificial neural networks. Neurocomputing **56**, 1–38 (2004)
2. Adam, S., Magoulas, G., Karras, D., Vrahatis, M.: Bounding the search space for global optimization of neural networks learning error: an interval analysis approach. J. of Machine Learning Research **17**, 1–40 (2016)
3. Adamu, A., Maul, T., Bargiela, A.: On training neural networks with transfer function diversity. In: International Conference on Computational Intelligence and Information Technology (CIIT 2013), Elsevier (2013)
4. Amato, S., Apolloni, B., Caporali, G., Madesani, U., Zanaboni, A.: Simulated annealing approach in backpropagation. Neurocomputing **3**(5), 207–220 (1991)
5. An, G.: The effects of adding noise during backpropagation training on a generalization performance. Neural computation **8**(3), 643–674 (1996)
6. Bagirov, A., Rubinov, A., Soukhoroukova, N., Yearwood, J.: Unsupervised and supervised data classification via nonsmooth and global optimization. Top **11**(1), 1–75 (2003)
7. Baldi, P., Hornik, K.: Neural networks and principal component analysis: Learning from examples without local minima. Neural networks **2**(1), 53–58 (1989)
8. Baldi, P., Lu, Z.: Complex-valued autoencoders. Neural Networks **33**, 136–147 (2012)
9. Baldi, P., Sadowski, P.: The dropout learning algorithm. Artificial intelligence **210**, 78–122 (2014)
10. Barhen, J., Protopopescu, V., Reister, D.: Trust: A deterministic algorithm for global optimization. Science **276**(5315), 1094–1097 (1997)
11. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research **13**(Feb), 281–305 (2012)
12. Bertsekas, D.P.: Nonlinear programming. Athena scientific Belmont (1999)
13. Bertsekas, D.P.: Incremental gradient, subgradient, and proximal methods for convex optimization: A survey. Optimization for Machine Learning **2010**(1-38), 3 (2011)
14. Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and Distributed Computation: Numerical Methods. Prentice-Hall, Englewood Cliffs, N.J. (1989)
15. Bertsekas, D.P., Tsitsiklis, J.N.: Gradient convergence in gradient methods with errors. SIAM Journal on Optimization **10**(3), 627–642 (2000)
16. Bertsimas, D., Shioda, R.: Classification and regression via integer optimization. Operations Research **55**(2), 252–271 (2007)
17. Bianchini, M., Frasconi, P., Gori, M.: Learning without local minima in radial basis function networks. IEEE Transactions on Neural Networks **6**(3), 749–756 (1995)
18. Bishop, C.: Improving the generalization properties of radial basis function neural networks. Neural computation **3**(4), 579–588 (1991)

19. Bishop, C.: Pattern Recognition and Machine Learning (Information Science and Statistics), 1st edn. 2006. corr. 2nd printing edn (2007)
20. Blum, A., Rivest, R.L.: Training a 3-node neural network is np-complete. In: Proceedings of the 1st International Conference on Neural Information Processing Systems, pp. 494–501. MIT Press (1988)
21. Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D.: Weight uncertainty in neural networks. arXiv preprint arXiv:1505.05424 (2015)
22. Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large-scale machine learning. arXiv preprint arXiv:1606.04838 (2016)
23. Boubezoul, A., Paris, S.: Application of global optimization methods to model and feature selection. Pattern Recognition **45**(10), 3676–3686 (2012)
24. Bousquet, O., Bottou, L.: The tradeoffs of large scale learning. In: Advances in neural information processing systems, pp. 161–168 (2008)
25. Branke, J.: Evolutionary algorithms for neural network design and training. In: In Proceedings of the First Nordic Workshop on Genetic Algorithms and its Applications. Citeseer (1995)
26. Bravi, L., Piccialli, V., Sciandrone, M.: An optimization-based method for feature ranking in nonlinear regression problems. IEEE transactions on neural networks and learning systems **28**(4), 1005–1010 (2017)
27. Bray, A.J., Dean, D.S.: Statistics of critical points of gaussian fields on large-dimensional spaces. Physical review letters **98**(15), 150,201 (2007)
28. Breuel, T.M.: On the convergence of sgd training of neural networks. arXiv preprint arXiv:1508.02790 (2015)
