Multilevel optimization strategies for the training of ANNs and application to the solution of PDEs

E. Ricciettti (IRIT-INP, Toulouse)

Joint work with: H. Calandra (Total)
S. Gratton (IRIT-INP, Toulouse)
X. Vasseur (ISAE-SUPAERO, Toulouse)

MATHIAS 2018 - Serris, France - 22-24 October 2018

The problem

We consider optimization problems arising in the training of artificial neural networks:

$$\min_{p} \mathcal{L}(p, z) \qquad z \in \mathcal{T}$$

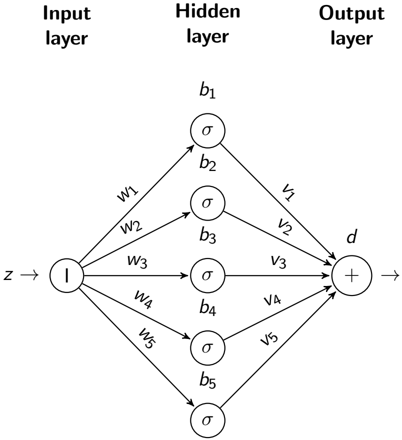
where $\mathcal{L}$ is the loss function, p is the vector of weights and biases of the network, z is the problem's variable and $\mathcal{T}$ is the training set.
The optimization problem may be a large-scale problem.

We look for an efficient scalable optimization method to solve the training problem.

$\Downarrow$

Can we exploit the structure of the network?

Network's architecture



$p = [v, w, b, d]^T$

$\hat{g}(p, z) = \sum_{i=1}^{r} v_i \sigma(w_i z + b_i) + d$

Activation funct. σ,
sigmoid: $\sigma(z) = \frac{e^z - 1}{e^z + 1}$,
tanh: $\sigma(z) = \frac{e^{2z} - 1}{e^{2z} + 1}$,
logit: $\sigma(z) = \frac{e^z}{e^z + 1}$,
softplus: $\sigma(z) = \log(e^z + 1)$.

Outline

- Part I: iterative high-order optimization methods and their multilevel extension
- Part II: use of the multilevel methods for the training of artificial neural network and application to the solution of PDEs

Part I

1. iterative high-order optimization methods
2. multilevel extension

High-order optimization methods

We have a nonlinear problem to solve

$$\min_x f(x)$$

Classical iterative optimization methods:

$$f(x_k + s) \simeq T_q(x_k, s) = f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T \nabla^2 f(x_k) s + \dots$$

with $T_q(x_k, s)$ Taylor model of order q. At each iteration we compute a step s_k to update the iterate:

$$\min_s m_k(x_k, s) = T_q(x_k, s) + \frac{\lambda_k}{q+1} \|s\|^{q+1}, \qquad \lambda_k > 0$$

Higher-order models

Classical choices:

- Least-squares: Levenberg-Marquardt (LM), $q = 1$, $\nabla^2 f(x_k) \sim B_k = J(x_k)^T J(x_k)$.
- Adaptive Cubic Regularization method (ARC), $q = 2$.

Extension to higher-order methods ($q > 2$)

Birgin, Gardenghi, Martnez, Santos, and Toint, 2017

Unifying framework for global convergence is presented.

- better complexity
- model is expensive to minimize

Subproblem solution

Solving

$$\min_s T_q(x_k, s) + \frac{\lambda_k}{q+1}\|s\|^{q+1}$$

represents greatest cost per iteration, which depends on the size of the problem.

$$\Downarrow$$

Multilevel trust region method, Gratton, Sartenaer, Toint, 2008

Hierarchy of problems

- $\{f_l(x_l)\}$, $x_l \in \mathcal{D}_l$
- $|\mathcal{D}_l| < |\mathcal{D}_{l+1}|$
- f_l is cheaper to optimize compared to f_{l+1}

Our contribution

Gratton, Sartenaer, Toint, 2008

- method for second order models
- used just for problems with a geometrical structure

