STOCHASTIC OPTIMIZATION IN MACHINE LEARNING

Case Studies in Nonlinear Optimization

F. Bauer S. Chambon R. Halbig S. Heidekrüger J. Heuke

July 11, 2015

Technische Universität München

We're not running out of data anytime soon. It's maybe the only resource that grows exponentially.

Andreas Weigend

OUTLINE

INTRODUCTION

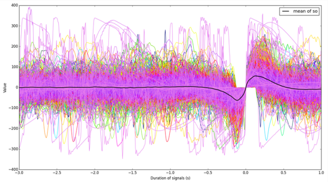
Implementation of autonomously learning software for:

- Discovery of patterns and relationships in data
- Prediction of future events

Examples:

Image Denoising

Electroencephalography (EEG)



Section 4

Section 5

INTRODUCTION: ML AND OPTIMIZATION I

Training a Machine Learning model means finding optimal parameters ω:

$$\omega^* = \text{argmin}_\omega F(\omega, X, z)$$

- F: Loss function
- X: The training data
- z: Training labels

After we have found ω^*, we can do Prediction on new data points:

$$\hat{z}_i := h(\omega^*, x_i)$$

- x_i: new data point with *unknown* label z_i
- h: hypothesis function of the ML model

- Massive amounts of training data
- Construction of very large models
- Handling high memory/computational demands

Stochastic Methods

$$F(\omega) := \mathbb{E}\left[f(\omega, \xi)\right]$$

$$F(\omega) := \mathbb{E}\left[f(\omega, \xi)\right]$$

- ξ: Random variable; takes the form of an input-output-pair (x_i, z_i)

$$F(\omega) := \mathbb{E}\left[f(\omega,\xi)\right] = \frac{1}{N}\sum_{i=1}^{N} f(\omega, x_i, z_i)$$

- ξ: Random variable; takes the form of an input-output-pair (x_i, z_i)
- f: Partial loss function corresponding to a single data point.

INTRODUCTION: STOCHASTIC FRAMEWORK

$$F(\omega) := \mathbb{E}\left[f(\omega, \xi)\right] = \frac{1}{N}\sum_{i=1}^{N} f(\omega, x_i, z_i)$$

- ξ: Random variable; takes the form of an input-output-pair (x_i, z_i)
- f: Partial loss function corresponding to a single data point.
- Example loss function: $f(\omega, x_i, z_i) = |z_i - \omega^T x_i|$ (Linear Regression)

Gradient Method

$$\min F(\omega)$$

$$\omega^{(k+1)} := \omega^k - \alpha_k \nabla F(\omega^k)$$

Stochastic Gradient Descent (SGD)

$$\min \mathbb{E}\left[f(\omega, \xi)\right]$$

Gradient Method

$$\min F(\omega)$$

$$\omega^{(k+1)} := \omega^k - \alpha_k \nabla F(\omega^k)$$

Stochastic Gradient Descent (SGD)

$$\min \mathbb{E}\left[f(\omega, \xi)\right]$$

$$\omega^{k+1} := \omega^k - \alpha_k \nabla \hat{F}(\omega^k)$$

with

$$\nabla \hat{F}(\omega^k) := \frac{1}{b} \sum_{i \in \mathcal{S}_k} \nabla f(\omega^k, x_i, z_i)$$

where $\mathcal{S}_k \subset [N], \quad b := |\mathcal{S}_k| \ll N$
"Mini Batch"

STOCHASTIC QUASI-NEWTON METHOD (SQN)

Stochastic Gradient Descent

$$\min \mathbb{E}\left[f(\omega, \xi)\right]$$

$$\omega^{k+1} := \omega^k - \alpha_k \nabla \hat{F}(\omega^k)$$

$$\nabla \hat{F}(\omega^k) := \frac{1}{b} \sum_{i \in \mathcal{S}_k} \nabla f(\omega^k, x_i, z_i)$$

Stochastic Newton Method

$$\min \mathbb{E}\left[f(\omega, \xi)\right]$$

Stochastic Gradient Descent

$$\min \mathbb{E}\left[f(\omega, \xi)\right]$$

$$\omega^{k+1} := \omega^{k} - \alpha_k \nabla \hat{F}(\omega^{k})$$

$$\nabla \hat{F}(\omega^{k}) := \frac{1}{b} \sum_{i \in \mathcal{S}_k} \nabla f(\omega^{k}, x_i, z_i)$$

