Optimization algorithms for machine learning

Coralia Cartis (University of Oxford)

Joint with Katya Scheinberg (Lehigh University)
Jose Blanchet (Columbia) and Matt Menickelly (Argonne)

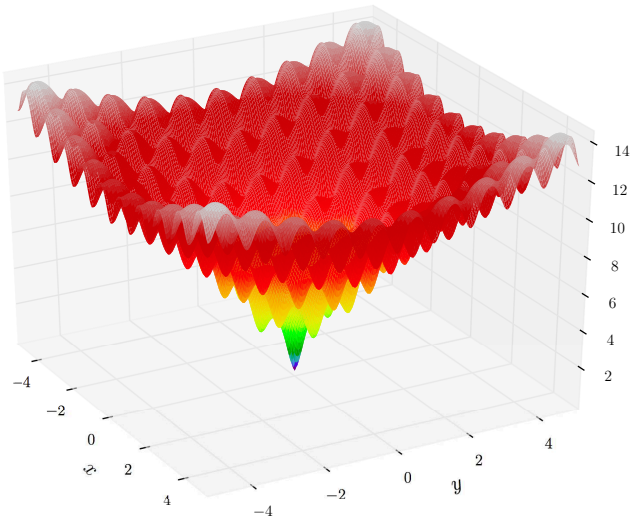
Games, Dynamics and Optimization 2019
Universitatea Babeș-Bolyai, Cluj-Napoca

Nonconvex optimization

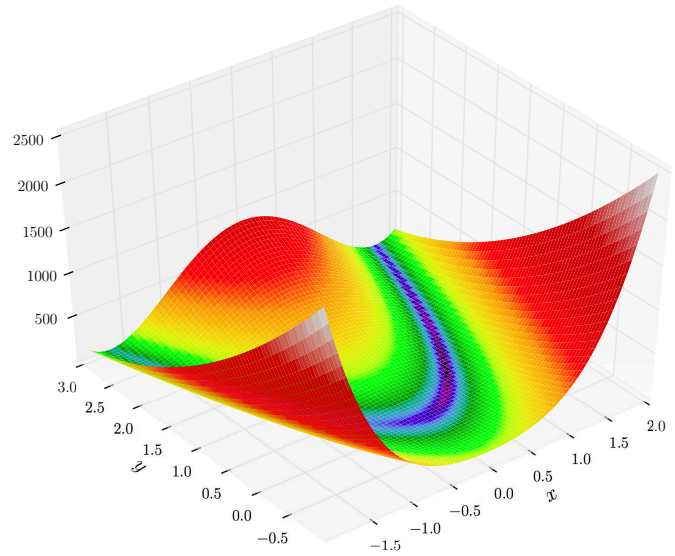
Find (local) solutions of the optimization problem:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \quad \text{where} \quad f \quad \text{is smooth}$$

with $f(x)$ possibly nonconvex and n possibly large.



Ackeley's function



Rosenbrock's function

Methods for nonconvex optimization

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \quad \text{where} \quad f \quad \text{is smooth.}$$

• f has gradient vector ∇f (first derivatives) and Hessian matrix $\nabla^2 f$ (second derivatives).

$\longrightarrow$ local minimizer x_* with $\nabla f(x_*) = 0$ (stationarity) and $\nabla^2 f(x_*) \succ 0$ (local convexity).

Derivative-based methods:

- user-given $x_0 \in \mathbb{R}^n$, generate iterates x_k, $k \geq 0$.
- $f(x_k + s) \approx m_k(s)$ simple model of f at x_k;
  m_k linear or quadratic Taylor approximation of f.
  $s_k \to \min_s m_k(s)$; $s_k \to x_{k+1} - x_k$
- terminate within ϵ of optimality (small gradient values).