We propose a family of multilevel methods using high-order models that do not need underlying geometry of the problem

Multigrid setting

- At each level l, $x \in \mathbb{R}^{n_l}$. $l_{\max}$ finest level, 0 coarsest level.

level $l_{\max}$	$\mathbb{R}^n$	$x^{l_{\max}}$	$f^{l_{\max}} = f$	$\mu^{l_{\max}} = f$
$\vdots$		$\vdots$	$\vdots$	$\vdots$
level $l+1$	$\mathbb{R}^{n^{l+1}}$	x^{l+1}	f^{l+1}	μ^{l+1}
	$R^{l+1} \Downarrow \quad \Uparrow P^{l+1}$			
level l	$\mathbb{R}^{n^l}$	x^l	f^l	μ^l
$\vdots$		$\vdots$	$\vdots$	$\vdots$
level 0	$\mathbb{R}^{n^0}$	x^0	f^0	μ^0

- f^l represent f on the coarse spaces (it is e.g. the discretization of f on a coarse space)
- The functions μ^l are modifications of the f^l's to ensure inter-level coherence.

Coherence between levels

Lower level model:

- Let $x_0^{l-1} = R x_k^l$. Model with first order correction:

$$\mu^{l-1} = T_q^{l-1}(x_0^{l-1}, s^{l-1}) + (R^l \nabla f^l(x_k^l) - \nabla f^{l-1}(x_k^{l-1}))^T s^{l-1}$$

This ensures that

$$\nabla \mu^{l-1}(x_0^{l-1}) = R^l \nabla f^l(x_k^l)$$

→ first-order behaviours of f^l and μ^{l-1} are coherent in a neighbourhood of the current approximation. If $s^l = P^l s^{l-1}$

$$\nabla f^l(x_k^l)^T s^l = \nabla f^l(x_k^l)^T P^l s^{l-1} = \frac{1}{\alpha} \nabla \mu^{l-1}(x_0^{l-1})^T s^{l-1}.$$

One level strategy

At level $l = 1$, let x_k^l be the current approximation. We look for a correction s_k^l to define the new approximation $x_{k+1}^l = x_k^l + s_k^l$.

x_k^l

One level strategy

At level $l = 1$, let x_k^l be the current approximation. We look for a correction s_k^l to define the new approximation $x_{k+1}^l = x_k^l + s_k^l$.

$$x_k^l \xrightarrow{\qquad T_q^l \qquad} x_{k+1}^l = x_k^l + s_k^l$$

Multilevel strategy

Two choices:

1. minimize regularized Taylor model, get s_k^l,
2. choose lower level model μ^{l-1}:

x_k^l

Multilevel strategy

Two choices:

1. minimize regularized Taylor model, get s_k^l,
2. choose lower level model μ^{l-1}:

$$x_k^l \xrightarrow{\qquad T_q^l \qquad} x_{k+1}^l = x_k^l + s_k^l$$

Multilevel strategy

Two choices:

1. minimize regularized Taylor model, get s_k^l,
2. choose lower level model μ^{l-1}:

$$x_k^l$$

Multilevel strategy

Two choices:

1. minimize regularized Taylor model, get s_k^l,
2. choose lower level model μ^{l-1}:

$$\begin{tikzcd}
x_k^l \arrow[d, "R^l"'] \\
R^l x_k^l := x_0^{l-1}
\end{tikzcd}$$

Multilevel strategy

Two choices:

1. minimize regularized Taylor model, get s_k^l,
2. choose lower level model μ^{l-1}:

$$\begin{tikzcd}
x_k^l \arrow[d, "R^l"'] & \\
R^l x_k^l := x_0^{l-1} \arrow[r, "\mu^{l-1}"] & x_*^{l-1}
\end{tikzcd}$$

Multilevel strategy

Two choices:

