# stochastic optimization in machine learning

Case Studies in Nonlinear Optimization

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

July 9, 2015

Technische Universitt Mnchen

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

*Andreas Weigend*

# outline

# introduction

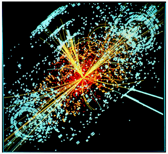
# introduction: what is machine learning?

Implementation of autonomously learning software for:

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

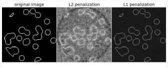
Examples:

Higgs-Boson



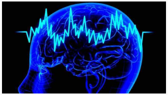
Section 2

Computed Tomography (CT)




Section 3

Electroencephalography (EEG)



Section 4

Image Denoising

Section 5

# ml and optimization i

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

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

# ml and optimization i

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 ($N :=$ #samples $\times$ #features matrix)
- $z$: Training labels (only in classification models; vector of size $N$)

# ml and optimization i

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 ($N :=$ #samples $\times$ #features matrix)
- $z$: Training labels (only in classification models; vector of size $N$)
- The dimension $n$ of $\omega$ is model dependent, often #features+1

# ml and optimization ii

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

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

# ml and optimization ii

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

# challenges in machine learning

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

# challenges in machine learning

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

Ansatz: Stochastic Methods

# stochastic framework

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

# stochastic framework

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

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

# stochastic framework

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

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

# stochastic methods

**Gradient Method**

$$\min F(\omega)$$

**Stochastic Gradient Descent**

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

# stochastic methods

**Gradient Method**

$$\min F(\omega)$$

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

**Stochastic Gradient Descent**

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

# stochastic methods

## 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

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

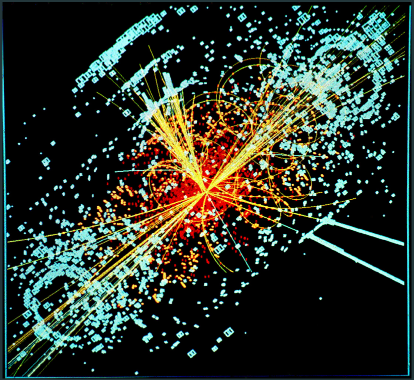
where $\mathcal{S}_k \subset [N], \quad b := |\mathcal{S}_k| \ll N$

"Mini Batch"

# sqn: a stochastic quasi-newton method

# Classification

## Did we just detect a Higgs-Boson?



# higgs-boson classification problem

- 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.

# stochastic quasi-newton method (sqn)

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

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

# 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

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

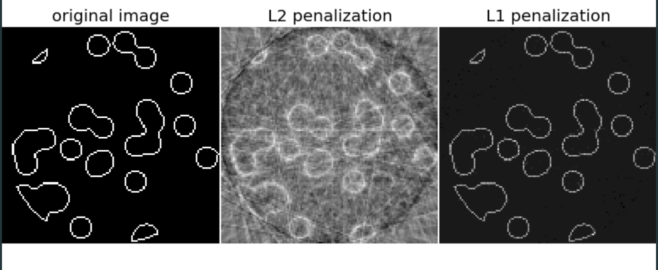
# results

- 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?



# proximal method

**Problem**

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

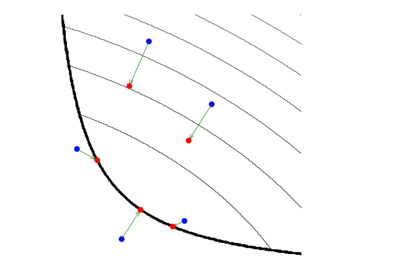
# proximal method

**Problem**

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

**Proximity Operator**

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



# proximal method

Traditional Proximal Gradient Step:

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

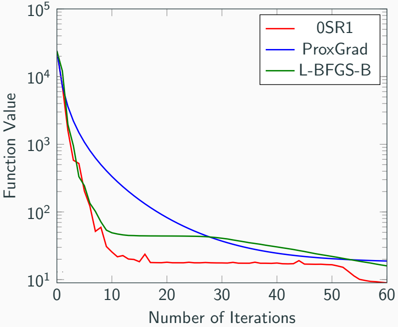
Quasi-Newton Proximal Step:

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

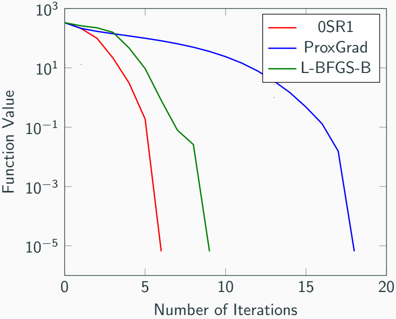
with $B_k = \underbrace{D_k}_{diag} + \underbrace{u_k}_{\in \mathbb{R}^n} u_k^T$.