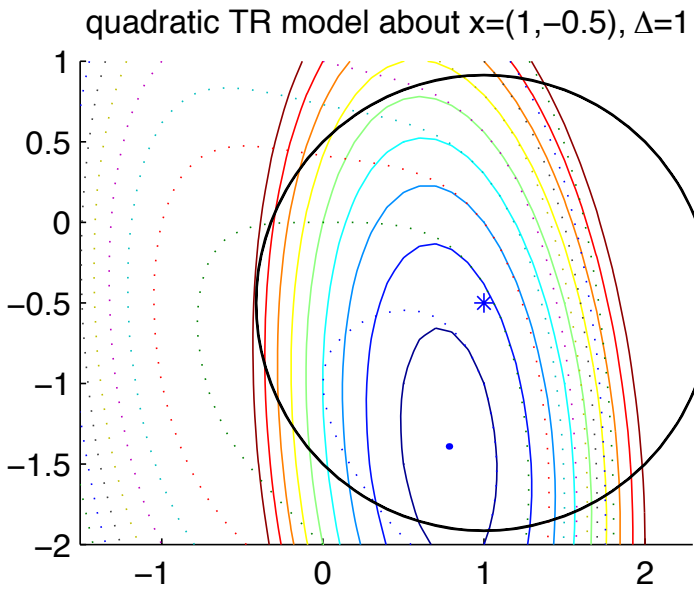
Derivative-based local models

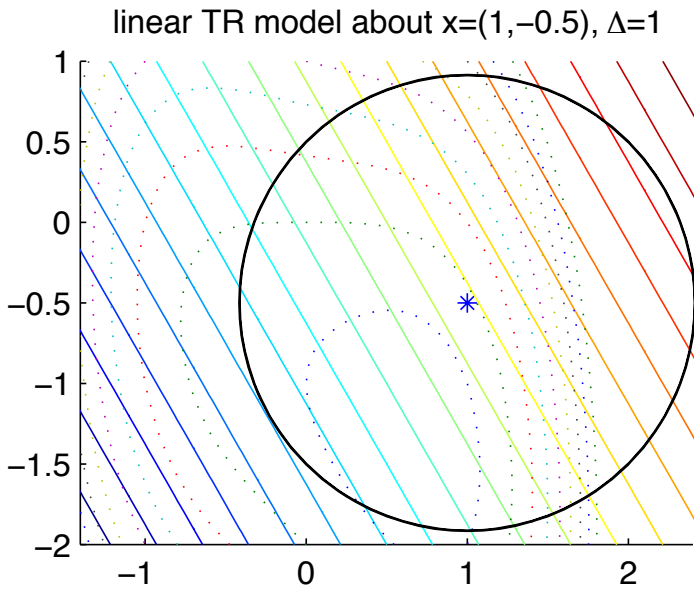
Choices of models

- linear : $m_k(s) = f(x_k) + \nabla f(x_k)^T s$
  $\longrightarrow$ s_k steepest descent direction.
- quadratic : $m_k(s) = f(x_k) + \nabla f(x_k)^T s + \frac{1}{2} s^T \nabla^2 f(x_k) s$
  $\longrightarrow$ s_k Newton-like direction.

Safeguard s_k to ensure method converges: linesearch, Trust-Region (TR).

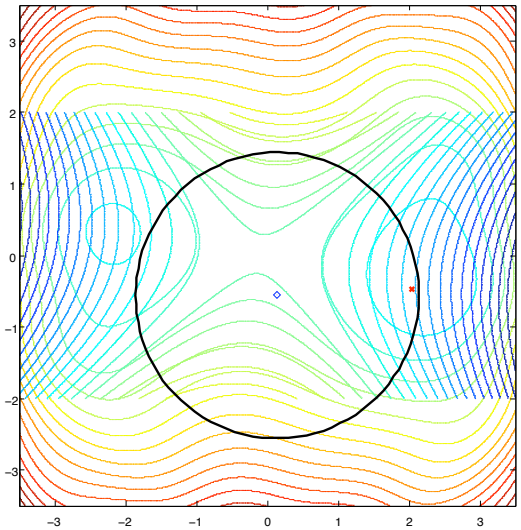
(TR subproblem) $s_k \longrightarrow (\text{approx.}) \min_s m_k(s)$ subject to $\|s\| \leq \Delta_k$.





Trust-region methods – a convergent framework

- compute $s_k \longrightarrow \min_s m_k(s)$ subject to $\|s\| \leq \Delta_k$ (TR)
- set $x_{k+1} = x_k + s_k$ if m_k and f 'agree' at $x_k + s_k$
- otherwise set $x_{k+1} = x_k$ and reduce the TR radius Δ_k



$$f(x) = -10x_1^2 + 10x_2^2 + 4sin(x_1x_2) - 2x_1 + x_1^4$$

k	Δ_k	s_k	$f(x_k + s_k)$	$\Delta f / \Delta m_k$	x_{k+1}
0	**1**	$(0.05, 0.93)$	43.742	0.998	$x_0 + s_0$
1	**2**	$(-0.62, 1.78)$	2.306	1.354	$x_1 + s_1$
2	**4**	$(3.21, 0.00)$	6.295	-0.004	x_2
3	**2**	$(1.90, 0.08)$	-29.392	0.649	$x_2 + s_2$

Models use curvature; go beyond steepest descent for best performance.
Methods are adaptive.

Worst-case evaluation complexity of methods

Global rates of convergence from any initial guess

Under sufficient smoothness assumptions on f (Lipschitz continuity), for any $\epsilon > 0$, the algorithms generate $\|\nabla f(x_k)\| \leq \epsilon$ (and $\lambda_{\min}(\nabla^2 f(x_k)) \geq -\sqrt{\epsilon}$) in at most $k_\epsilon^{\rm alg}$ iterations/evaluations:

Criticality	SD	Newton/TR/LS	ARC	TR+/ LS+
$\|\nabla f(x_k)\| \leq \epsilon$	$\mathcal{O}(\epsilon^{-2})$	$\mathcal{O}(\epsilon^{-2})$	$\mathcal{O}(\epsilon^{-\frac{3}{2}})$	$\mathcal{O}(\epsilon^{-\frac{3}{2}})$
$\lambda_{\min}(\nabla^2 f(x_k)) \geq -\sqrt{\epsilon}$	–	$\mathcal{O}(\epsilon^{-\frac{3}{2}})$	$\mathcal{O}(\epsilon^{-\frac{3}{2}})$	$\mathcal{O}(\epsilon^{-\frac{3}{2}})$

- $\mathcal{O}(\cdot)$ contains $f(x_0) - f_{\rm low}$, $L_{\rm grad}$ or $L_{\rm Hessian}$ and algorithm parameters, independent of accuracy $\epsilon > 0$.
- all bounds are sharp, ARC bound is optimal for second-order methods

[C, Gould & Toint,'10,'11, '17; Carmon et al ('18)]

Derivative-based optimization algorithms

Competing, sophisticated, mature techniques available: employing L-BFGS, linesearch, trust-region.