1. minimize regularized Taylor model, get s_k^l,
2. choose lower level model μ^{l-1}:

$$\begin{tikzcd}
x_k^l \arrow[d, "R^l"'] & x_{k+1}^l = x_k^l + s_k^l \\
R^l x_k^l := x_0^{l-1} \arrow[r, "\mu^{l-1}"] & x_*^{l-1} \arrow[u, "s_k^l = P^l(x_*^{l-1} - x_0^{l-1})"']
\end{tikzcd}$$

Multilevel strategy

Two choices:

1. minimize regularized Taylor model, get s_k^l,
2. choose lower level model μ^{l-1}:

$$\begin{tikzcd}
x_k^l \arrow[d, "R^l"'] & & x_{k+1}^l = x_k^l + s_k^l \\
R^l x_k^l := x_0^{l-1} \arrow[rr, "\mu^{l-1}"] & & x_*^{l-1} \arrow[u, "s_k^l = P^l(x_*^{l-1} - x_0^{l-1})"']
\end{tikzcd}$$

- The lower level model is cheaper to optimize.
- The procedure is recursive: more levels can be used.

Part II

1. use of the multilevel methods for the training of artificial neural networks
2. application to the solution of PDEs

Exploit multilevel method for training of ANNs

How to build the coarse problem?

The variables to be optimized are the network's weights:
NO evident geometrical structure to exploit!

$$\Downarrow$$

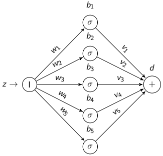
The objective function depends on the output of the network:

$$\min_p \mathcal{L}(p,z) = \mathcal{F}(\hat{g}(p,z)), \qquad z \in \mathcal{T}$$

$$\hat{g}(p,z) = \sum_{i=1}^{r} v_i \sigma(w_i z + b_i) + d$$

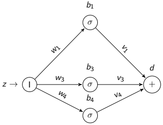
The network possesses a strong hierarchical structure: can we exploit it to build a hierarchy of problems approximating the original one?

Exploit multilevel method for training of ANNs



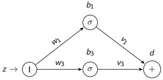
$R_1 \Downarrow P_1 \Uparrow$

$$\mathcal{F}_1 : \mathbb{R}^{3r_1} \to \mathbb{R}$$
$$\hat{g}(p,z) = \sum_{i \in I_1} v_i \sigma(w_i z + b_i) + d$$
$$|I_1| = r_1$$



$R_2 \Downarrow P_2 \Uparrow$

$$\mathcal{F}_2 : \mathbb{R}^{3r_2} \to \mathbb{R}$$
$$\hat{g}(p,z) = \sum_{i \in I_2} v_i \sigma(w_i z + b_i) + d$$
$$I_2 \subset I_1,\ |I_2| = r_2 < r_1$$



$$\mathcal{F}_3 : \mathbb{R}^{3r_3} \to \mathbb{R}$$
$$\hat{g}(p,z) = \sum_{i \in I_3} v_i \sigma(w_i z + b_i) + d$$
$$I_3 \subset I_2,\ |I_3| = r_3 < r_2$$

Build the coarse problems

How do we select the hierarchy of variables?

Algebraic multigrid

We can take inspiration from algebraic multigrid techniques.
When solving linear systems $Ax = b$, the structure is discovered through the matrix A. R and P are built just looking at the entries of the matrix.

Which matrix should we use?

Assume to use a second-order model.
At each iteration we have to solve a linear system of the form:

$$(B_k + \tilde{\lambda}_k I)s = -\nabla f(x_k), \quad \tilde{\lambda}_k > 0.$$

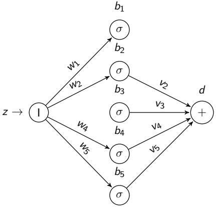
As in AMG for linear systems, we use information contained in matrix B_k.

Which matrix should we use?

Remark

Variables are coupled!