29. Buchtala, O., Klimek, M., Sick, B.: Evolutionary optimization of radial basis function classifiers for data mining applications. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **35**(5), 928–947 (2005)
30. Burges, C.J.: A tutorial on support vector machines for pattern recognition. Data mining and knowledge discovery **2**(2), 121–167 (1998)
31. Buzzi, C., Grippo, L., Sciandrone, M.: Convergent decomposition techniques for training rbf neural networks. Neural Computation **13**(8), 1891–1920 (2001)
32. Carrizosa, E., Morales, D.R.: Supervised classification and mathematical optimization. Computers & Operations Research **40**(1), 150–165 (2013)
33. Cetin, B., Barhen, J., Burdick, J.: Terminal repeller unconstrained subenergy tunneling (trust) for fast global optimization. Journal of optimization theory and applications **77**(1), 97–126 (1993)
34. Cetin, B.C., Burdick, J.W., Barhen, J.: Global descent replaces gradient descent to avoid local minima problem in learning with artificial neural networks. In: Neural Networks, 1993., IEEE International Conference on, pp. 836–842. IEEE (1993)
35. Chandrashekar, G., Sahin, F.: A survey on feature selection methods. Computers & Electrical Engineering **40**(1), 16–28 (2014)
36. Chao, J., Hoshino, M., Kitamura, T., Masuda, T.: A multilayer rbf network and its supervised learning. In: Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on, vol. 3, pp. 1995–2000. IEEE (2001)
37. Chapelle, O., Sindhwani, V., Keerthi, S.S.: Optimization techniques for semi-supervised support vector machines. Journal of Machine Learning Research **9**(Feb), 203–233 (2008)
38. Chen, S., Wu, Y., Luk, B.: Combined genetic algorithm optimization and regularized orthogonal least squares learning for radial basis function networks. IEEE Transactions on Neural Networks **10**(5), 1239–1243 (1999)
39. Chiang, H.D., Reddy, C.K.: Trust-tech based neural network training. In: Neural Networks, 2007. IJCNN 2007. International Joint Conference on, pp. 90–95. IEEE (2007)
40. Cho, S.y., Chow, T.W.: Training multilayer neural networks using fast global learning algorithm-least-squares and penalized optimization methods. Neurocomputing **25**(1), 115–131 (1999)
41. Choromanska, A., Henaff, M., Mathieu, M., Arous, G.B., LeCun, Y.: The loss surfaces of multilayer networks. In: AISTATS (2015)
42. Choromanska, A., LeCun, Y., Arous, G.B.: Open problem: The landscape of the loss surfaces of multilayer networks. In: COLT, pp. 1756–1760 (2015)
43. Cohen, S., Intrator, N.: Global optimization of rbf networks. URL: http://www. cs. tau. ac. il/˜ nin/papers/rbf. pdf (cf. p. 156) (2000)
44. Cohen, S., Intrator, N.: A hybrid projection-based and radial basis function architecture: initial values and global optimisation. Pattern Analysis & Applications **5**(2), 113–120 (2002)

45. Dai, Q., Ma, Z., Xie, Q.: A two-phased and ensemble scheme integrated backpropagation algorithm. Applied Soft Computing **24**, 1124–1135 (2014)
46. Dauphin, Y.N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., Bengio, Y.: Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In: Advances in neural information processing systems, pp. 2933–2941 (2014)
47. David, O.E., Greental, I.: Genetic algorithms for evolving deep neural networks. In: Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation, pp. 1451–1452. ACM (2014)
48. Dietterich, T.G.: Ensemble methods in machine learning. In: International workshop on multiple classifier systems, pp. 1–15. Springer (2000)
49. Duch, W., Jankowski, N.: New neural transfer functions. Applied Mathematics and Computer Science **7**, 639–658 (1997)
50. Duch, W., Jankowski, N.: Survey of neural transfer functions. Neural Computing Surveys **2**(1), 163–212 (1999)
51. Duch, W., Korczak, J.: Optimization and global minimization methods suitable for neural networks. Neural computing surveys **2**, 163–212 (1998)
52. Feng-wen, H., Ai-ping, J.: An improved method of wavelet neural network optimization based on filled function method. In: Industrial Engineering and Engineering Management, 2009. IE&EM'09. 16th International Conference on, pp. 1694–1697. IEEE (2009)
53. Fischetti, M.: Fast training of support vector machines with gaussian kernel. Discrete Optimization **22**, 183–194 (2016)
