Gradient-based optimization methods for neural network training

Ph.D. in Industrial Engineering
Università di Firenze, A.A. 2020/2021

Simone Rebegoldi
Dipartimento di Ingegneria Industriale
Università di Firenze

UNIVERSITÀ DEGLI STUDI FIRENZE

Seminar content

- Overview on models arising in machine learning[1] ✓
- Analysis of stochastic gradient methods[2,3] ✓
- Noise-reduction techniques
- Methods with adaptive choice of the learning-rate
- Application to Artificial Neural Networks

[1] Goodfellow, Bengio, Courville, *Deep Learning*, MIT Press, 2016, `http://www.deeplearningbook.org`.

[2] Bottou, Curtis, Nocedal. *Optimization Methods for Large-Scale Machine Learning*, Siam Review 60 (2), 223–311, 2018.

[3] Bianconcini, Bellavia, Krejic, Morini, *Subsampled first-order optimization methods with applications in imaging*, Handbook of Mathematical Models and Algorithms in Computer Vision and Imaging, 2021.

Finite-sum minimization problems in machine learning

Training a neural network involves solving a problem of the form:

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{N} \sum_{i=1}^{N} \phi_i(x)$$

where $\phi_i : \mathbb{R}^n \to \mathbb{R}$, $\phi_i \in C^1(\mathbb{R}^n)$, N is large.

- Compute $\hat{x}$ s.t. $\|\nabla f(\hat{x})\| \leqslant \epsilon$ using iterative gradient-based methods
- Training sets often show redundancy in the data
  $\Rightarrow$ using all the sample data in every optimization iteration is inefficient
- Idea: gradient subsampling

First order (stochastic) gradient methods

Gradient method (GD)

Compute

$$x_{k+1} = x_k - \frac{\alpha}{N} \sum_{i=1}^{N} \nabla \phi_i(x_k), \quad k = 0, 1, 2, \dots$$

Stochastic Gradient method (SG)

Choose randomly and uniformly the index $i_k \in \{1, \dots, N\}$ and compute

$$x_{k+1} = x_k - \alpha_k \nabla \phi_{i_k}(x_k), \quad k = 0, 1, 2, \dots$$

Mini-batch Stochastic gradient (Mini-batch SG)

Choose randomly and uniformly the sample $\mathcal{S}_k \subseteq \{1, \dots, N\}$, let $N_k = card(\mathcal{S}_k)$ be the sample size and compute

$$x_{k+1} = x_k - \frac{\alpha_k}{N_k} \sum_{i \in \mathcal{S}_k} \nabla \phi_i(x_k), \quad k = 0, 1, 2, \dots$$

Stochastic optimization methods

Algorithm : Stochastic Gradient Methods

```
1. Choose $x_0 \in \mathbb{R}^n$
2. For $k = 0, 1, \ldots$ do
    2.1 Generate a realization of the random variable $\xi_k$.
    2.2 Compute a stochastic gradient $g(x_k, \xi_k)$.
    2.3 Choose a stepsize $\alpha_k > 0$.
    2.4 Set $x_{k+1} = x_k - \alpha_k g(x_k, \xi_k)$.
```

SG analysis: strongly convex f, constant stepsize α

- Suppose that the stochastic gradient $g(x_k, \xi_k)$ satisfies

$$\begin{aligned} \nabla f(x_k)^T \mathbb{E}_{\xi_k}\left[g(x_k,\xi_k)\right] &\geqslant \mu \|\nabla f(x_k)\|_2^2 \\ \mathbb{E}_{\xi_k}\left[\|g(x_k,\xi_k)\|_2^2\right] &\leqslant M_1 + M_2 \|\nabla f(x_k)\|_2^2, \quad M_2 \geqslant \mu^2. \end{aligned}$$

  The first property implies that $g(x_k, \xi_k)$ is a descent direction in expectation.
- SG: if f is strongly convex and $\alpha < \frac{\mu}{LM_2}$ (sufficiently small stepsize), then

$$\mathbb{E}[f(x_k) - f(x_*)] \leqslant \frac{\bar{\alpha} L M_1}{2c\mu} + (1 - \bar{\alpha} c \mu)^k \left(f(x_0) - f(x_*) - \frac{\bar{\alpha} L M_1}{2c\mu} \right)$$