Stochastic Newton Method

$$\min \mathbb{E}\left[f(\omega, \xi)\right]$$

$$\omega^{k+1} := \omega^{k} - \alpha_k \nabla^2 \hat{F}(\omega^{k})^{-1} \nabla \hat{F}(\omega^{k})$$

with

$$\nabla^2 \hat{F}(\omega^{k}) := \frac{1}{b_H} \sum_{i \in \mathcal{S}_{H,t}} \nabla^2 f(\omega^{t}, x_i, z_i)$$

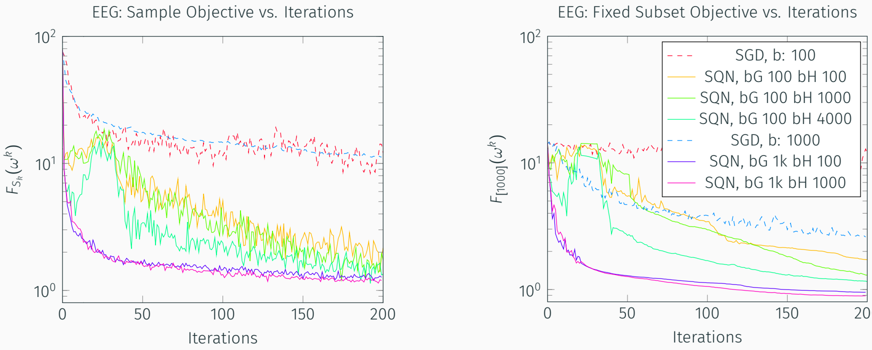
where

$$\mathcal{S}_{H,t} \subset [N], \quad b_H := |\mathcal{S}_{H,t}| \ll N,$$

(t) subsequence of (k)

STOCHASTIC QUASI-NEWTON METHOD (SQN)

- Stochastically use second-order information
- Approximate $\nabla^2 \hat{F}(\omega^k)$ by BFGS matrix H_t
- t running on slower time-scale than k.
- H_t update in $\mathcal{O}(n)$ time and constant memory, using several tricks

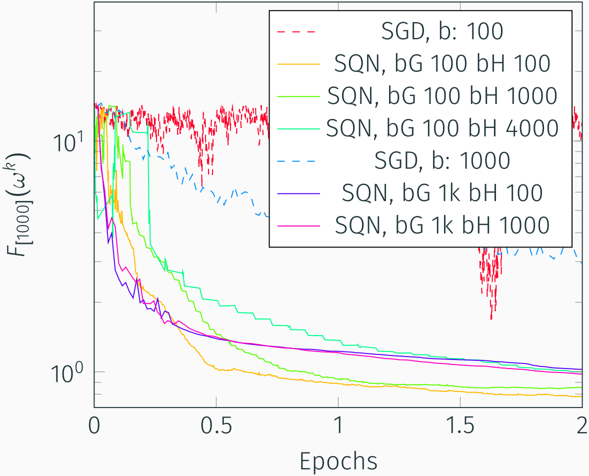


Performance on Logistic Regression, Problem size: 69550×600

Armijo-stepsizes, Further SQN-parameters: $L = 10$, $M = 5$

SQN: PERFORMANCE II

EEG: Fixed Subset Objective vs. Accessed Data Points



Performance on Logistic Regression, Problem size: 69550×600

Armijo-stepsizes, Further SQN-parameters: $L = 10$, $M = 5$

- Can be faster than SGD on appropriate Datasets
- Requires tedious, manual tuning of hyperparameters to be efficient!
- Choice of termination criterion unclear.

PROXIMAL METHOD

Problem $$\min_x F(x) := \underbrace{f(x)}_{smooth} + \underbrace{h(x)}_{non-smooth}$$

Proximity Operator $$\mathrm{prox}_h(v) = \operatorname*{argmin}_x \left(h(x) + \frac{1}{2}\|x - v\|_2^2\right)$$

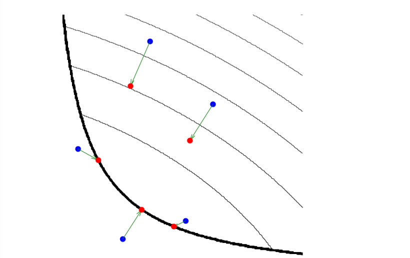


Figure 1: Evaluating a proximal operator at various points. *N Parikh, S Boyd, Proximal Methods, Foundations and Trends in Optimization 1, 2014*

Traditional Proximal Gradient Step:

$$x_{k+1} = \mathrm{prox}_{\lambda_k h}(x_k - \lambda_k \nabla f(x_k))$$

Quasi-Newton Proximal Step:

$$x_{k+1} = \mathrm{prox}_h^{B_k}(x_k - B_k^{-1}\nabla f(x_k)),$$

with $B_k = \underbrace{D_k}_{diag} + \underbrace{u_k}_{\in\mathbb{R}^n} u_k^T$.