$\{w_i, b_i, v_i\}$



We do not use the full matrix B_k and we define A as:

$$B_k = \begin{bmatrix} f_{v,v} & .. & .. \\ .. & f_{w,w} & .. \\ .. & .. & f_{b,b} \end{bmatrix} \rightarrow A = \frac{f_{v,v}}{\|f_{v,v}\|_\infty} + \frac{f_{w,w}}{\|f_{w,w}\|_\infty} + \frac{f_{b,b}}{\|f_{b,b}\|_\infty}$$

We define the coarse/fine splitting based on the auxiliary matrix A.

Preliminary results: solution of PDEs

Approximate the solution u of a PDE:

$$\begin{aligned} &D(z, u(z)) = g(z), \ z \in (a, b); \\ &u(a) = A, \ u(b) = B. \end{aligned}$$

We approximate $u \sim \hat{u}(p, z)$ for $p \in \mathbb{R}^n$ and we define

$$\mathcal{L}(p, z) = \frac{1}{2t}(\|D(z, u(z)) - g(z)\|^2 + \lambda_p(\|u(a) - A\|^2 + \|u(b) - B\|^2))$$

for $z \in \mathcal{T}$ training set.

Least-squares problem → multi-level Levenberg-Marquardt method

Choice of the true solution

$$D(z, u(z)) = g(z), \ z \in (a, b);$$

- We choose g to have true solution $u_T(z, \nu)$ depending on ν

Remark

- As ν increases the function becomes more oscillatory and it is harder to approximate.
- The size of the problem increases with the number of nodes.
- $\mathcal{T}$: equispaced points in $(0, 1)$ with $h = \frac{1}{3\nu}$ (Shannon's criterion).

Poisson's equation 10 runs

1D		$\nu = 20$	$r = 2^9$		$\nu = 25$	$r = 2^{10}$
Solver	iter	RMSE	save	iter	RMSE	save
LM	869	1.e-4		1439	1.e-3	
MLM	507	1.e-4	1.1-2.6-4.3	1325	1.e-3	1.2-1.7-2.8

Table: 1D Poisson's equation, $u_T(z, \nu) = cos(\nu z)$, 10 runs

2D		$\nu = 5$	$r = 2^{10}$		$\nu = 6$	$r = 2^{11}$
Solver	iter	RMSE	save	iter	RMSE	save
LM	633	1.e-3		1213	1.e-3	
MLM	643	1.e-3	1.1-1.5-2.1	1016	1.e-3	1.2-1.9-2.4

Table: 2D Poisson's equation, $u_T(z, \nu) = cos(\nu z)$, 10 runs

save(min,average,max)=ratio between total number of flops required for matrix-vector products

Helmholtz's equation, 10 runs

		$\nu = 5$	$r = 2^{10}$
Solver	iter	RMSE	save
LM	1159	1.e-3	
MLM	1250	1.e-3	1.2-1.9-3.1

Table: Helmholtz's equations. $\Delta u(z) + \nu^2 u(z) = 0$, $u_T(z, \nu) = sin(\nu z) + cos(\nu z)$

save=ratio between total number of flops required for matrix-vector products

Conclusions and future work

- Theoretical contribution: We have presented a class of multilevel high-order methods for optimization and proved their global convergence and complexity.
- Practical contribution: We have proposed a AMG strategy to build coarse representations of the problem to use some methods in the family for the training of artificial neural networks.
- Future work: Preliminary tests show encouraging results. In future work we will consider more realistic applications.

Thank you for your attention!

For more details:

On high-order multilevel optimization strategies and their application to the training of artificial neural networks

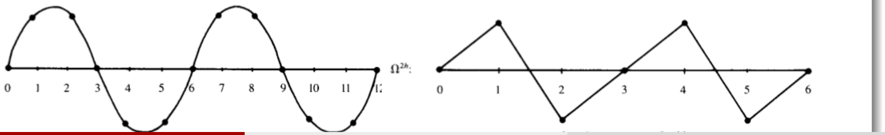
Classical multigrid methods

- Consider a linear elliptic PDE: $D(z, u(z)) = f(z)\ z \in \Omega +$ b.c.
- Discretize on grid h.
- Get a large-scale linear system $A_h x_h = b_h$.