- Mini-batch SG with $N_k = N_{mb}$: if f is strongly convex and α is sufficiently small, then

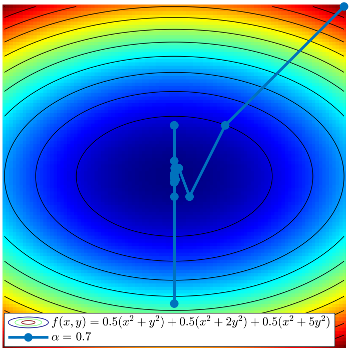
$$\mathbb{E}[f(x_k) - f(x_*)] \leqslant \underbrace{\frac{\bar{\alpha} L M_1}{2c\mu N_{\mathrm{mb}}}}_{\text{smaller asymptotic gap}} + (1 - \bar{\alpha} c \mu)^k \left(f(x_0) - f(x_*) - \frac{\bar{\alpha} L M_1}{2c\mu N_{\mathrm{mb}}} \right)$$

  However the computation of $g(x_k, w_k)$ is N_{mb} times more expensive than in SG.

Issue of SG (1): effect of noise

SG suffers from the effect of noisy gradient estimates.

Example with $g(x_k, \xi_k) = \nabla \phi_{i_k}(x_k)$



Noise reduction

Observe that condition

$$\mathbb{E}_{\xi_k}\left[\|g(x_k,\xi_k)\|_2^2\right] \leqslant M_1 + M_2\|\nabla f(x_k)\|_2^2, \quad M_2 \geqslant \mu^2$$

is implied by the following two properties:

1. $\|\mathbb{E}_{\xi_k}[g(x_k,\xi_k)]\| \leqslant \mu_G\|\nabla f(x_k)\|_2$
2. $\mathbb{V}_{\xi_k}[g(x_k,\xi_k)] \leqslant M_1 + M_G\|\nabla f(x_k)\|_2^2$, with $M_2 = M_G + \mu_G^2$ where $\mathbb{V}_{\xi_k}[g(x_k,\xi_k)]$ is the variance of $g(x_k,\xi_k)$:

$$\mathbb{V}_{\xi_k}[g(x_k,\xi_k)] = \mathbb{E}_{\xi_k}\left[\|g(x_k,\xi_k)\|_2^2\right] - \|\mathbb{E}_{\xi_k}[g(x_k,\xi_k)]\|^2.$$

Noise reduction

Note that the descent lemma and the independence of x_k from ξ_k implies:

$$\mathbb{E}_{\xi_k}[f(x_{k+1})] - f(x_k) \leqslant -\alpha_k \nabla f(x_k)^T \mathbb{E}_{\xi_k}[g(x_k,\xi_k)] + \frac{1}{2}\alpha_k^2 L\, \mathbb{E}_{\xi_k}[\|g(x_k,\xi_k)\|_2^2].$$

If also $\mathbb{E}_{\xi_k}[g(x_k,\xi_k)] = \nabla f(x_k)$ (unbiased estimator), then

$$\mathbb{E}_{\xi_k}[f(x_{k+1})] - f(x_k) \leqslant -\alpha_k \underbrace{\|\nabla f(x_k)\|^2}_{\leqslant 0} + \frac{1}{2}\alpha_k^2 L\, \mathbb{E}_{\xi_k}[\|g(x_k,\xi_k)\|_2^2].$$

- Since $\mathbb{E}_{\xi_k}\left[\|g(x_k,\xi_k)\|_2^2\right] = \|\nabla f(x_k)\|_2^2 + \mathbb{V}_{\xi_k}\left[g(x_k,\xi_k)\right]$, the variance $\mathbb{V}_{\xi_k}\left[g(x_k,\xi_k)\right]$ controls the convergence rate.
- If the variance decreases fast enough along with $\|\nabla f(x_k)\|_2^2$, then the effect of having noisy directions will not impede a “fast” rate of convergence.
- Possible approach: design procedures such that the variance decreases along the iterations.