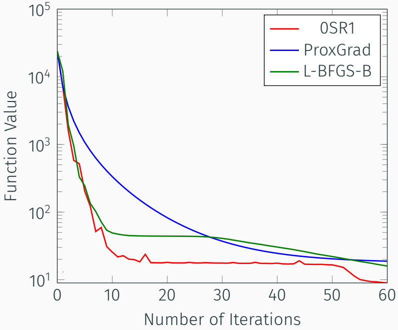
A zero-memory approach is used

PROXIMAL METHOD: PERFORMANCE I

$F(x) = \|Ax - b\| + \lambda \|x\|_1$

$A \in \mathbb{R}^{1500 \times 3000},\ b \in \mathbb{R}^{1500}$

$A_{ij},\ b_i\ \sim \mathcal{N}(0,1),\ \lambda = 0.1$



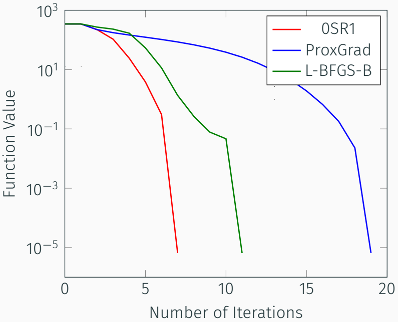
	0SR1	ProxGrad	L-BFGS-B
Iterations	1,822	135,328	1,989
Run-Time	68 s	1,144 s	56 s

$F(x) = \|Ax - b\| + \lambda \|x\|_1$

$A \in \mathbb{R}^{2197 \times 2197},\ b \in \mathbb{R}^{2197}$

A: Discretization of 3D Laplacian

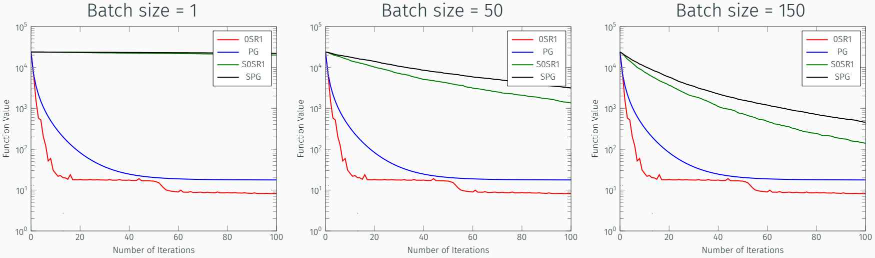
$\lambda = 1$



	0SR1	ProxGrad	L-BFGS-B
Iterations	7	18	10
Run-Time	0.037 s	0.004 s	0.022 s

High-dimensional data: Extension to stochastic framework

Effect of batch size



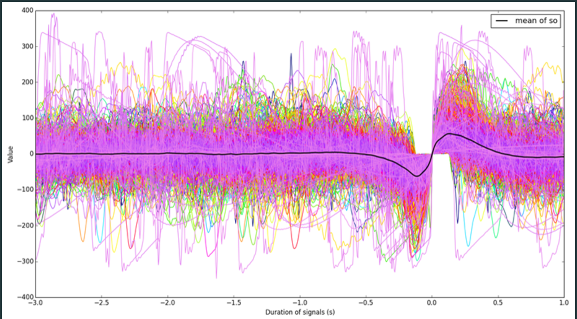
PROXIMAL METHOD: MAIN RESULTS

- Superior results to standard proximal gradient
- Competitive with other standard methods
- Extension to stochastic framework possible
- Applicable to large-scale problems

CLASSIFICATION

Electroencephalography (EEG)

How deep is your sleep?



Sleeping patient / 20 nights of EEG recordings

Predict next slow wave

$$f(\omega, x_i, y_i) = -y_i \log(h(\omega, x_i)) - (1 - y_i)\log(1 - h(\omega, x_i))$$

with

$$h(\omega, x_i) := sigmoid(\omega^T x_i) := \frac{1}{1 + e^{-\omega^T x_i}}$$

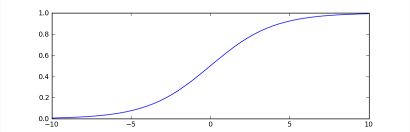


Figure 2: The sigmoid function.