# proximal method

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

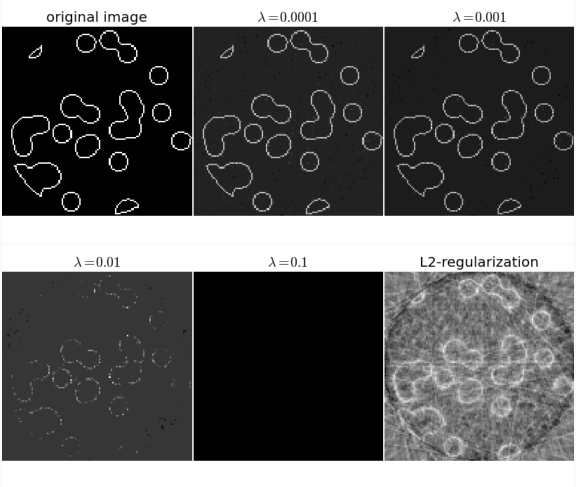



$F(x) = \|Ax - b\| + \lambda\|x\|_1$
$A \in \mathbb{R}^{2197\times 2197},\ b \in \mathbb{R}^{2197}$
$A$ from 7-point finite difference stencil for 3D Laplacian on a Box
$\lambda = 1$



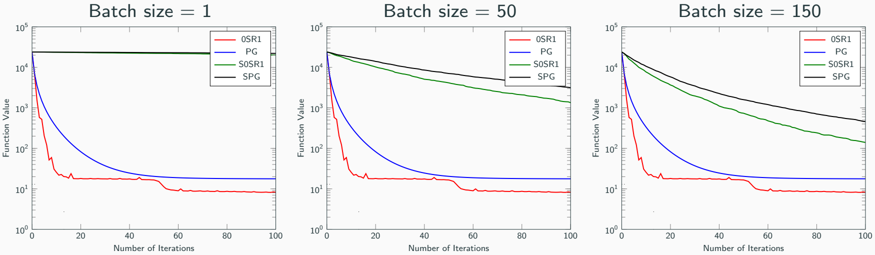
# proximal method

Effect of regularization parameter $\lambda$ on solution:



High-dimensional data: Extension to stochastic framework

## Effect of batch size



# logistic regression: an example

# task

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

# eeg data

Recording:

- eeg signal
- 20 nights from healthy patient
- each almost 10 hours
- 1 eeg channel, 200 Hz

Figure 1: eeg channels

# detection of slow oscillations

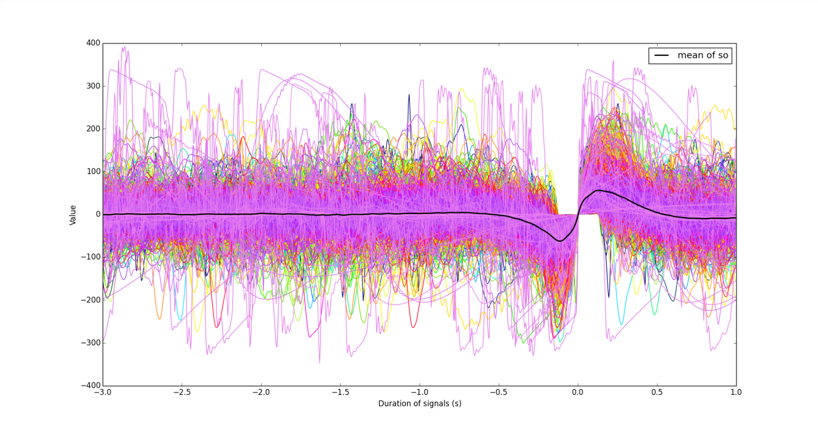



Figure 2: Slow oscillations for one subject

# the classification problem - roc auc metrics

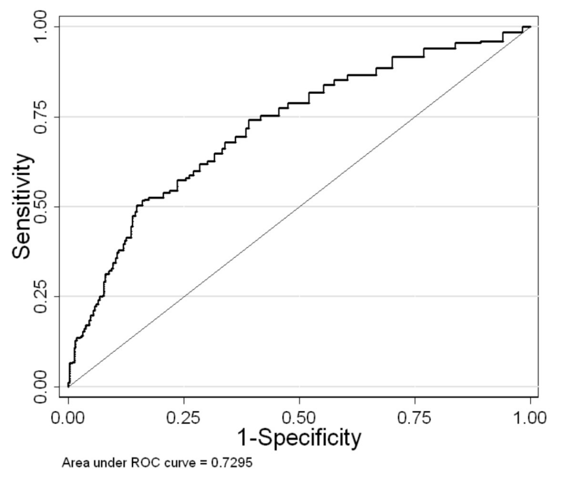



Figure 3: AUC metrics

# 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

# summary

Questions?

# main references I