Noise reduction

Classes of methods

- *Dynamic sample-sized methods*

$$N_k = card(\mathcal{S}_k) = \lceil \tau^{k-1} \rceil, \quad \text{for some } \tau > 1.$$

The variance decrease as $O(\frac{1}{N_k}) = O(\tau^{1-k})$
$\Rightarrow k = O(\epsilon^{-1})$ with constant sufficiently small stepsize

- *Gradient aggregation methods*
Rather than increasing the sample size at each iteration, reuse previously computed gradient estimations

SVRG, Johnson and Zhang, NIPS 2013
SAGA, Defazio, Bach, Lacoste-Julien, Advances in Neural Information Processing Systems, 2014
SARAH Nguyen, Liu, Scheinberg, Takac, Proceedings of the 34th International Conference on Machine Learning, 2017

⋮

Stochastic Variance Reduced Gradient (SVRG)

Main idea:

- Use the full gradient every m iterations (Outer loop, indexed by k - GD iteration)
- In the inner loop, indexed by t, use SGD equipped with the following gradient estimator:

$$\tilde{g}_t = \nabla \phi_{i_t}(\tilde{x}_t) + \underbrace{\nabla f(\tilde{x}_0) - \nabla \phi_{i_t}(\tilde{x}_0)}_{\text{bias in the gradient estimate } \nabla \phi_{i_t}(\tilde{x}_0)}$$

- $\tilde{g}_t$ is an unbiased estimator of $\nabla f(\tilde{x}_t)$; smaller variance than $\nabla \phi_{i_t}(\tilde{x}_t)$ is expected;

Implementation issues:

- The length of the inner cycle m and the stepsize α need by chosen by trial and error.
- To reduce the memory requirements instead of storing all gradients $\nabla \phi_i(\tilde{x}_0)$ separately, at each inner iteration $\nabla \phi_{i_t}(\tilde{x}_0)$ is evaluated along with $\nabla \phi_{i_t}(\tilde{x}_t)$.
- When the full gradient evaluation is too expensive, a mini-batch stochastic gradient is used.

SVRG: the algorithm

Hyperparameters: α, m number of steps in the inner loop

```
Step 0: Initialization.
Choose an initial point $x_0 \in \mathbb{R}^n$, an inner loop size $m > 0$, a steplength $\alpha > 0$,
the option for the iterate update. Set $k = 1$.
Step 1: Outer iteration, full gradient evaluation.
Set $\tilde{x}_0 = x_{k-1}$. Compute $\nabla f(\tilde{x}_0)$.
Step 2: Inner iterations
For $t = 0, \ldots, m-1$
        Uniformly and randomly choose $i_t \in \{1, \ldots, N\}$.
        Set $\tilde{x}_{t+1} = \tilde{x}_t - \alpha(\nabla \phi_{i_t}(\tilde{x}_t) - \nabla \phi_{i_t}(\tilde{x}_0) + \nabla f(\tilde{x}_0))$.
Step 3: Outer iteration, iterate update.
Set $x_k = \tilde{x}_m$ (Option I).
Set $x_k = \tilde{x}_t$ for randomly uniformly chosen $t \in \{0, \ldots, m-1\}$ (Option II).
Increment $k$ by one and go to Step 1.
```

SVRG for strongly convex f

For small enough stepzises, converges at linear rate

Theorem -option II

Suppose that all ϕ_i are convex and SVRG-option II is run with a fixed stepsize $\alpha_k = \bar{\alpha}, \forall k$ and m s.t.

$$\rho = \frac{1}{c\bar{\alpha}(1-L\bar{\alpha})m} + \frac{L\bar{\alpha}}{1-L\bar{\alpha}} < 1,$$

then

$$\mathbb{E}[f(x_k)] - f(x_*) \leqslant \rho^k (f(x_0) - f(x_*)).$$

- Choosing $m > L/c$ and $\alpha \lessapprox 1/L$ yields $0 < \rho < 1/2$
  $\Rightarrow \log(\epsilon^{-1})$ outer iterations to get $\mathbb{E}[f(x_k)] - f(x_*) < \epsilon$
- $1 + m/N$ full gradient evaluations at each outer iteration

SVRG for strongly convex f

Theorem -option I

Suppose that SVRG-option I is run with a fixed stepsize $\alpha_k = \bar{\alpha}, \forall k$ and m s.t.

