

# STOCHASTIC OPTIMIZATION IN MACHINE LEARNING

## CASE STUDIES IN NONLINEAR OPTIMIZATION

---

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

July 7, 2015

Technische Universität München

*WE'RE NOT RUNNING OUT OF DATA ANYTIME  
SOON. IT'S MAYBE THE ONLY RESOURCE THAT  
GROWS EXPONENTIALLY.*

*ANDREAS WEIGEND*

1. Introduction
2. SQN: A Stochastic Quasi-Newton Method
3. Proximal Method
4. Logistic Regression: An Example
5. Conclusion

## INTRODUCTION

---

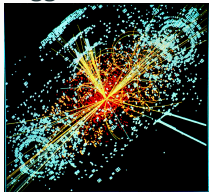
# WHAT IS MACHINE LEARNING?

Implementation of autonomously learning software for:

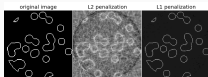
- Discovery of patterns and relationships in data
- Prediction of future events

Examples:

Higgs-Boson



Compressed Sensing



EEG

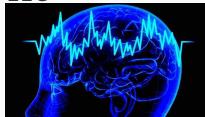


Image Reconstruction



Training a Machine Learning model means finding optimal parameters  $\omega$ :

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

Training a Machine Learning model means finding optimal parameters  $\omega$ :

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

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

Training a Machine Learning model means finding optimal parameters  $\omega$ :

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

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



After we have found  $\omega^*$ , we can do **Prediction** on new data points:

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

After we have found  $\omega^*$ , 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

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

Ansatz: Stochastic Methods

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

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

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

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

- $\xi$ : 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

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



## Gradient Method

$$\min F(\omega)$$

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

## Stochastic Gradient Descent

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

## Gradient Method

$$\min F(\omega)$$

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

## Stochastic Gradient Descent

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

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

with

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

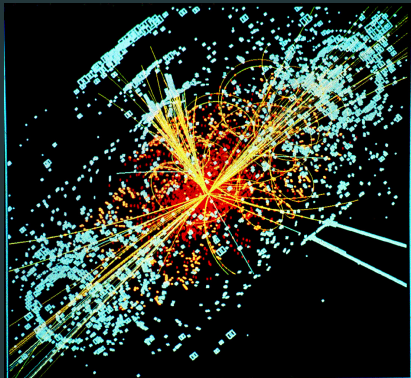
where  $\mathcal{S}_k \subset [N]$ ,  $b := |\mathcal{S}_k| \ll N$   
"Mini Batch"

# SQN: A STOCHASTIC QUASI-NEWTON METHOD

---

# CLASSIFICATION

DID WE JUST DETECT A HIGGS-BOSON?



- 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.

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

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

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

- 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



Pretty pictures about the behaviour of SQN on HIGGS and comparison with traditional SGD

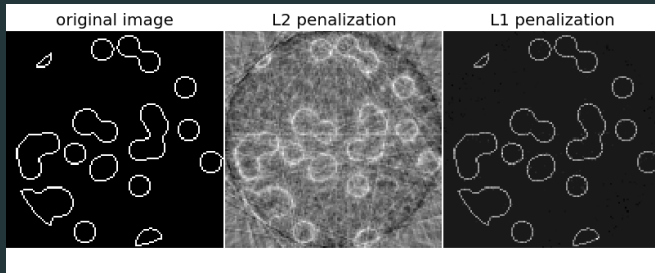
- Can be faster than SGD on appropriate Datasets
- Requires tedious, manual tuning of hyperparameters to be efficient!

## PROXIMAL METHOD

---

# IMAGE RECONSTRUCTION

WHAT DID THE ORIGINAL IMAGE LOOK LIKE?



Problem

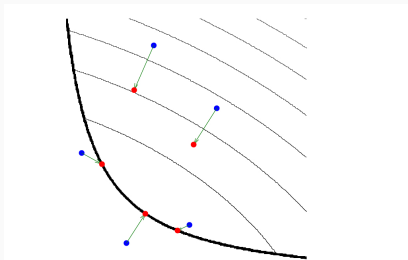
$$\min_x F(x) := \underbrace{f(x)}_{\text{smooth}} + \underbrace{h(x)}_{\text{non-smooth}}$$

## Problem

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

## Proximity Operator

$$\text{prox}_f(v) = \arg\min_x \left( f(x) + \frac{1}{2} \|x - v\|_2^2 \right)$$



Traditional Proximal Gradient Step:

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

Quasi-Newton Proximal Step:

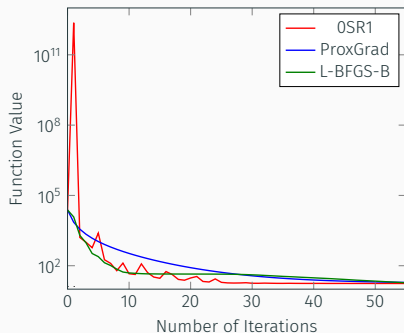
$$x_{k+1} = \text{prox}_h^{B_k}(x_k - B_k^{-1} \nabla f(x_k)),$$

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

# PROXIMAL METHOD

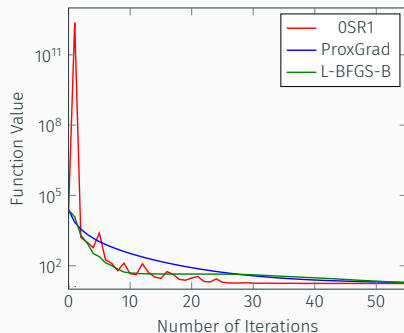
$$F(x) = \|Ax - b\| + \lambda \|x\|_1$$

$A \in \mathbb{R}^{1500 \times 3000}$ ,  $b \in \mathbb{R}^{1500}$   
 $A_{ij}, b_i \sim \mathcal{N}(0, 1)$ ,  $\lambda = 0.1$



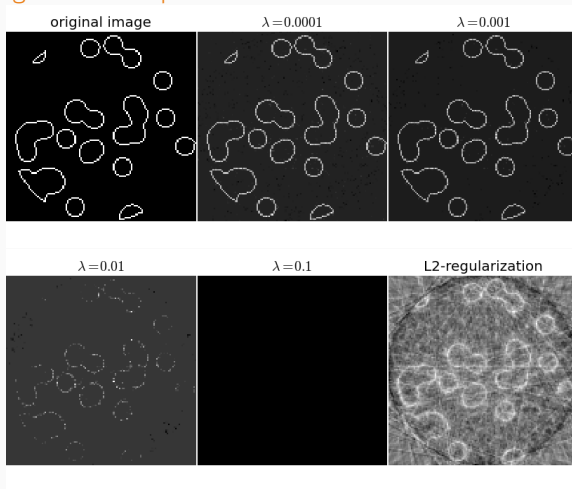
$$F(x) = \|Ax - b\| + \lambda \|x\|_1$$

$A \in \mathbb{R}^{1500 \times 3000}$ ,  $b \in \mathbb{R}^{1500}$   
 $A_{ij}, b_i \sim \mathcal{N}(0, 1)$ ,  $\lambda = 0.1$





Effect of regularization parameter  $\lambda$  on solution:



## LOGISTIC REGRESSION: AN EXAMPLE

---

Explain what we want to do, and explain the dataset, and why using both SQN and Prox makes sense

## 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

	$F(\omega^*)$	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

## CONCLUSION

---



QUESTIONS?





S. Becker and J. Fadili.

**A quasi-newton proximal splitting method.**

*In Advances in Neural Information Processing Systems*, pages 2618–2626, 2012.