Powerful theoretical guarantees of convergence (from arbitrary initial guess; fast asymptotically) for large class of nonconvex pbs

Much reliable and efficient software suitable for large-scale problems ($n \gg 10^3$): KNITRO, GALAHAD, IPOPT, NAG...

Methods/solvers require accurate function and derivative(s) values to be provided - manually written code, automatic differentiation or finite-differences.

$\Longrightarrow$ Limitations...

Modern challenges to derivative-based solvers

Limitations of derivative-based solvers:

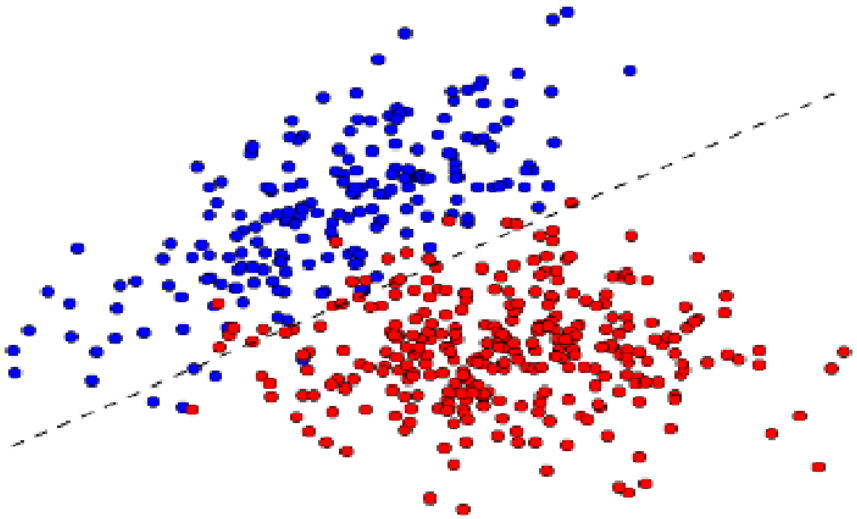
- require accurate/exact function values and (at least) first-derivatives of f to be available to the solver
  - use derivative-free optimization methods when derivatives are unavailable: suitable for noisy/stochastic problems; only guaranteed to provide local solutions of nonconvex optimization landscapes, but successful for global optimization
  - suitable for $O(100)$ variables

The optimization challenges of modern applications: huge scale, stochastic, inexact data/problems.

Optimization in machine learning

Supervised learning problems

[Scheinberg, 2018; Curtis & Scheinberg, 2017; Bouttou et al, 2018]



Binary classification: Map $w \in \mathcal{W} \subseteq \Re^{d_w}$ to $y \in \mathcal{Y} \subseteq \{-1, 1\}$

Choose predictor $p(w; x) : \mathcal{W} \to \mathcal{Y}$

If $p(w; x) = w^T x$ - linear classifier; more generally, $p(w; x)$ nonlinear (such as neural network).

Selection of the best classifier:

- Minimize Expected/Empirical Error, Loss, AUC

Finding the best predictor

[Curtis & Scheinberg, 2017; Scheinberg, 2018]

$$\min_{x \in \mathcal{X}} f(x) := \int_{\mathcal{W} \times \mathcal{Y}} 1[yp(w; x) \leq 0] dP(w, y).$$

$\longrightarrow$ intractable due to unknown distribution

Use instead the empirical risk of $p(w; x)$ over finite training set $\mathcal{S}$,

$$\min_{x \in \mathcal{X}} f_{\mathcal{S}}(x) := \frac{1}{m} \sum_{i=1}^{m} 1[y_i p(w_i; x) \leq 0].$$

$\longrightarrow$ hard to solve, nonsmooth.

Use the smooth and 'easy' empirical loss of $p(w; x)$ over the finite training set $\mathcal{S}$,

$$\min_{x \in \mathcal{X}} \hat{f}_{\mathcal{S}}(x) := \frac{1}{m} \sum_{i=1}^{m} l(p(w_i; x), y_i) = \sum_{i=1}^{m} f_i(x).$$

$\longrightarrow$ tractable but huge scale in n and m; deterministic formulation.

Care also about expected loss $\mathbb{E}(l(p(w; x), y))$ (stochastic).

Standard stochastic gradient method

At iterate x^k,

- generate predefined mini-batch size $|\mathcal{S}_k| \ll m$ and (random) components $i \in \mathcal{S}_k$, and calculate

$$\nabla_{\mathcal{S}_k} f(x_k) := \frac{1}{|\mathcal{S}_k|} \sum_{i \in \mathcal{S}_k} \nabla f_i(x_k)$$

- calculate the next iterate

$$x_{k+1} = x_k - \alpha_k \nabla_{\mathcal{S}_k} f(x_k),$$

  where α_k is a predefined stepsize (learning rate).

Commonly assumes $\mathbb{E}(\nabla_{\mathcal{S}} f(x)) = \nabla f(x)$.

Our work: adaptive methods (for $\mathcal{S}_k$ and α_k), including curvature (ie second-order), allowing biased estimates, with complexity guarantees.

Methods with probabilistically accurate models

Probabilistic local models and methods

Context/purpose: f still smooth, but derivatives are inaccurate/impossible/expensive to compute.

