STOCHASTIC OPTIMIZATION IN MACHINE LEARNING

Case Studies in Nonlinear Optimization

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

July 10, 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:

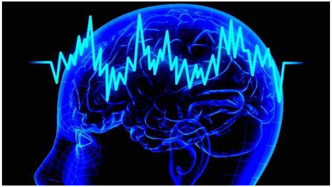
- Discovery of patterns and relationships in data
- Prediction of future events

Examples:

Image Denoising

Section 5

Electroence-phalography (EEG)



Section 4

Training a Machine Learning model means finding optimal parameters ω:

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

Training a Machine Learning model means finding optimal parameters ω:

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

- F: Loss function of chosen ML-model
- X: The training data ($N :=$ #samples $\times$ #features matrix)
- z: Training labels (only in classification models; vector of size N)

Training a Machine Learning model means finding optimal parameters ω:

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

- F: Loss function of chosen ML-model
- X: The training data ($N :=$ #samples $\times$ #features matrix)
- z: Training labels (only in classification models; vector of size N)
- The dimension n of ω is model dependent, often #features+1

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

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

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

CHALLENGES IN MACHINE LEARNING

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

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

Ansatz: Stochastic Methods

STOCHASTIC FRAMEWORK

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

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

- 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.

Gradient Method

$$\min F(\omega)$$

Stochastic Gradient Descent

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

Gradient Method

$$\min F(\omega)$$

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

Stochastic Gradient Descent

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

Gradient Method

$$\min F(\omega)$$

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

Stochastic Gradient Descent

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

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

with

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

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

"Mini Batch"

SQN: A STOCHASTIC QUASI-NEWTON METHOD

- Data from Monte-Carlo simulations
- $X \in \mathbb{R}^{11.000.000 \times 29}$
  Lots of samples, relatively small, dense feature set.
- Here, we use *Logistic Regression* for classification.

STOCHASTIC QUASI-NEWTON METHOD (SQN)

- Stochastically use second-order information
- Based on BFGS-method.

- Stochastically use second-order information
- Based on BFGS-method.
- Basic idea:

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

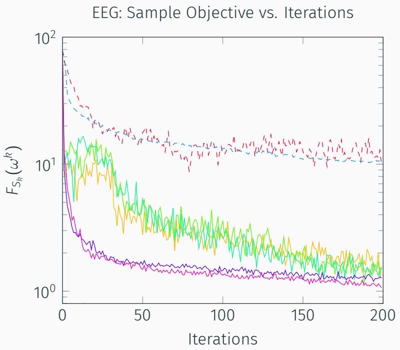
STOCHASTIC QUASI-NEWTON METHOD (SQN)

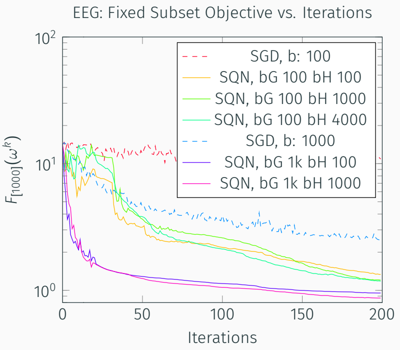
- Stochastically use second-order information
- Based on BFGS-method.
- Basic idea:

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

- t running on slower time-scale than k.
- H_t update in $\mathcal{O}(n)$ time and constant memory, using several tricks

BEHAVIOR


EEG: Sample Objective vs. Iterations
10^2
10^1
10^0
$F_{S_k}(\omega^k)$
0
50
100
150
200
Iterations


EEG: Fixed Subset Objective vs. Iterations
10^2
10^1
10^0
$F_{[1000]}(\omega^k)$
0
50
100
150
200
Iterations
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

RESULTS

- Can be faster than SGD on appropriate Datasets
- Requires tedious, manual tuning of hyperparameters to be efficient!

PROXIMAL METHOD

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

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

Proximity Operator $$\text{prox}_f(v) = \underset{x}{\text{argmin}} \left(f(x) + \frac{1}{2}\|x - v\|_2^2 \right)$$

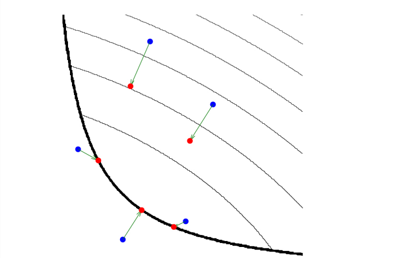


Figure 1: Evaluating a proximal operator at various points. *N Parikh, S Boyd, Proximal Methods, Foundations and Trends in Optimization 1, 2014*

Traditional Proximal Gradient Step:

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

Quasi-Newton Proximal Step:

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

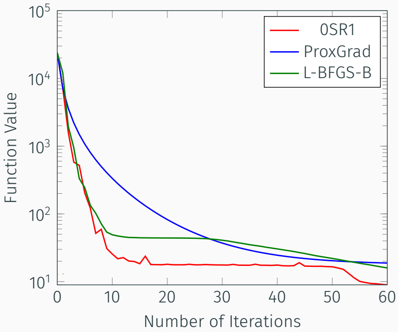
with $B_k = \underbrace{D_k}_{diag} + \underbrace{u_k}_{\in \mathbb{R}^n} u_k^T.$

PROXIMAL METHOD

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

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

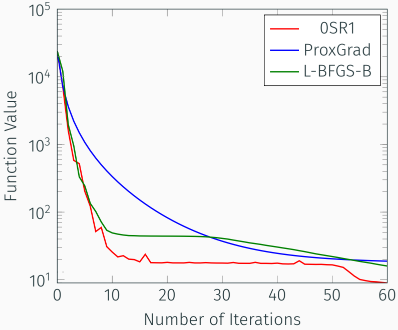
$$A_{ij},\ b_i\ \sim \mathcal{N}(0,1),\ \lambda = 0.1$$