Multigrid methods

Consider the discretization of the same PDE problem on a coarser grid: $A_H x_H = b_H$, $H > h$.

- Relaxation methods fails to eliminate smooth components of the error efficiently.
- Smooth components projected on a coarser grid appear more oscillatory.



Coarse problem construction

Define transfer grid operators: P prolongation and R restriction to project vectors from a grid to another: $x_H = Rx_h$, $x_h = Px_H$, such that $R = \alpha P^T$.

Geometry exploitation

The geometrical structure of the problem is exploited to build R and P.

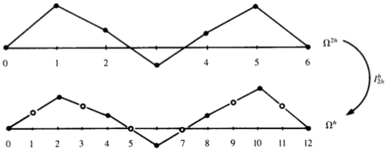


Figure 3.2: *Interpolation of a vector on coarse grid Ω^{2h} to fine grid Ω^h.*

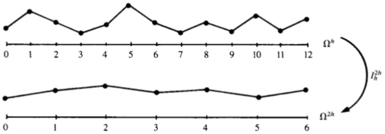


Figure 3.4: *Restriction by full weighting of a fine-grid vector to the coarse grid.*

Remark

This strategy is also available in the nonlinear case (Full Approximation Scheme (FAS) algorithm).

Basic iterative optimization algorithm

Until convergence

- Define the local model m_k of f around x_k, depending on λ_k
- Compute a trial point $x_k + s_k$ that decreases this model
- Compute the predicted reduction $m_k(x_k) - m_k(x_k + s_k)$
- Evaluate change in the objective function $f(x_k) - f(x_k + s_k)$
- If achieved change $\sim$ predicted reduction then
    - Accept trial point as new iterate $x_{k+1} = x_k + s_k$

  else

    - Reject the trial point $x_{k+1} = x_k$
    - Increase λ_k

Multilevel strategy

At level l, let x_k^l be the current approximation. We look for a correction s_k^l to define the new approximation $x_{k+1}^l = x_k^l + s_k^l$. Two choices:

1. minimize regularized Taylor model, get s_k^l,
2. choose lower level model μ^{l-1}:

$$\begin{tikzcd}
x_k^l \arrow[d, "R^l"'] & x_{k+1}^l = x_k^l + s_k^l \\
R^l x_k^l := x_0^{l-1} \arrow[r, "\mu^{l-1}"] & x_*^{l-1} \arrow[u, "s_k^l = P^l(x_*^{l-1} - x_0^{l-1})"']
\end{tikzcd}$$

Recursive multi-scale q-order methods

Until convergence

- Choose $q \geq 1$. Choose either a Taylor or a (useful) recursive model.
  - Taylor model: compute a Taylor step satisfying a sufficient decrease property
  - Recursive: apply the algorithm recursively
- Evaluate change in the objective function
- If achieved change $\sim$ predicted reduction then
  - Accept trial point as new iterate

  else

  - Reject the trial point
  - Increase λ

The algorithm is proved globally convergent to first order critical points

Ruge and Stueben AMG

To build the coarse problem, the variables are divided into two sets, set C of coarse variables and set F of fine variables.

Ruge and Stueben C/F splitting

- Two variables i, j are said to be *coupled* if $a_{i,j} \neq 0$.
- We say that a variable i is strongly coupled to another variable j, if

$$-a_{i,j} \geq \epsilon \max_{a_{i,k}<0} |a_{i,k}|$$

  for a fixed $0 < \epsilon < 1$, usually $\epsilon = 0.25$.
- Each F variable is required to have a minimum number of its strong couplings be represented in C. The C/F splitting is usually made choosing some first variable i to become a coarse variable. Then, all variables strongly coupled to it become F variables. The process is repeated until all variables have been split.