- Local models may be "good" / "sufficiently accurate" only with certain probability, for example:

  $\longrightarrow$ models based on random sampling of function values (within a ball)

  $\longrightarrow$ finite-difference schemes in parallel, with total probability of any processor failing less than 0.5

  $\longrightarrow$ stochastic gradient with varying batch size $\mathcal{S}_k$ and stepsize

- Use these probabilistic models inside classical linesearch, trust-region, ARC methods.
- Expected number of iterations to generate sufficiently small true gradients?

Connections to model-based derivative-free optimization (Powell; Conn, Scheinberg & Vicente'06)

Probabilistic trust region framework

Assume that f is accurate/exact.

- Probabilistically accurate local model:

$$m_k(s) = f(x_k) + s^T g_k + \tfrac{1}{2} s^T B_k s$$

with $g_k \approx \nabla f(x_k)$ and $B_k \approx \nabla^2 f(x_k)$ [along the step s_k], where $\approx$ holds with a certain probability $p \in (0, 1]$ (conditioned on the past).

$\longrightarrow I_k$ occurs : k true iteration; else, k false.

- $min_s m_k(s)$ s.t. $\|s\| \leq \Delta_k$ [cf. derivative-based methods!];
- adjust Δ_k [cf. derivative-based methods!]

$$\Delta_k \nearrow \text{ if } f(x_k + s_k)' <' f(x_k)$$
$$\Delta_k \searrow \text{ if } f(x_k + s_k) \geq f(x_k)$$

Algorithm : stochastic process and its realizations.

Probabilistic Trust Region (P-TR) - complexity guarantees

Assume that f is accurate/exact.

Complexity: If f is sufficiently smooth, then the expected number of iterations that P-TR takes until $\|\nabla f(x^k)\| \leq \epsilon$ satisfies

$$\mathbb{E}(N_\epsilon) \leq \frac{1}{2p-1} \cdot \kappa_{\mathrm{p-ls}} \cdot (f(x_0) - f_{\mathrm{low}}) \cdot \epsilon^{-2}$$

provided the probability of sufficiently accurate models is $p > \frac{1}{2}$.

This implies $\lim_{k\to\infty} \inf_k \|\nabla f(x_k)\| = 0$ with probability one.

Expected number of iterations $\mathbb{E}(N_\epsilon)$ to reach ϵ accuracy:
$\longrightarrow$ N_ϵ hitting time for stochastic process $\{\|\nabla f(X^k)\| \leq \epsilon\}$

Probabilistic ARC (P-ARC) - complexity guarantees

Assume that f is accurate/exact. Use the local models

$$m_k(s) = f(x_k) + s^T g_k + \tfrac{1}{2} s^T B_k s + \tfrac{1}{3}\sigma_k \|s\|^3.$$

Complexity: If f is sufficiently smooth, then the expected number of iterations that P-ARC takes until $\|\nabla f(x^k)\| \leq \epsilon$ satisfies

$$\mathbb{E}(N_\epsilon) \leq \frac{1}{2p-1} \cdot \kappa_{\mathrm{p-arc}} \cdot (f(x_0) - f_{\mathrm{low}}) \cdot \epsilon^{-\frac{3}{2}}$$

provided the probability of sufficiently accurate models is $p > \frac{1}{2}$.

This implies $\lim_{k\to\infty} \inf_k \|\nabla f(x_k)\| = 0$ with probability one.

These bounds match the deterministic complexity bounds of corresponding methods (in accuracy order).

Generating probabilistic models

- Stochastic gradient and batch sampling [Nocedal et al, 2012]

$$\|\nabla f_{S_k}(x^k) - \nabla f(x^k)\| \leq \mu \|\nabla f_{S_k}(x^k)\|$$

with $\mu \in (0, 1)$ and fixed, and sufficiently small and constant $\alpha_k = \alpha \leq \frac{1-\mu}{L_g}$.

Then model $m_k(s) = f(x^k) + \nabla f_{S_k}(x^k)^T(x - x^k)$ is sufficiently accurate for a given fixed step size α.

- we allow the model to fail with probability less than 0.5, variable stepsize α_k and f nonconvex.

If $\mathbb{E}(\nabla_S f(x^k)) = \nabla f(x^k)$, we can show that $\nabla_{S_k} f(x^k)$ is probabilistically sufficiently accurate with prob. $p > 0.5$ provided $|S_k|$ is sufficiently large.

$\longrightarrow$ generalization of linesearch stochastic gradient methods.

Generating (p)-accurate models...

Models formed by sampling of function values in a ball $B(x_k, \Delta_k)$ (model-based dfo) [Conn et al, 2008; Bandeira et al, 2015]

M_k (p)-fully linear model: if the event

$$I_k^l = \{\|\nabla f(X^k) - G^k\| \leq \kappa_g \Delta_k\}$$

holds at least w.p. p (conditioned on the past).

Linesearch methods: choose $\Delta_k = \alpha_k \xi_k$. Then m_k fully linear implies m_k sufficiently accurate if:

- ξ_k sufficiently small, of order ϵ; or
- adjust ξ_k in the algorithm: accept step when $\|g^k\| \geq \kappa \xi_k$, shrink ξ_k and reject step otherwise.

Generating (p)-accurate models...