$$\rho = (1 - 2\bar{\alpha}c(1-\bar{\alpha}L)^m) + \frac{\bar{\alpha}L^2}{c(1-\bar{\alpha}L)} < 1,$$

then

$$\mathbb{E}[\|x_k - x_*\|^2] \leqslant \rho^k \|x_0 - x_*\|^2$$

ADAptive Moment estimation (ADAM) Kingma e Jimmy Ba, 2014

1. ADAM does not require full gradient computation,
2. It computes individual adaptive steplengths for updating the iterate component-wise on the basis of the current first and second order momentum of the stochastic gradient

```
**Hyperparameters:** $\{\alpha\}$; $\beta_1, \beta_2 \in [0, 1)$ exponential decay rates for the moment estimates;
$\epsilon \ll 1$; $N_{mb}$;
**Step 0: Initialization.** Choose an initial point $x_0 \in \mathbb{R}^n$, set $\boldsymbol{v}_0 = 0 \in \mathbb{R}^n$, $q_0 = 0 \in \mathbb{R}^n$
**Step 1** for $k = 0, 1, \ldots$ do
    Choose $\mathcal{S}_k \subseteq \{1, \ldots, n\}$ con $|\mathcal{S}_k| = N_{mb}$
    Set $\boldsymbol{g}_k = \frac{1}{N_{mb}} \sum_{i \in \mathcal{S}_k} \nabla \phi_i(x_k)$
    Set $\boldsymbol{v}_k \leftarrow \beta_1 \cdot \boldsymbol{v}_{k-1} + (1 - \beta_1) \cdot g_k$          *Update biased first moment estimate*
    Set $\boldsymbol{q}_k \leftarrow \beta_2 \cdot \boldsymbol{q}_{k-1} + (1 - \beta_2)\ g_k \odot \boldsymbol{g}_k$        *Update biased second moment estimate*
    Set $\hat{\boldsymbol{v}}_k \leftarrow \boldsymbol{v}_k / (1 - \beta_1^k)$          *Compute bias-corrected first moment estimate*
    Set $\hat{\boldsymbol{q}}_k \leftarrow \boldsymbol{q}_k / (1 - \beta_2^k)$          *Compute bias-corrected second moment estimate*
    Set $x_{k+1} \leftarrow \boldsymbol{x}_k - \alpha \cdot \hat{\boldsymbol{v}}_k ./ (\sqrt{\hat{\boldsymbol{q}}_k} + \boldsymbol{\epsilon})$
```

$g_k \odot \boldsymbol{g}_k$ denotes the vector $((g_k)_1^2, (g_k)_2^2, \ldots, (g_k)_n^2)^T$ and ./ denotes the componentwise division

ADAM: some comments

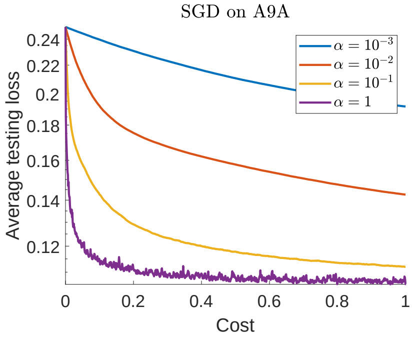
- $\boldsymbol{v}_k = (1-\beta_1)\sum_{i=1}^{k} \beta_1^{k-i}\ \boldsymbol{g}_i$ small weights to gradients too far in the past ;
- $\boldsymbol{q}_k = (1-\beta_2)\sum_{i=1}^{k} \beta_2^{k-i}\ \boldsymbol{g}_i \odot \boldsymbol{g}_i$.
- Divide by $(1-\beta_2^k)$ to correct the initialization bias

