Optimization strategies to deal with large-scale problems: an opportunity for mixed precision?

Elisa Riccietti

ENS Lyon

Seminaire Taran, 08/03/2022.

Context

$$\min_x f(x) \quad \rightarrow \quad \min_{x \in \mathbb{R}^n} f(x) = \sum_{i=1}^{m} f_i(x)$$

Large scale problems

- f is the sum of a large number of functions: large m
- f depends on a large number of variables: large n

Examples from machine learning

Large m: large datasets

$\mathcal{D} = \{(z_1, y_1), \dots, (z_m, y_m)\}$, want to predict the hidden relation ϕ that relates z to y

Look for a model $m(x)$ such that $m(x; z) \sim \phi(y)$ for all couples (z, y).

Given a loss function ℓ, we define

$$f(x) = \sum_{i=1}^{m} \ell(m(x, z_i) - y_i) = \sum_{i=1}^{m} f_i(x)$$

Large n: deep neural networks

$m(x)$ is a deep neural network, n=#edges + # neurons

Context

$$\min_{x} f(x) \quad \rightarrow \quad \min_{x\in\mathbb{R}^n} f(x) = \sum_{i=1}^{m} f_i(x)$$

Large scale problems

- f is the sum of a large number of functions: large m (ex: classification of large data sets)
- f depends on a large number of variables: large n (ex: deep learning)

Common objective

Exploit approximations of the objective function to reduce the computational cost of the solution

Context

$$\min_{x} f(x) \quad \rightarrow \quad \min_{x \in \mathbb{R}^n} f(x) = \sum_{i=1}^{m} f_i(x)$$

Large scale problems

- f is the sum of a large number of functions: large m (ex: classification of large data sets) $\rightarrow$ **subsampled methods**
- f depends on a large number of variables: large n (ex: deep learning) $\rightarrow$ **multilevel methods**

Common objective

Exploit approximations of the objective function to reduce the computational cost of the solution

Outline

- Background material: introduction to trust-region methods.
- I part: Subsampled methods
- II part: Multilevel methods
- Opportunities for mixed precision?

Optimization methods

The solution is approximated by a sequence x_k converging to a stationary point x^* such that $\nabla f(x^*) = 0$.

First order

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k),$$

where α_k is the step length (*learning rate*).

- ☺ Low computational cost and memory consumption
- ☹ Better suited for convex problems, dependent on the choice of α_k, slow convergence

Second order

$$x_{k+1} = x_k - H(x_k)^{-1} \nabla f(x_k)$$

where H is the Hessian matrix.

- ☹ Need for linear systems solution, high computational cost and memory consumption
- ☺ Efficient on nonconvex problems, robust, fast convergence

Newton's method

It builds $\{x_k\}$ such that $x_{k+1} = x_k + p_k$ where p_k is the solution of:

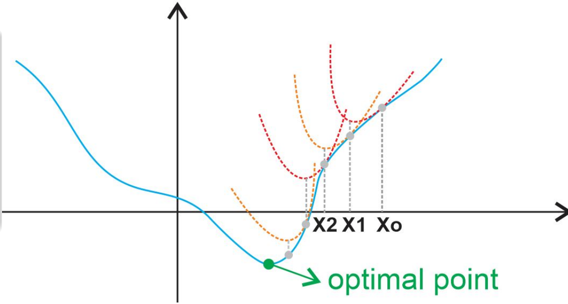
$$\min_{p\in\mathbb{R}^n} m_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T H(x_k) p$$

where H is the Hessian of f.

Remark

p_k is the solution of

$$H(x_k)p_k = -\nabla f(x_k)$$



Properties of Newton's method

Remark

- Very fast convergence: requires few iterations to reach the solution
- Each iteration is expensive
- No global convergence: convergence not guaranteed for any initial guess

Newton trust-region method

Globally convergent improvement over Newton's method: restrict the minimization to a ball (the trust region)

$$\min_p m_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T H(x_k) p$$
$$\text{s.t. } \|p\| \leq \Delta_k$$

where H is the Hessian of f.

Remark

p_k is the solution of

$$(H(x_k) + \lambda_k I) p_k = -\nabla f(x_k)$$

with λ_k implicitly defined by the trust region radius Δ_k

Trust-region methods

- Given x_k and the trust-region radius $\Delta_k > 0$ find the step p_k solving

$$\min_p m_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T H(x_k) p,$$
$$\text{s.t. } \|p\| \leq \Delta_k$$

- Compute

$$\rho_k(p_k) = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)}.$$

- Step acceptance and trust-region radius update. Given $\eta \in (0, 1)$:
  - If $\rho_k < \eta$ then set $\Delta_{k+1} < \Delta_k$ and $x_{k+1} = x_k$.
  - If $\rho_k \geq \eta$ then set $\Delta_{k+1} \geq \Delta_k$ and $x_{k+1} = x_k + p_k$.