54. Floudas, C.A.: Deterministic global optimization: theory, methods and applications, vol. 37. Springer Science & Business Media (2013)
55. Fukumizu, K., Amari, S.i.: Local minima and plateaus in hierarchical structures of multilayer perceptrons. Neural Networks **13**(3), 317–327 (2000)
56. González, J., Rojas, I., Ortega, J., Pomares, H., Fernandez, F.J., Díaz, A.F.: Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation. IEEE Transactions on Neural Networks **14**(6), 1478–1495 (2003)
57. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT Press (2016)
58. Goodfellow, I.J., Vinyals, O.: Qualitatively characterizing neural network optimization problems. CoRR (2014). URL http://arxiv.org/abs/1412.6544
59. Gori, M., Tesi, A.: On the problem of local minima in backpropagation. IEEE Transactions on Pattern Analysis and Machine Intelligence **14**(1), 76–86 (1992)
60. Gorse, D., Shepherd, A.J., Taylor, J.G.: Avoiding local minima by a classical range expansion algorithm. In: ICANN94, pp. 525–528. Springer London (1994)
61. Gorse, D., Shepherd, A.J., Taylor, J.G.: A classical algorithm for avoiding local minima. In: Proceedings of the World Congress on Neural Networks, pp. 364–369. Citeseer (1994)
62. Gorse, D., Shepherd, A.J., Taylor, J.G.: The new era in supervised learning. Neural Networks **10**(2), 343–352 (1997)
63. Graves, A.: Practical variational inference for neural networks. In: Advances in Neural Information Processing Systems, pp. 2348–2356 (2011)
64. Grippo, L.: Convergent on-line algorithms for supervised learning in neural networks. IEEE Transactions on Neural Networks **11**(6), 1284–1299 (2000)
65. Grippo, L., Manno, A., Sciandrone, M.: Decomposition techniques for multilayer perceptron training. IEEE Transactions on Neural Networks and Learning Systems **27**(11), 2146–2159 (2016)
66. Grippo, L., Sciandrone, M.: Globally convergent block-coordinate techniques for unconstrained optimization. Optimization methods and software **10**(4), 587–637 (1999)
67. Grippo, L., Sciandrone, M.: Nonmonotone globalization techniques for the barzilai-borwein gradient method. Computational Optimization and Applications **23**(2), 143–169 (2002)
68. Hamey, L.G.: Xor has no local minima: A case study in neural network error surface analysis. Neural Networks **11**(4), 669–681 (1998)
69. Hamm, L., Brorsen, B.W., Hagan, M.T.: Comparison of stochastic global optimization methods to estimate neural network weights. Neural Processing Letters **26**(3), 145–158 (2007)
70. Haykin, S.: Neural networks and learning machines, vol. 3. Pearson Upper Saddle River, NJ, USA: (2009)
71. Horst, R., Tuy, H.: Global optimization: Deterministic approaches. Springer Science & Business Media (2013)

72. Huang, G., Huang, G.B., Song, S., You, K.: Trends in extreme learning machines: A review. Neural Networks **61**, 32–48 (2015)
73. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: a new learning scheme of feedforward neural networks. In: Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on, vol. 2, pp. 985–990. IEEE (2004)
74. Hui, L.C.K., Lam, K.Y., Chea, C.W.: Global optimisation in neural network training. Neural Computing & Applications **5**(1), 58–64 (1997)
75. Jin, Y., Sendhoff, B.: Pareto-based multiobjective machine learning: An overview and case studies. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **38**(3), 397–415 (2008)
76. Kawaguchi, K.: Deep learning without poor local minima. In: Advances In Neural Information Processing Systems, pp. 586–594 (2016)
77. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
78. LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.R.: Efficient backprop. In: Neural networks: Tricks of the trade, pp. 9–48. Springer (2012)
79. Lee, J.D., Simchowitz, M., Jordan, M.I., Recht, B.: Gradient descent only converges to minimizers. In: Conference on Learning Theory, pp. 1246–1257 (2016)
80. Lee, J.S., Park, C.H.: Global optimization of radial basis function networks by hybrid simulated annealing. Neural Network World **20**(4), 519 (2010)
81. Li, H.R., Li, H.L.: A global optimization algorithm based on filled-function for neural networks. JOURNAL-NORTHEASTERN UNIVERSITY NATURAL SCIENCE **28**(9), 1247 (2007)