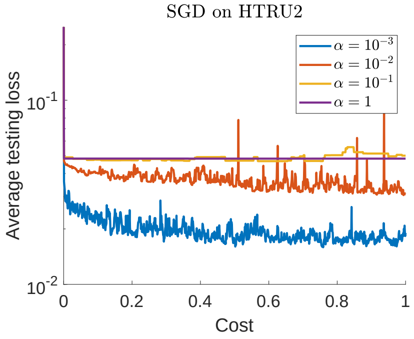
$$
\begin{aligned}
\mathbb{E}[\boldsymbol{q}_k] &= \mathbb{E}\left[(1-\beta_2)\sum_{i=1}^{k}\beta_2^{k-i}\ \boldsymbol{g}_i\odot\boldsymbol{g}_i\right] = (1-\beta_2)\sum_{i=1}^{k}\beta_2^{k-i}\,\mathbb{E}\left[\boldsymbol{g}_i\odot\boldsymbol{g}_i\right] \\
&= (1-\beta_2)\,\mathbb{E}\left[\boldsymbol{g}_k\odot\boldsymbol{g}_k\right]\sum_{i=1}^{k}\beta_2^{k-i} + \underbrace{(1-\beta_2)\sum_{i=1}^{k}\beta_2^{k-i}\Big(\mathbb{E}\left[\boldsymbol{g}_i\odot\boldsymbol{g}_i\right]-\mathbb{E}\left[\boldsymbol{g}_k\odot\boldsymbol{g}_k\right]\Big)}_{\zeta\text{can be kept small}} \\
&= \mathbb{E}\left[\boldsymbol{g}_k\odot\boldsymbol{g}_k\right](1-\beta_2)\sum_{i=1}^{k}\beta_2^{k-i} + \zeta = \mathbb{E}\left[\boldsymbol{g}_k\odot\boldsymbol{g}_k\right](1-\beta_2^k) + \zeta
\end{aligned}
$$

- Adaptive learning rate: The step used to update the iterate is $\alpha\hat{\boldsymbol{v}}_k./\sqrt{\hat{\boldsymbol{q}}_k}$. Components corresponding to small ratios $\hat{\boldsymbol{v}}_k./\sqrt{\hat{\boldsymbol{q}}_k}$ (estimates of first and second order moment do not agree).
- Effective and widely used, but still lacking of well-assessed convergence analysis, at least to our knowledge.

Issue of SG (2): tuning of the learning rate

The choice of the stepsize/learning rate is problem-dependent.
SG with constant stepsize for strongly convex f requires $\alpha < \mu/(LM_2)$
$\Rightarrow$ If L is not known, the stepsize α needs to be tuned!

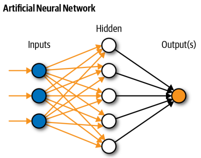




Binary classification on the datasets A9A and HTRU2.

Issue of SG (2): tuning of the learning rate

- For large-scale applications, tuning the parameters may require weeks or months of effort on a supercomputer before the algorithm performs well.
- H. Asi, J.C. Duchi, *The importance of better models in stochastic optimization*, Proc. Natl. Acad. Sci. USA, 2019
  "In searching for optimal neural network architectures and hyperparameters, the paper ... uses approximately 750000 CPU days in its parameter search ... Assuming standard CPU energy use of between 60-100 Watts, the energy is roughly between 4 *and* $6 \cdot 10^{12}$ *Joules. At* 10^9 *Joules per tank of gas, this is sufficient to drive* 4000 *Toyota Camrys the* 380 *miles between San Francisco and Los Angeles."*



Adaptive choice of the learning rate

SG and its variants employ stochastic (possibly and occasionally full) gradient estimates and do not rely on any machinery from standard globally convergent optimization procedures, such as linesearch or trust-region.

On the other hand, a few recent papers rely on such strategies for selecting the steplength[1,2,3]. Part of them mimic traditional step acceptance rules using stochastic estimates of functions and gradients, which are required to be sufficiently accurate in probability.

The purpose of these methods is to partially overcome the dependence of the steplengths from the Lipschitz constant of the gradient.

[1] S. Bellavia, N. Krejić, B. Morini, Comput. Optim. Appl. 76, 701–736, 2020

[2] S. Bellavia, N. Krejić, B. Morini, S. Rebegoldi, submitted, 2021

[3] R. Chen, M. Menickelly, K. Scheinberg, Math. Progr. 169(2), 447–487, 2018

Stochastic trust-region approach

- Let $g_k \in \mathbb{R}^n$ be a stochastic gradient (e.g. subsampled gradient).
- Consider a linear model of f_N around the iterate x_k

$$m_k(p) = f_N(x_k) + g_k^T p.$$

- Minimize the model over the ball $B(x_k, \Delta_k)$ of trust-region radius Δ_k

$$p_k = \underset{\|p\| \leqslant \Delta_k}{\text{argmin}}\ m_k(p)$$

  and compute the trial point as

$$x_k + p_k = x_k - \frac{\Delta_k}{\|g_k\|} g_k.$$

  This is an SG step with adaptive steplength!