Part I: Large datasets

Collaboration with: Stefania Bellavia, University of Florence
S. Gratton, INP-ENSEEIHT, Toulouse

Large datasets

Subsampling techniques

- Large set of data at disposal: $\{1, \dots, N\}$.
  Subsampling: $X_k \subseteq \{1, \dots, N\}$.
- Sequence of approximations $\{f_{\delta_k}\}$ of the original objective function

$$f_{\delta_k}(x) = \sum_{i \in X_k} f_i(x) \sim f(x)$$

- δ_k is the accuracy level of the approximations:

$$|f_{\delta_k}(x_k) - f(x_k)| \leq \delta_k.$$

- We assume that the accuracy level can be changed along the optimization process

Inexact trust-region method

- Approximated model:

$$m_k(p_k) = f_{\delta_k}(x) + \nabla f_{\delta_k}(x)^T p + \frac{1}{2} p^T H_{\delta_k}(x_k) p.$$

- At each iteration the step is found minimizing the noisy model, i.e. solving a linear systems of the form:

$$(H_{\delta_k}(x_k) + \lambda_k I) p_k = -\nabla f_{\delta_k}(x_k)$$

Step acceptance

- After the step is computed, we have to decide whether to accept the step.
- Step acceptance is based on the ratio:

$$\rho_k^{\delta_k}(p_k) = \frac{f_{\delta_k}(x_k) - f_{\delta_k}(x_k + p_k)}{m_k(0) - m_k(p_k)}.$$

- If the noise is too high, the reduction in f_{δ_k} can be just an effect of the presence of the noise.

Step acceptance

- After the step is computed, we have to decide whether to accept the step.
- Step acceptance is based on the ratio:

$$\rho_k^{\delta_k}(p_k) = \frac{f_{\delta_k}(x_k) - f_{\delta_k}(x_k + p_k)}{m_k(0) - m_k(p_k)}.$$

- If the noise is too high, the reduction in f_{δ_k} can be just an effect of the presence of the noise.

Need for a strategy to control the noise!!

Noise control

Noise control

Let

$$|f_{\delta_k}(x_k) - f(x_k)| \leq \frac{\eta}{2}[m_k(0) - m_k(p_k)].$$

If

$$\rho_k^{\delta_k}(p_k) = \frac{f_{\delta_k}(x_k) - f_{\delta_k}(x_k + p_k)}{m_k(0) - m_k(p_k)} > \eta$$

then also

$$\rho_k(p_k) = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)} > \eta.$$

$\rightarrow$ True reduction in the noise-free objective function f

Algorithm : k-th iteration of the subsumpled trust-region method

Given δ_0.

For $k = 0, 1, 2, \ldots$

1. Compute a solution p_k of the TR subproblem.
2. Estimate $\delta_k = |f_{\delta_k}(x_k) - f(x_k)|$
3. If $\delta_k \leq \frac{\eta}{2}[m_k(0) - m_k(p_k)]$, continue with the classic TR update
4. Else, else reduce δ_k by providing a new approximation to f and go back to 1.

Numerical results

- **Data assimilation problem**. We look for the initial state x of a system, from the knowledge of observations y_j, $t_j > 0$:

$$\min_{x \in \mathbb{R}^n} \frac{1}{2}\|x - x_b\|^2_{B^{-1}} + \frac{1}{2}\sum_{j=0}^{N_t} \|H_j(x(t_j)) - y_j\|^2_{R_j^{-1}}$$

- **Machine learning problem**. Binary classification problem: $\{(z^i, y^i)\}$ with $z^i \in \mathbb{R}^n$, $y^i \in \{-1, +1\}$ and $i = 1, \dots, N$.
  Training objective function: logistic loss with l_2 regularization

$$f(x) = \frac{1}{2N}\sum_{i=1}^{N} \log(1 + \exp(-y^i x^T z^i)) + \frac{1}{2N}\|x\|^2.$$

	Data Assimilation		Machine learning	
	All samples	Subsampled	All samples	Subsampled
it	9	12	52	38
cost_f	10	3	53	16
cost_p	67	15	808	316
RMSE	1.2e-2	3.8e-2	5.4e-2	6.0e-2
save_f		67%		70%
save_p		78%		61%

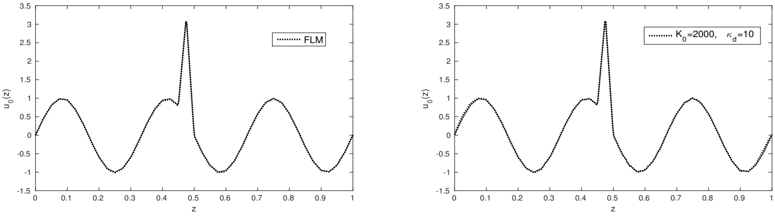


Figure: Solution approximation, Left: all samples, Right: Subsampled

Perspectives

Can this be extended to mixed precision?