Models formed by sampling of function values in a ball $B(x_k, \Delta_k)$ (model-based dfo) [Conn et al, 2008; Bandeira et al, 2015]

M_k (p)-fully quadratic model: if the event

$$I_k^q = \{\|\nabla f(X^k) - G^k\| \leq \kappa_g \Delta_k^2 \quad \text{and} \quad \|\nabla^2 f(X^k) - B^k\| \leq \kappa_H \Delta_k\}$$

holds at least w.p. p (conditioned on the past).

Cubic regularization methods: choose $\Delta_k = \xi_k/\sigma_k$. Then m_k fully quadratic implies m_k sufficiently accurate if:

- ξ_k sufficiently small, of order ϵ; or
- adjust ξ_k in the algorithm: accept step when $\|s^k\| \geq \kappa \xi_k$, shrink ξ_k and reject step otherwise.

This framework applies to subsampling gradients and Hessians in ARC [Kohler & Lucchi ('17), Roosta et al. ('17)]

Stochastic optimization

Now let us assume that our (observed) function values are also inaccurate/noisy/random. Still,

$$\underset{x \in \mathbf{R}^n}{\text{minimize}} \quad f(x) \quad \text{where } f \text{ smooth},$$

with $f(x)$ possibly nonconvex; but $f(x)$ can only be computed with some noise, so we observe

$$\hat{f}(x) = f(x, \omega), \text{ where } \omega \text{ is a random variable.}$$

- in trust-region method, use models $m_k(s)$ that are (p)-accurate in $B(x_k, \Delta_k)$ with probability p.
- given Δ_k, assume estimates $\hat{f}(x_k) \approx f(x_k)$ and $\hat{f}(x_k + s_k) \approx f(x_k + s_k)$ are accurate with probability q:

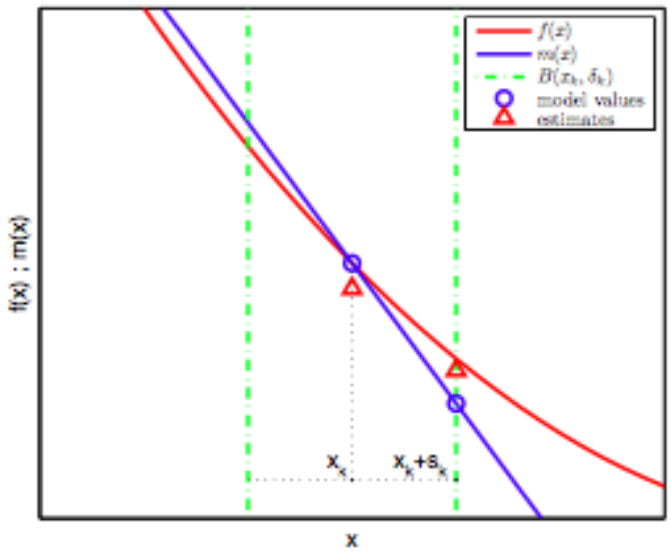
$$|\hat{f}(x_k) - f(x_k)| \leq \epsilon_F \Delta_k^2 \text{ and } |\hat{f}(x_k + s_k) - f(x_k + s_k)| \leq \epsilon_F \Delta_k^2$$

STORM - a stochastic trust-region method

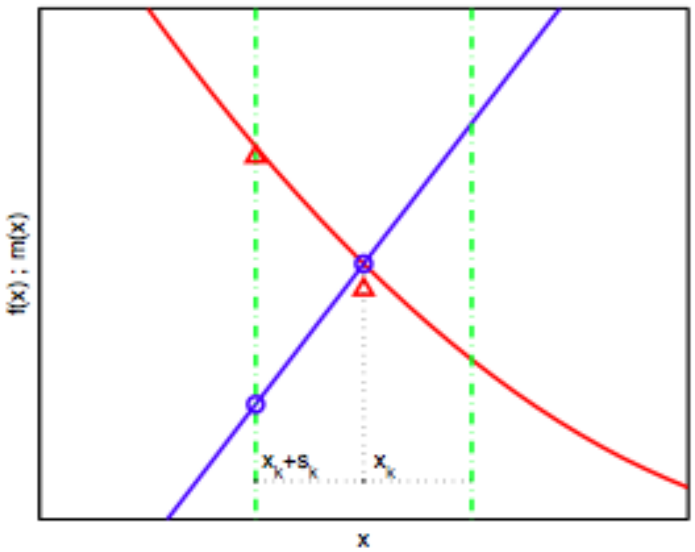
[Chen, Mineckelly & Scheinberg, 2015]

In P-TR, let $f(x) \longrightarrow \hat{f}(x)$ estimates. Occurs in $m_k(s)$ and in measuring progress $\hat{f}(x_k + s_k)' <' \hat{f}(x_k)$. Also require $\|g_k\| \geq \kappa \Delta_k$ for step acceptance.

Six types of iterations (successful, unsuccessful, true and false, good and bad)



(a) Good model; good estimates. True successful steps.