- Accept or reject the trial point according to an acceptance criterion based on sufficiently accurate estimates of functions and gradients.

STORM (STochastic Optimization with Random Models)[4]

Choose $x_0 \in \mathbb{R}^n$, $0 < \Delta_0 < \Delta_{\max}$, $\gamma > 1$, $\eta_1, \eta_2 > 0$.
FOR $k = 0, 1, 2, \ldots$

1. Build the model
$$m_k(x_k + s) = f_N(x_k) + g_k^T s, \quad \forall\ s \in B(x_k, \Delta_k).$$
2. Compute the step
$$s_k = \underset{\|s\| \leqslant \Delta_k}{\operatorname{argmin}}\ m_k(x_k + s) = -\frac{\Delta_k}{\|g_k\|} g_k.$$
3. Compute estimates $f^{k,0}$ and $f^{k,s}$ of f_N at x_k and $x_k + s_k$ and
$$\rho_k = \frac{f^{k,0} - f^{k,s}}{m_k(x_k) - m_k(x_k + s_k)}.$$
4. If $\rho_k \geqslant \eta_1$ and $\|g_k\| \geqslant \eta_2 \Delta_k$ set
$$x_{k+1} = x_k + s_k, \quad \Delta_{k+1} = \min\{\gamma \Delta_k, \Delta_{\max}\},$$
otherwise set $x_{k+1} = x_k$, $\Delta_{k+1} = \Delta_k / \gamma$.

[4] R. Chen, M. Menickelly, K. Scheinberg, Math. Progr. 169(2), 447–487, 2018

STORM algorithm

Function and gradient estimates need to be sufficiently accurate in probability:

$$Pr\{|f_N(y) - m_k(y)| \leqslant \kappa\Delta_k^2, \quad \|\nabla f_N(y) - \nabla m_k(y)\| \leqslant \kappa\Delta_k, \; y \in B(x_k, \Delta_k)\} \geqslant \alpha$$

$$Pr\{|f_k^0 - f_N(x_k)| \leqslant \epsilon_F\Delta_k^2, \quad |f_k^s - f_N(x_k + s_k)| \leqslant \epsilon_F\Delta_k^2\} \geqslant \beta$$

where $\alpha, \beta \in (0,1)$, $\epsilon_F, \kappa > 0$.

- Probabilistic accuracy of m_k, f_k^0, f_k^s guarantees convergence in probability
- Function and gradient need to be estimated with increasingly high precision!

SIRTR (Stochastic Inexact Restoration Trust-Region)[1]

In a recent paper[1], the authors propose a stochastic first-order trust-region method with the following features.

- The trust-region model and acceptance rule employ both function and gradient estimates (similarly to STORM).
- The function sample size is computed dynamically according to a deterministic rule inspired by the Inexact Restoration (IR) Method[2] (unlike STORM).
- We require probabilistic accuracy for the gradient estimates only when the full function sample size is reached (unlike STORM).

GOAL: delay the use of the full function sample size and the adoption of probabilistically accurate random models as much as possible.

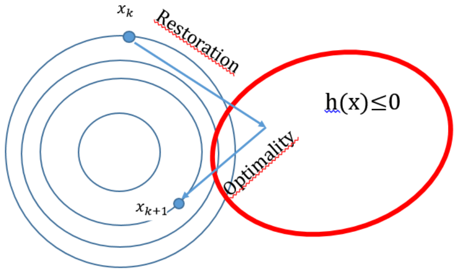
[1] S. Bellavia, N. Krejić, B. Morini, S. Rebegoldi, submitted, 2021

[2] J.M. Martinez, E.A. Pilotta, J. Optim. Theory Appl. 104, 135–163, 2000

The Inexact Restoration method

The Inexact Restoration (IR) method is a constrained optimization tool suitable for problems feasibility need not be imposed at all iterations.
The key idea is to improve feasibility and optimality in separate procedures.
Each iteration ensures the sufficient decrease of a suitable merit function and, under certain assumptions, convergence to a feasible optimal point.

$$\min_{h(x)\leqslant 0} \; f(x)$$



SIRTR algorithm

IDEA: apply the IR strategy to dynamically select the function sample size. To this aim, let us rewrite the finite-sum minimization problem as

$$\min_{x\in\mathbb{R}^n} f_M(x) = \frac{1}{M}\sum_{i\in I_M}\phi_i(x)$$

$$\text{s.t. } M = N$$

where $I_M \subset \{1,\dots,N\}$, $|I_M| = M$.