- Theoretical results: we recover "good" convergence results with the assumption that $\delta_k \to 0$, while in mixed precision we have a discrete setting

Perspectives

Can this be extended to mixed precision?

- Theoretical results: we recover "good" convergence results with the assumption that $\delta_k \to 0$, while in mixed precision we have a discrete setting
- First attempt: *A note on solving nonlinear optimization problems in variable precision*, S. Gratton, Ph. L. Toint, COAP, 2019

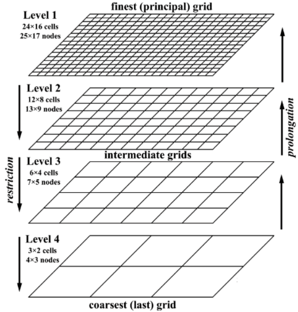
  Idea: Select $\delta_k^+ \leq \frac{\eta}{2}[m_k(0) - m_k(p_k)]$, if $\delta_k^+ < \delta_k$, improve the approximation
  - They consider second order methods: what about gradient method?
  - They don't allow inexactness in the hessian
  - Not tested on neural networks

Part II: high-dimensional problems

Collaboration with: , H.Calandra, Total, S. Gratton, INP-ENSEEIHT, Toulouse, X.Vasseur, ISAE-SUPAERO, Toulouse

Idea for dimensionality reduction: multigrid methods for PDEs

State-of-the-art methods for PDEs: exploit representation of the problem at different scales



- Fine scales: eliminate high frequency components of the error
- Coarse scales: eliminate low frequency components of the error

Transposition of this idea to optimization: multilevel methods

Hierarchy of problems

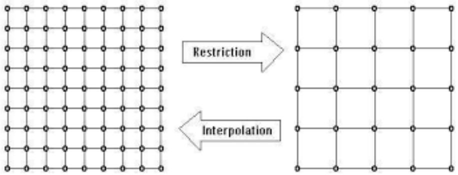
- $\{f_\ell(x_\ell)\}$, $x_\ell \in \mathbb{R}^{n_\ell}$
- $n_{\ell-1} < n_\ell \rightarrow f_{\ell-1}$ less expensive to optimize than f_ℓ

Example

Solve

$\Delta u = g \rightarrow \min_x f(x) = \|\Delta x - g\|^2$.

- Discretize over a fine grid: $\{f_1(x_1)\}$, $x_1 \in \mathbb{R}^{81}$
- Discretize over a coarse grid: $\{f_2(x_2)\}$, $x_2 \in \mathbb{R}^{25}$



Transposition of this idea to optimization: multilevel methods

How to exploit the lower levels?

Need to update $x_{k+1} = x_k + p_k$

Look for p_k in the lower dimensional space and project back!

x_k^ℓ

Transposition of this idea to optimization: multilevel methods

How to exploit the lower levels?

Need to update $x_{k+1} = x_k + p_k$
Look for p_k in the lower dimensional space and project back!

$$
\begin{array}{c}
x_k^\ell \\
\Big\downarrow R_\ell \\
x_{0,k}^{\ell-1} := R_\ell x_k^\ell
\end{array}
$$

Transposition of this idea to optimization: multilevel methods

How to exploit the lower levels?

Need to update $x_{k+1} = x_k + p_k$
Look for p_k in the lower dimensional space and project back!

$$\begin{tikzcd} x_k^\ell \arrow[d, "R_\ell"'] & \\ x_{0,k}^{\ell-1} := R_\ell x_k^\ell \arrow[r, "\min_x f_{\ell-1}(x)"] & x_{*,k}^{\ell-1} \end{tikzcd}$$

Transposition of this idea to optimization: multilevel methods

How to exploit the lower levels?

Need to update $x_{k+1} = x_k + p_k$

Look for p_k in the lower dimensional space and project back!

$$
\begin{tikzcd}
x_k^\ell \arrow[d, "R_\ell"'] & x_{k+1}^\ell = x_k^\ell + p_k^\ell \\
x_{0,k}^{\ell-1} := R_\ell x_k^\ell \arrow[r, "\min_x f_{\ell-1}(x)"] & x_{*,k}^{\ell-1} \arrow[u, "p_k^\ell = P_\ell(x_{*,k}^{\ell-1} - x_{0,k}^{\ell-1})"']
\end{tikzcd}
$$

Transposition of this idea to optimization: multilevel methods

How to exploit the lower levels?