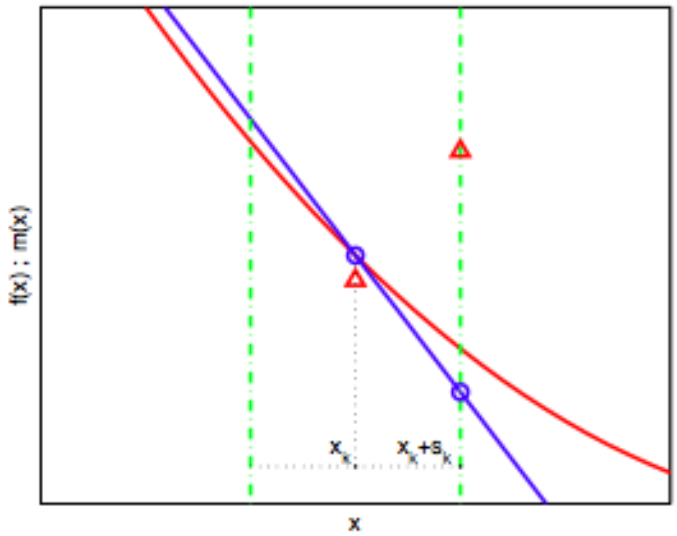
(b) Bad model; good estimates. Unsuccessful steps.

STORM - a stochastic trust-region method

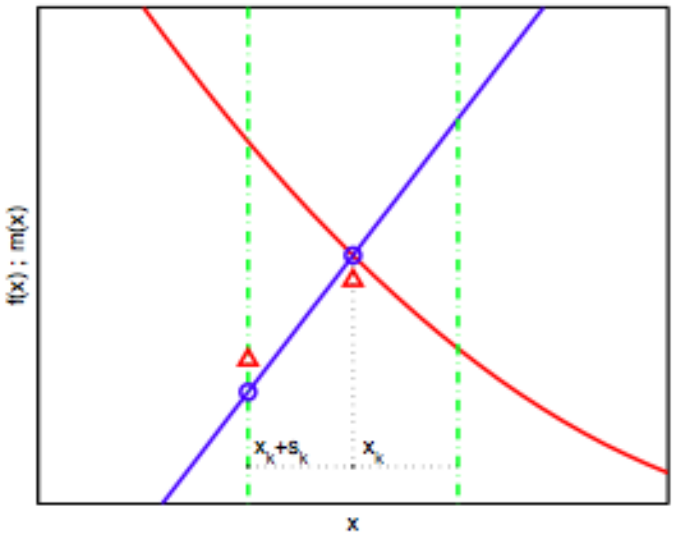
[Chen, Mineckelly & Scheinberg, 2015]

In P-TR, let $f(x) \longrightarrow \hat{f}(x)$ estimates. Occurs in $m_k(s)$ and in measuring progress $\hat{f}(x_k + s_k)' <' \hat{f}(x_k)$. Also require $\|g_k\| \geq \kappa \Delta_k$ for step acceptance.

Six types of iterations (successful, unsuccessful, true and false, good and bad)



(c) Good model; bad estimates.
Unsuccessful steps.



(d) Bad model; bad estimates.
False successful steps: f can increase!

Probabilistic TR - complexity

If f sufficiently smooth, then the expected number of iterations that STORM takes until $\|\nabla f(x^k)\| \leq \epsilon$ satisfies

$$\mathbb{E}(N_\epsilon) \leq \frac{1}{2pq-1} \cdot \kappa_{\mathrm{p-tr}} \cdot \max\{f(x_0) - f_{\mathrm{low}}, \Delta_0\} \cdot \epsilon^{-2}$$

provided m_k and $\hat{f}$ are (p)-accurate with probabilities p and q sufficiently large and accuracy ϵ_F sufficiently small.

Then also, $\lim_{k\to\infty} \inf_k \|\nabla f(x_k)\| = 0$ with probability one.

Analysis

- Define stochastic process
$$\Phi_k = \tau(f(x_k) - f_*) + (1-\tau)\Delta_k^2$$
and analyze joint process $\{\Phi_{k+1} - \Phi_k, \Delta_k\}$. A renewal-reward process, general framework.

Probabilistic TR - second-order framework

- we also want $\lambda_{\min}(\nabla^2 f)\} \geq -\sqrt{\epsilon}$
- (p)-probabilistically fully accurate quadratic models w.p. p - as usual for quadratic models
- stronger assumption on evaluations:
  $\mathbb{E}(|\hat{f}(x_k) - f(x_k)|) \leq \epsilon_F \Delta_k^3$, $\mathbb{E}(|\hat{f}(x_k + s_k) - f(x_k + s_k)|) \leq \epsilon_F \Delta_k^3$

Theorem: in the same conditions as before, the expected number of iterations that second-order STORM takes until it also satisfies $\lambda_{\min}(\nabla^2 f)\} \geq -\sqrt{\epsilon}$ is at most

$$\mathbb{E}(N_\epsilon) \leq \frac{1}{2pq-1} \cdot \kappa_{\mathrm{p-tr-2}} \cdot \max\{f(x_0) - f_{\mathrm{low}}, \Delta_0\} \cdot \epsilon^{-\frac{3}{2}}$$

Probabilistic TR for stochastic optimization

Advantages:

