SNM: Stochastic Newton Method for Optimization of Discrete Choice Models

Gael Lederrey
PhD student @ TRANSP-OR, EPFL
http://glederrey.ch
glederrey
gael.lederrey@epfl.ch

with
Pr. Virginie Lurkin (EPFL-TUE)
Pr. Michel Bierlaire (EPFL)

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

TRANSP-OR

7th of November 2018

Outline

1. Motivation
2. State-of-the-art
3. Optimization of DCMs
4. Future work
5. Conclusion

Paper

Motivation

DCMs in a nutshell (1/2)

- **Discrete Choice** Theory (McFadden, 2000)
- Economics, Transportation, Health, …

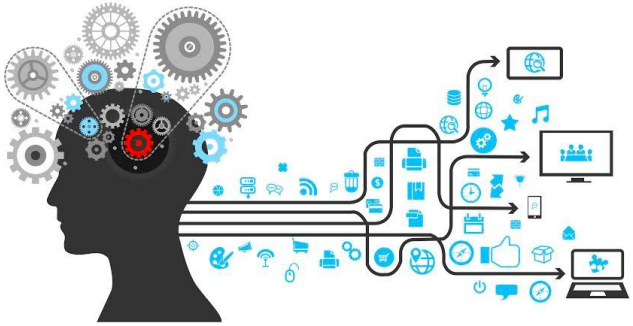
- Softwares: Biogeme, Larch, etc.

Interested? Have a look at **our MOOC on edX!** (Search "*Discrete Choice*")
https://courses.edx.org/courses/course-v1:EPFLx+DiscreteChoiceX+3T2017/course/

DCMs in a nutshell (2/2)

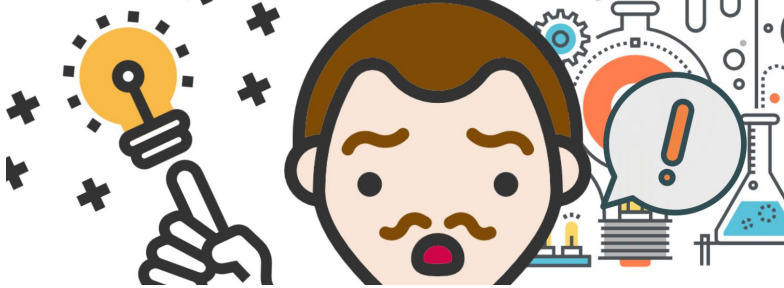
- Many success stories until now...

Solution: Machine Learning



- ML is **THE** field dealing with a lot of data!

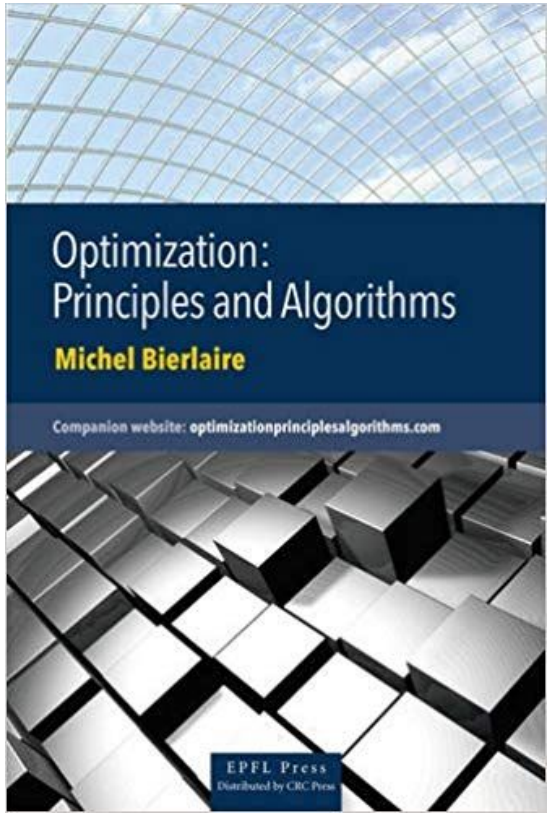
What can we do for DCMs?