SIRTR algorithm

We introduce the merit function

$$\Psi(x, M, \theta) = \theta f_M(x) + (1-\theta) h(M), \quad \theta \in (0,1).$$

where $h : \mathbb{N} \to \mathbb{R}$ is a measure of the level of infeasibility with respect to the constraint $M = N$.

Example
$h(M) = (N - M)/N$.

SIRTR algorithm

1. Restoration phase

- compute $\tilde{N}_{k+1}$ such that $h(\tilde{N}_{k+1}) \leqslant rh(N_k)$, $r \in (0,1)$;
- compute the function sample size $N_{k+1} \leqslant \tilde{N}_{k+1}$ such that

$$h(N_{k+1}) - h(\tilde{N}_{k+1}) \leqslant \mu\Delta_k^2, \quad \mu > 0.$$

2. Optimality phase

- compute the trial point $x_k + p_k$ using a model $m_k(p)$;
- consider the predicted and actual reduction defined as

$$\begin{aligned}
&\mathrm{Pred}_k(\theta_{k+1}) = \theta_{k+1}(f_{N_k}(x_k) - m_k(p_k)) + (1-\theta_{k+1})(h(N_k) - h(\tilde{N}_{k+1})) \\
&\mathrm{Ared}(x_k + p_k, \theta_{k+1}) = \Psi(x_k, N_k, \theta_{k+1}) - \Psi(x_{k+1}, N_{k+1}, \theta_{k+1})
\end{aligned}$$

and accept the trial point only if

$$\mathrm{Ared}(x_k + p_k, \theta_{k+1}) \geqslant \eta\,\mathrm{Pred}_k(\theta_{k+1}), \quad \eta \in (0,1).$$

Numerical experience

Let $\{(a_i, b_i)\}_{i=1}^N$ denote the pairs forming the training set, being $a_i \in \mathbb{R}^n$ the vector containing the entries of the i-th example and $b_i \in \{0, 1\}$ its label. The classification problem is solved by solving the nonconvex problem

$$\min_{x \in \mathbb{R}^n} f_N(x) = \frac{1}{N} \sum_{i=1}^{N} \left(b_i - \frac{1}{1 + e^{-a_i^T x}} \right)^2 .$$

	Training set		Testing set
Data set	N	n	N_T
CINA0	10000	132	6033
A9A	22793	123	9768
IJCNN1	49990	22	91701
MNIST	60000	784	10000
HTRU2	10000	8	7898

Numerical experience

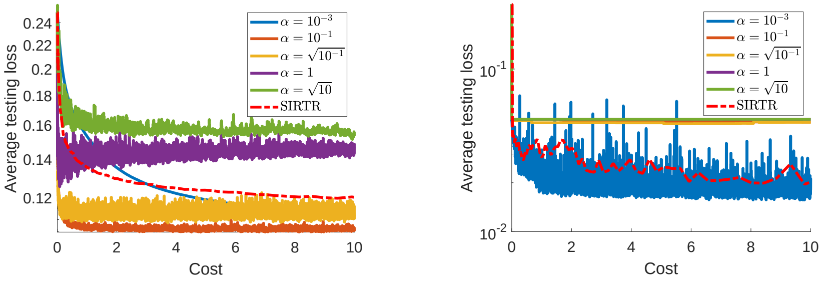


Figure: SIRTR versus Trust-Regionish algorithm[1] for several choices of the steplength α. Decrease of the (average) testing loss f_{N_T} w.r.t. the (average) computational time. From left to right: A9A and HTRU2 datasets.

[1] F.E. Curtis, K. Scheinberg, R. Shi, INFORMS J. Optim. 1(3), 200–220, 2019

[1] F.E. Curtis, K. Scheinberg, R. Shi, INFORMS J. Optim. 1(3), 200–220, 2019