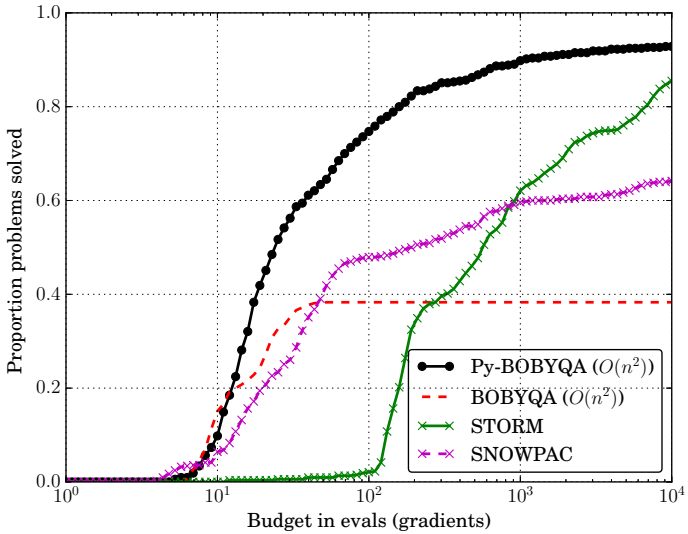
- sampling rate varies according to TR radius; diverse models of noise, including biased noise;
- encouraging numerical performance for different noise models[Chen, Mineckelly & Scheinberg ('15); C, Fiala, Marteau, Roberts ('18)]

Constructing (p)-sufficiently accurate models and function values:

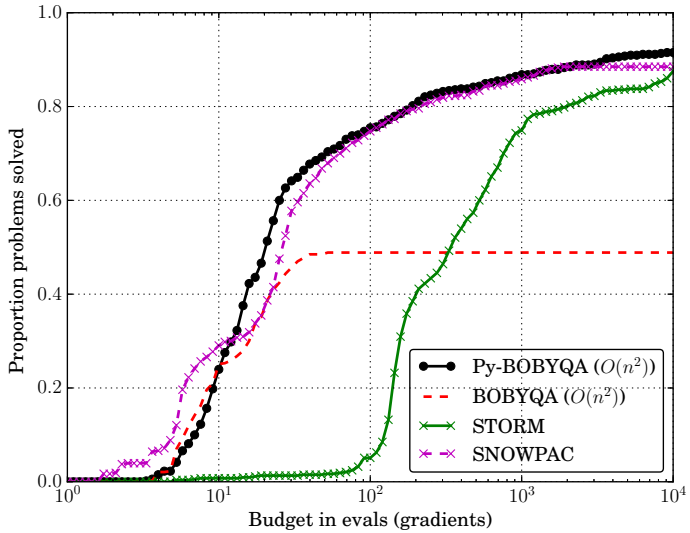
- stochastic noise: iid noise, or more generally, unbiased for all x ($\mathbb{E}(\hat{f}) = f$; $\mathrm{Var}\hat{f} < \infty$). Use sample averaging of r noisy $\hat{f}$ values/gradients for r large enough, of order Δ_k^{-4} (in numerics, $r \sim \Delta_k^{-1}$).
- allows failure in computation of function values.

Numerical results: STORM vs derivative-free optimization

[C, Fiala, Marteau, Roberts, 2018]



Multiplicative noise



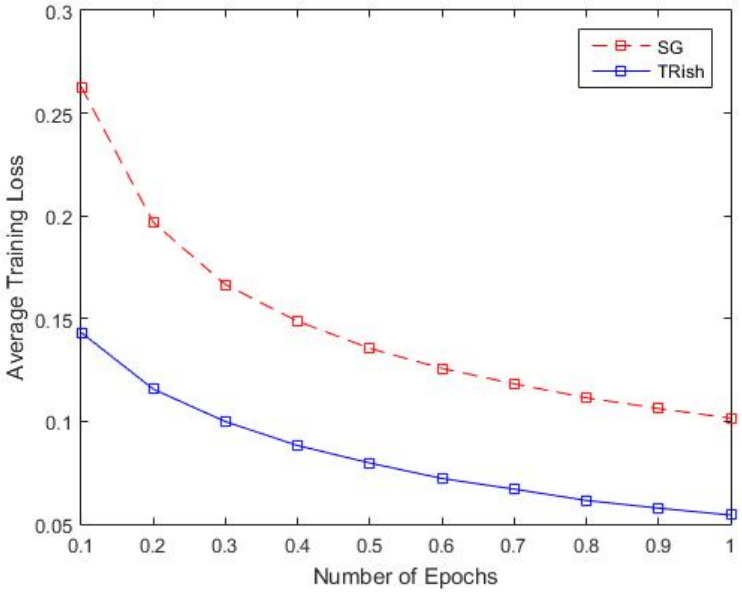
Additive noise

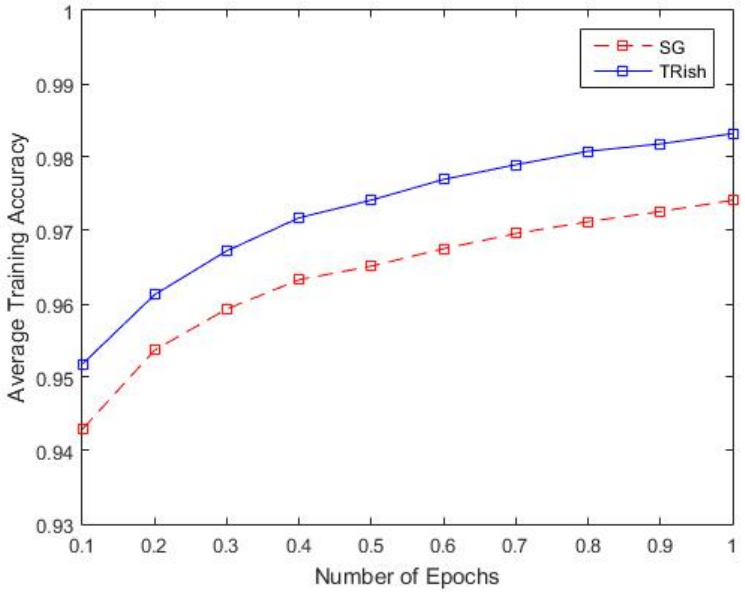
Average % test problems solved after given # function evaluations; higher values better

Standard optimization test set (Moré-Wild), data profiles.