Batch-size	1000, 1000	500, 500
Mean Score	0.8	0.8
Std	0.007	0.006
Running Time	65 s	31 s
M	5	5
L	10	10

	λ=0.1	λ=0.01	λ=0.1	λ=0.01
Batch-size	100	100	1000	1000
Mean Score	0.8	0.67	0.8	0.8
Std	0.01	0.14	0.01	0.016
Running Time	63 s	45 s	68 s	69 s

DICTIONARY LEARNING

Image Denoising

Can we recover the image?

Image is partially destroyed

Reconstruct image

Well-known machine learning model:

$$\min_{D,\alpha} \frac{1}{N} \sum_{i=1}^{N} \| \underbrace{x_i - D\alpha_i}_{\text{a) SQN}} \|_2^2 + \underbrace{\lambda \|\alpha_i\|_1}_{\text{b) Prox}}$$

2-phase optimization problem

1. Update "dictionary"
2. Induce sparsity

$\Rightarrow$ Example: Reconstruction of partially distorted images

Figure 3: Noisy image

Figure 4: Reconstructed image

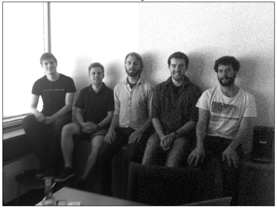
CONCLUSION

SUMMARY

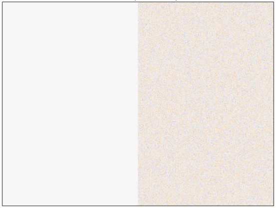
- Large amounts of data
- Need for stochastic algorithms
- Second order methods to improve speed
- For smooth and non-smooth problems
- Good performance of implementation on various problems

Distorted image

Image

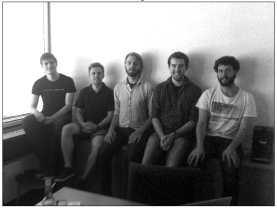


Difference (norm: 59.44)



first try (time: 619.9s)

Image



Difference (norm: 19.65)

Questions?

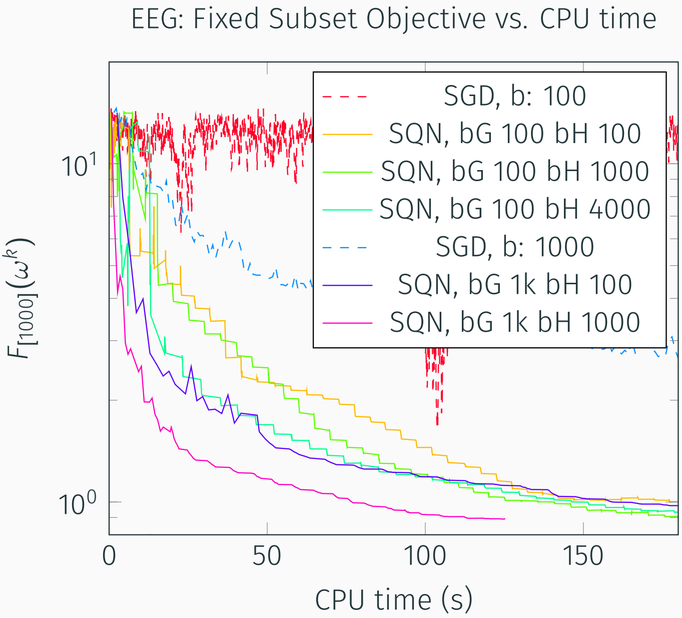
S. Becker and J. Fadili.
A quasi-newton proximal splitting method.
In *Advances in Neural Information Processing Systems*, pages 2618–2626, 2012.

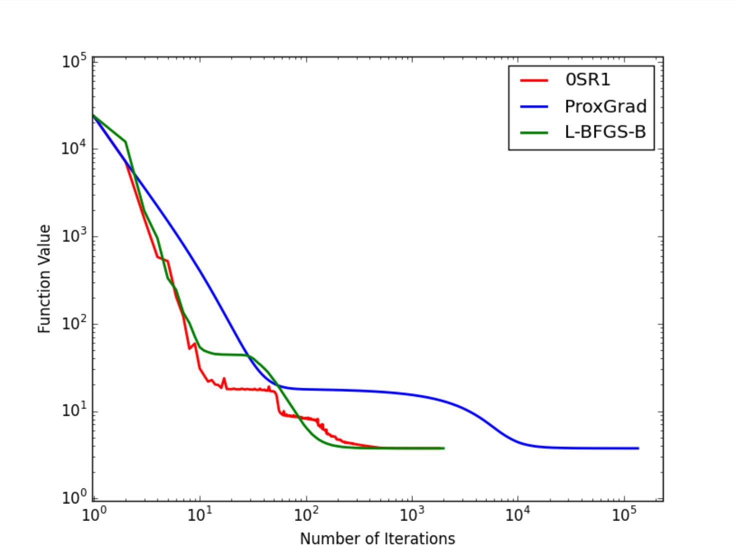
R. H. Byrd, S. Hansen, J. Nocedal, and Y. Singer.
A stochastic quasi-newton method for large-scale optimization.
arXiv.org Preprint: arXiv:1401.7020, 2014.