82. Lin, S.W., Tseng, T.Y., Chou, S.Y., Chen, S.C.: A simulated-annealing-based approach for simultaneous parameter optimization and feature selection of back-propagation networks. Expert Systems with Applications **34**(2), 1491–1499 (2008)
83. Lisboa, P., Perantonis, S.: Complete solution of the local minima in the xor problem. Network: Computation in Neural Systems **2**(1), 119–124 (1991)
84. Liu, H., Wang, Y., Guan, S., Liu, X.: A new filled function method for unconstrained global optimization. International Journal of Computer Mathematics pp. 1–14 (2017)
85. Locatelli, M., Schoen, F.: Global optimization: theory, algorithms, and applications. SIAM (2013)
86. Magoulas, G., Plagianakos, V., Vrahatis, M.: Hybrid methods using evolutionary algorithms for on-line training. In: Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on, vol. 3, pp. 2218–2223. IEEE (2001)
87. Martin-Guerreo, J., Gómez-Chova, L., Calpe-Maravilla, J., Camps-Valls, G., Soria-Olivas, E., Moreno, J.: A soft approach to era algorithm for hyperspectral image classification. In: Image and Signal Processing and Analysis, 2003. ISPA 2003. Proceedings of the 3rd International Symposium on, vol. 2, pp. 761–765. IEEE (2003)
88. Neelakantan, A., Vilnis, L., Le, Q.V., Sutskever, I., Kaiser, L., Kurach, K., Martens, J.: Adding gradient noise improves learning for very deep networks. arXiv preprint arXiv:1511.06807 (2015)
89. Nesterov, Y.: A method of solving a convex programming problem with convergence rate o (1/k2). In: Soviet Mathematics Doklady, vol. 27, pp. 372–376 (1983)
90. Ojha, V.K., Abraham, A., Snášel, V.: Metaheuristic design of feedforward neural networks: A review of two decades of research. Engineering Applications of Artificial Intelligence **60**, 97–116 (2017)
91. Palmes, P.P., Hayasaka, T., Usui, S.: Mutation-based genetic neural network. IEEE Transactions on Neural Networks **16**(3), 587–600 (2005)
92. Peng, C.C., Magoulas, G.D.: Adaptive nonmonotone conjugate gradient training algorithm for recurrent neural networks. In: Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on, vol. 2, pp. 374–381. IEEE (2007)
93. Peng, C.C., Magoulas, G.D.: Nonmonotone levenberg–marquardt training of recurrent neural architectures for processing symbolic sequences. Neural Computing and Applications **20**(6), 897–908 (2011)
94. Pintér, J.D.: Calibrating artificial neural networks by global optimization. Expert Systems with Applications **39**(1), 25–32 (2012)
95. Plagianakos, V., Magoulas, G., Vrahatis, M.: Learning in multilayer perceptrons using global optimization strategies. Nonlinear Analysis: Theory, Methods & Applications **47**(5), 3431–3436 (2001)
96. Plagianakos, V., Magoulas, G., Vrahatis, M.: Improved learning of neural nets through global search. In: Global Optimization, pp. 361–388. Springer (2006)

97. Plagianakos, V.P., Magoulas, G.D., Vrahatis, M.N.: Deterministic nonmonotone strategies for effective training of multilayer perceptrons. IEEE Transactions on Neural Networks **13**(6), 1268–1284 (2002)
98. Poggio, T., Girosi, F.: Networks for approximation and learning. Proceedings of the IEEE **78**(9), 1481–1497 (1990)
99. Polyak, B.T.: Some methods of speeding up the convergence of iteration methods. USSR Computational Mathematics and Mathematical Physics **4**(5), 1–17 (1964)
100. Prieto, A., Prieto, B., Ortigosa, E.M., Ros, E., Pelayo, F., Ortega, J., Rojas, I.: Neural networks: An overview of early research, current frameworks and new challenges. Neurocomputing **214**, 242–268 (2016)
101. Renpu, G.: A filled function method for finding a global minimizer of a function of several variables. Mathematical programming **46**(1-3), 191–204 (1990)
102. Rere, L.R., Fanany, M.I., Arymurthy, A.M.: Simulated annealing algorithm for deep learning. Procedia Computer Science **72**, 137–144 (2015)
103. Robbins, H., Monro, S.: A stochastic approximation method. The annals of mathematical statistics pp. 400–407 (1951)
104. RoyChowdhury, P., Singh, Y.P., Chansarkar, R.: Dynamic tunneling technique for efficient training of multilayer perceptrons. IEEE transactions on Neural Networks **10**(1), 48–55 (1999)