Numerical results: first-order trust-region vs SG

[Curtis, Scheinberg, Shi, 2017]





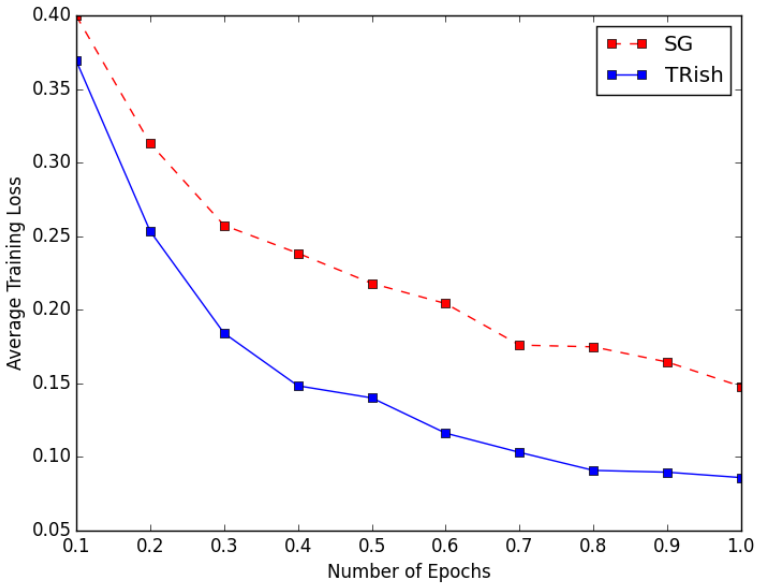
Average training loss and accuracy

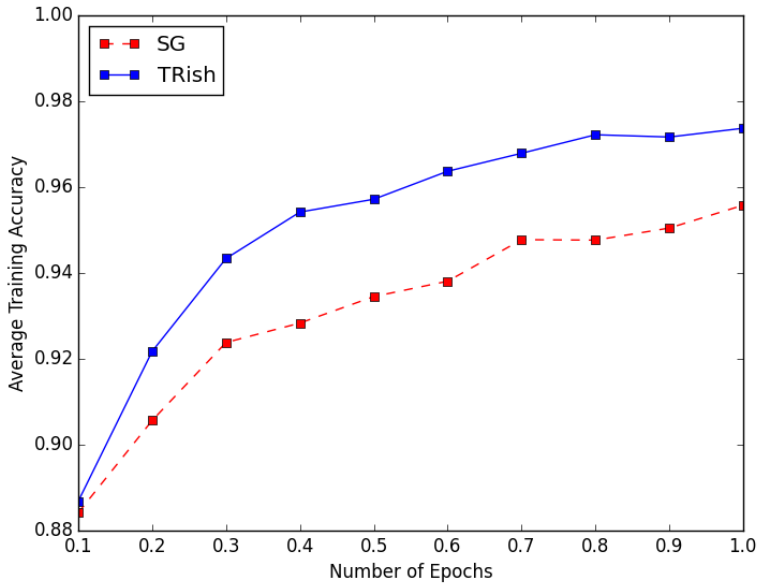
Logistic regression for binary classification (LIBSVM, $n = O(10^4)$)

Applied a first-order stochastic trust-region variant, with adaptive trust region and scaling.

Numerical results: first-order trust-region vs SG

[Curtis, Scheinberg, Shi, 2017]





Average training loss and accuracy

Two-layer CNN, MNIST data set ($n = O(10^5)$).

Applied a first-order stochastic trust-region variant, with adaptive trust region and scaling.

Conclusions

Going beyond Stochastic Gradient for training NN with second order methods and adaptivity? Need powerful implementations

Tune your (algorithm) parameters with DFO (derivative-free) codes; suitable for medium scale (noisy/stochastic) problems; scaling up model-based DFO methods?

References

- C, Scheinberg, Global convergence rate analysis of unconstrained optimization methods based on probabilistic models, Mathematical Programming, 2017
- Blanchet, C, Menickelly, Scheinberg, Convergence rate analysis of a stochastic trust region via submartingales, INFORMS J Optimization, Special Issue on Optimization for Machine Learning, 2019

Overview articles

Suggested reading (review articles):

- **F. E. Curtis and K. Scheinberg. Optimization Methods for Supervised Machine Learning: From Linear Models to Deep Learning.** In INFORMS Tutorials in Operations Research, chapter 5, page 89–114. Institute for Operations Research and the Management Sciences (INFORMS), 2017.
- **L. Bottou, F. E. Curtis, and J. Nocedal. Optimization Methods for Large-Scale Machine Learning.** SIAM Review, 60(2):223–311, 2018.

Further reading (monographs, edited volumes):

- **P. Jain and P. Kar. Non-convex optimization for machine learning,** IEEE 2018. (available on ArXiv)
- **A. Beck. First order methods in optimization.** MOS-SIAM Series on Optimization, SIAM 2017.
- **S. Sra, S. Nowozin, and S.J. Wright. Optimization for machine learning.** MIT Press, 2012. [A classic reference, edited volume]