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

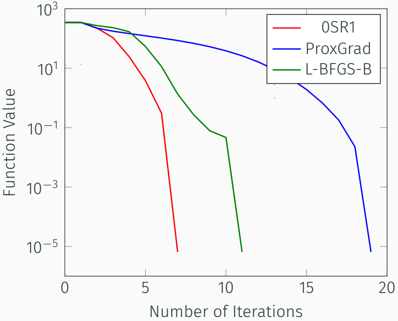
PROXIMAL METHOD

$$F(x) = \|Ax - b\| + \lambda\|x\|_1$$
$$A \in \mathbb{R}^{1500\times3000},\ b \in \mathbb{R}^{1500}$$
$$A_{ij},\ b_i\ \sim \mathcal{N}(0,1),\ \lambda = 0.1$$



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

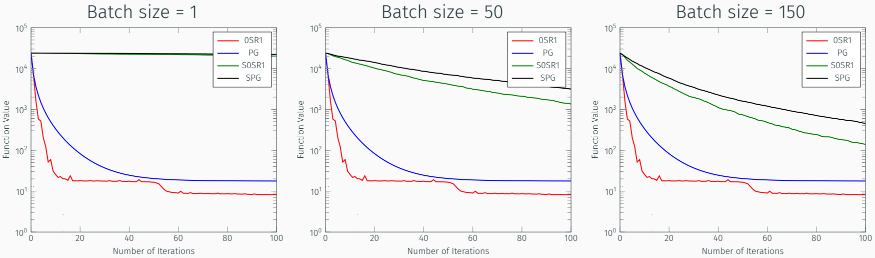
$$F(x) = \|Ax - b\| + \lambda\|x\|_1$$
$$A \in \mathbb{R}^{2197\times2197},\ b \in \mathbb{R}^{2197}$$
A: Discretization of 3D Laplacian
$$\lambda = 1$$



	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

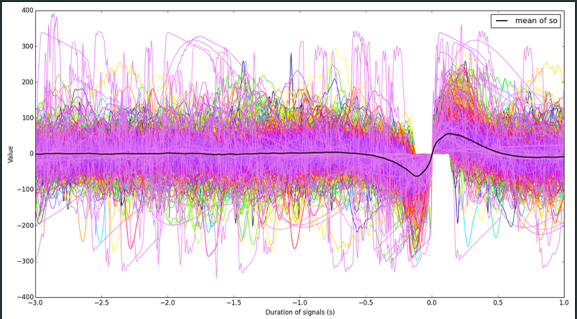
Effect of batch size



LOGISTIC REGRESSION: AN EXAMPLE

Electroencephalography (EEG)

How deep is your sleep?



Sleeping patient / 20 nights of EEG recordings

Predict next slow wave

RESULTS

Nice table with SQN, SGD (no reg, L2), (Lasso,) Prox (L1) showing Obj. value in found optimum, CPU time, Iterations, F1 score of prediction model

	$F(\omega^*)$	**Model Score**	**Cost**
No regularization			
SGD	0.01	96%	x sec, y AP
SQN	0.5	96%	x sec, y AP
Prox	0.01	96%	x sec, y AP
L1			
LASSO	.71	55%	blablabla
Prox	0.01	96%	x sec, y AP
L2			
SGD	.71	55%	blablabla
SQN	0.01	96%	x sec, y AP

DICTIONARY LEARNING

Dictionary Learning

Can we recover the image?

Image is partially destroyed

Reconstruct image

bla

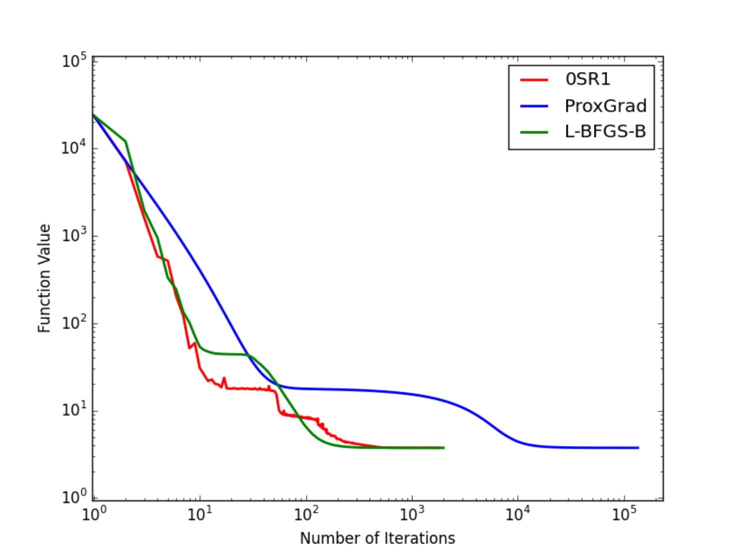
CONCLUSION

SUMMARY

Questions?

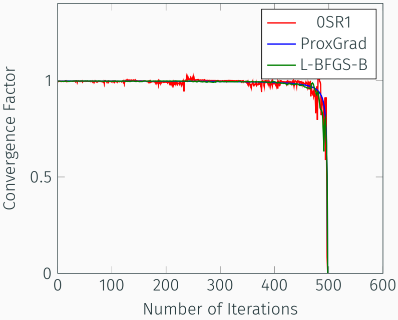
S. Becker and J. Fadili.
A quasi-newton proximal splitting method.
In *Advances in Neural Information Processing Systems*, pages 2618–2626, 2012.

APPENDIX



Function Value
Number of Iterations
0SR1
ProxGrad
L-BFGS-B
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
$\lambda = 1$