105. Saad, D.: On-line learning in neural networks, vol. 17. Cambridge University Press (2009)
106. Scardapane, S., Wang, D.: Randomness in neural networks: an overview. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **7**(2) (2017)
107. Schaffer, J.D., Whitley, D., Eshelman, L.J.: Combinations of genetic algorithms and neural networks: A survey of the state of the art. In: Combinations of Genetic Algorithms and Neural Networks, 1992., COGANN-92. International Workshop on, pp. 1–37. IEEE (1992)
108. Schmidhuber, J.: Deep learning in neural networks: An overview. Neural networks **61**, 85–117 (2015)
109. Schwenker, F., Kestler, H.A., Palm, G.: Three learning phases for radial-basis-function networks. Neural networks **14**(4), 439–458 (2001)
110. Sexton, R.S., Dorsey, R.E., Johnson, J.D.: Toward global optimization of neural networks: a comparison of the genetic algorithm and backpropagation. Decision Support Systems **22**(2), 171–185 (1998)
111. Sexton, R.S., Dorsey, R.E., Johnson, J.D.: Optimization of neural networks: A comparative analysis of the genetic algorithm and simulated annealing. European Journal of Operational Research **114**(3), 589–601 (1999)
112. Shang, Y., Wah, B.W.: Global optimization for neural network training. Computer **29**(3), 45–54 (1996)
113. Šíma, J.: Training a single sigmoidal neuron is hard. Neural Computation **14**(11), 2709–2728 (2002)
114. Soudry, D., Carmon, Y.: No bad local minima: Data independent training error guarantees for multilayer neural networks. arXiv preprint arXiv:1605.08361 (2016)
115. Sprinkhuizen-Kuyper, I.G., Boers, E.J.: The error surface of the 2-2-1 xor network: The finite stationary points. Neural Networks **11**(4), 683–690 (1998)
116. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15**(1), 1929–1958 (2014)
117. Steijvers, M., Grünwald, P.: A recurrent network that performs a context-sensitive prediction task. In: Proceedings of the 18th Annual Conference of the Cognitive Science Society, pp. 335–339 (1996)
118. Sutskever, I., Martens, J., Dahl, G.E., Hinton, G.E.: On the importance of initialization and momentum in deep learning. ICML (3) **28**, 1139–1147 (2013)
119. Swirszcz, G., Czarnecki, W.M., Pascanu, R.: Local minima in training of neural networks. stat **1050**, 17 (2017)
120. Teboulle, M.: A unified continuous optimization framework for center-based clustering methods. Journal of Machine Learning Research **8**(Jan), 65–102 (2007)
121. Teo, C.H., Smola, A., Vishwanathan, S., Le, Q.V.: A scalable modular convex solver for regularized risk minimization. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 727–736. ACM (2007)
122. Tirumala, S.S., Ali, S., Ramesh, C.P.: Evolving deep neural networks: A new prospect. In: Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2016 12th International Conference on, pp. 69–74. IEEE (2016)

123. Toh, K.A.: Deterministic global optimization for FNN training. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **33**(6), 977–983 (2003)
124. Vapnik, V.: The nature of statistical learning theory. Springer science & business media (2013)
125. Voglis, C., Lagaris, I.: A global optimization approach to neural network training. Neural, Parallel and Scientific Computations **14**(2), 231 (2006)
126. Voglis, C., Lagaris, I.E.: Towards ideal multistart. a stochastic approach for locating the minima of a continuous function inside a bounded domain. Applied Mathematics and Computation **213**(1), 216–229 (2009)
127. Wang, D.: Editorial: Randomized algorithms for training neural networks. Information Sciences **364** (2016)
128. Werbos, P.J.: Supervised learning: Can it escape its local minimum? In: Theoretical Advances in Neural Computation and Learning, pp. 449–461. Springer (1994)
129. Yu, W., Zhuang, F., He, Q., Shi, Z.: Learning deep representations via extreme learning machines. Neurocomputing **149**, 308–315 (2015)
130. Zhang, J.R., Zhang, J., Lok, T.M., Lyu, M.R.: A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training. Applied mathematics and computation **185**(2), 1026–1037 (2007)
131. Zhigljavsky, A., Žilinskas, A.: Stochastic global optimization, vol. 9. Springer Science & Business Media (2007)