J. Mairal, F. Bach, J. Ponce, and G. Sapiro.
Online learning for matrix factorization and sparse coding.
The Journal of Machine Learning Research, 11:19–60, 2010.

N. Parikh and S. Boyd.
Proximal algorithms.
Foundations and Trends in optimization, 1(3):123–231, 2013.

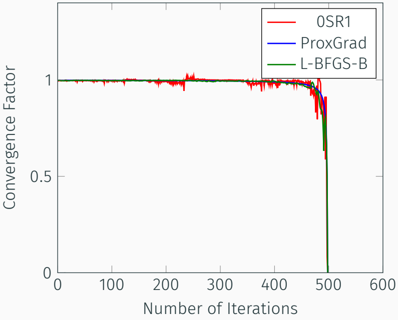
APPENDIX


EEG: Fixed Subset Objective vs. CPU time
SGD, b: 100
SQN, bG 100 bH 100
SQN, bG 100 bH 1000
SQN, bG 100 bH 4000
SGD, b: 1000
SQN, bG 1k bH 100
SQN, bG 1k bH 1000
10^1
10^0
$F_{[1000]}(\omega^k)$
0
50
100
150
CPU time (s)

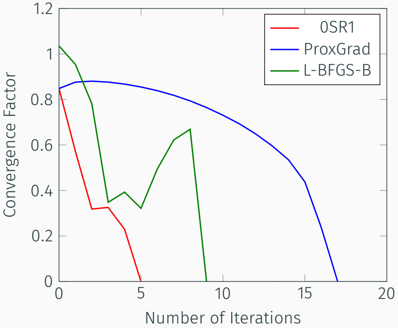


0SR1
ProxGrad
L-BFGS-B
Function Value
Number of Iterations
10^0
10^1
10^2
10^3
10^4
10^5

$F(x) = \|Ax - b\| + \lambda\|x\|_1$
$A \in \mathbb{R}^{1500\times 3000},\ b \in \mathbb{R}^{1500}$
$A_{ij},\ b_i \sim \mathcal{N}(0,1),\ \lambda = 0.1$



$F(x) = \|Ax - b\| + \lambda\|x\|_1$
$A \in \mathbb{R}^{2197\times 2197},\ b \in \mathbb{R}^{2197}$
A: Discretization of 3D Laplacian
$\lambda = 1$



PROXIMAL METHOD

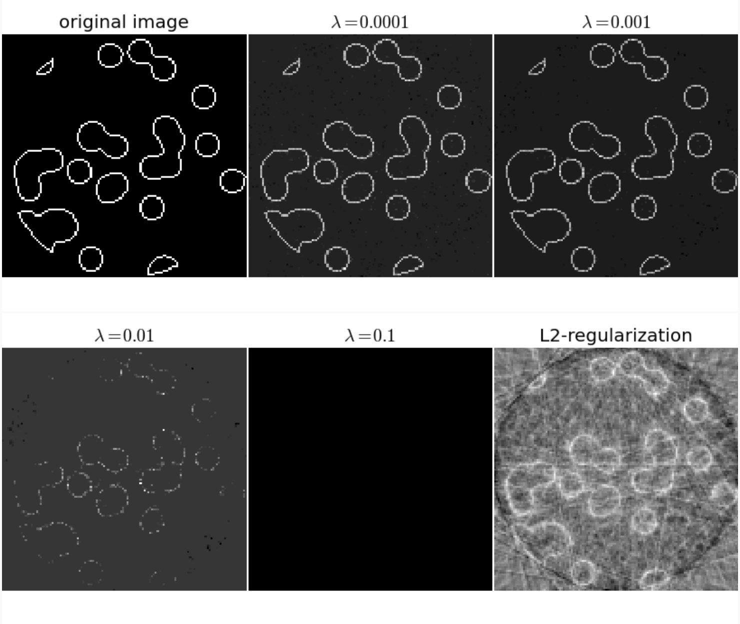


Figure 1: Dictionary learned