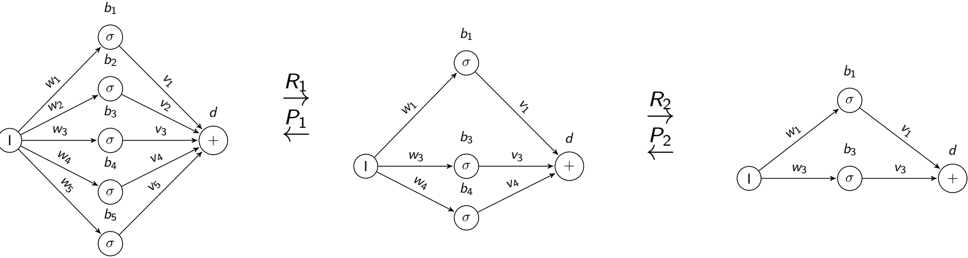
Need to update $x_{k+1} = x_k + p_k$
Look for p_k in the lower dimensional space and project back!

$$\begin{tikzcd} x_k^\ell \arrow[d, "R_\ell"'] & & x_{k+1}^\ell = x_k^\ell + p_k^\ell \\ x_{0,k}^{\ell-1} := R_\ell x_k^\ell \arrow[rr, "\min_x f_{\ell-1}(x)"] & & x_{*,k}^{\ell-1} \arrow[u, "p_k^\ell = P_\ell(x_{*,k}^{\ell-1} - x_{0,k}^{\ell-1})"'] \end{tikzcd}$$

- If $\|R_\ell \nabla f_\ell\| > \kappa \|\nabla f_{\ell-1}\|$ is go to go down !

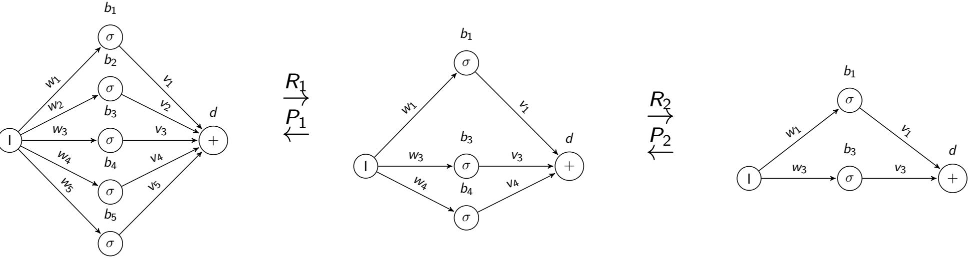
$$f_{\ell-1}(x_{*,k}^{\ell-1}) < f_{\ell-1}(x_{0,k}^{\ell-1}) \rightarrow f_\ell(x_k^\ell) < f_\ell(x_k^\ell + p_k^\ell)$$

Multilevel training methods for DNN



- Networks are algebraic objects, no geometry $\Rightarrow$ how to chose the hierarchy?

Multilevel training methods for DNN



- Networks are algebraic objects, no geometry $\Rightarrow$ how to chose the hierarchy?
- We use a algebraic multigrid (AMG) strategy from PDEs on the Hessian

$$D(z, u(z)) = g_{\nu}(z),\ z \in \Omega \subset \mathbb{R}^2, \qquad u(z) = f_{\nu}(z)\ z \in \partial\Omega$$

2D Helmholtz's equation

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

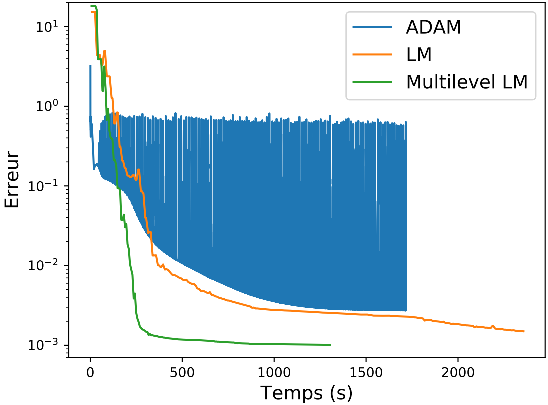
Nonlinear equations

	$\nu = 20$		$r = 2^9$	$\nu = 1$		$r = 2^9$
Method	iter	RMSE	save	iter	RMSE	save
TR	950	10^{-5}		270	10^{-3}	
MTR	1444	10^{-5}	0.8-2.9-5.3	320	10^{-3}	1.2-2.3-3.1

Numerical results

Poisson's equation (2D, $n = 4096$).

Method	ADAM	1 level	2 levels
Iterations	10000	200	200



Perspectives: can we use mixed precision?

- Build hierarchy based on precision levels
- Provides an automatic rule to switch the precision
- How to design the transfer operators?

Thank you for your attention!