1. Faster optimization of DCMs.

State-of-the-art

Optimization of DCM and ML



First-Order optimization - the ancestors

GD

(Cauchy, 1847)

Specificities:
Gradient computed on **all** the data

Update step:

$$\theta = \theta - \alpha \cdot \nabla_\theta f(\theta; x)$$

Where

θ: Parameters

α: Step size

f: Function, $f \in C^1(\mathbb{R}^n)$

x: Data, $x \in \mathbb{R}^n$

SGD

(???, 1940's)

Specificities:
Gradient computed on **only one** data

Update step:

$$\theta = \theta - \alpha \cdot \nabla_\theta f(\theta; x_i)$$

Where

θ: Parameters

α: Step size

f: Function, $f \in C^1(\mathbb{R}^n)$

x: Data, $x \in \mathbb{R}^n$

mbSGD

(???, 1940's)

Specificities:
Gradient computed on **a batch of** data

Update step:

$$\theta = \theta - \alpha \cdot \nabla_\theta f(\theta; x_{\sigma(k)})$$

Where

θ: Parameters

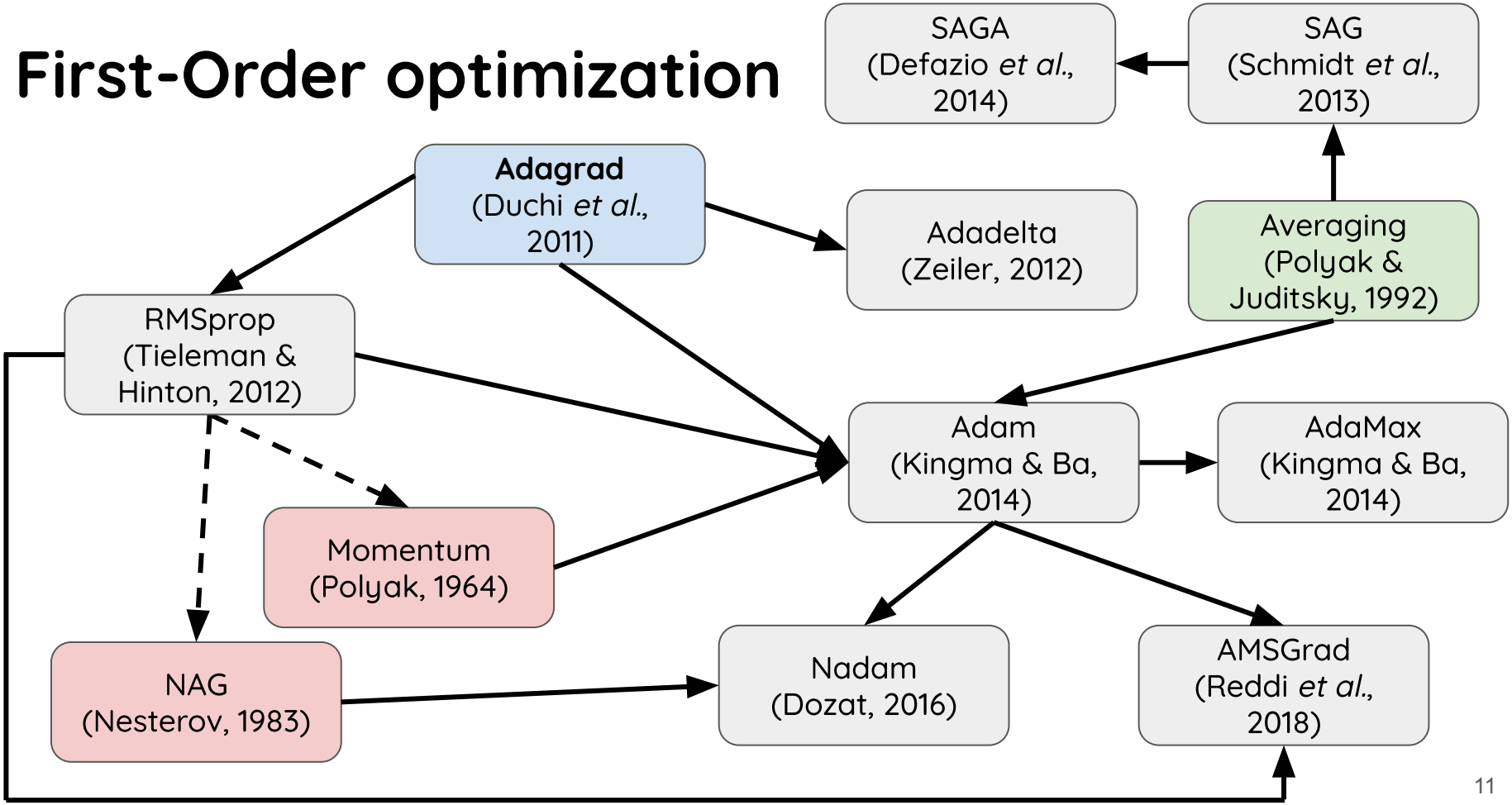
α: Step size

f: Function, $f \in C^1(\mathbb{R}^n)$

x: Data, $x \in \mathbb{R}^n$

$\sigma(k)$: Choice of k indices

First-Order optimization



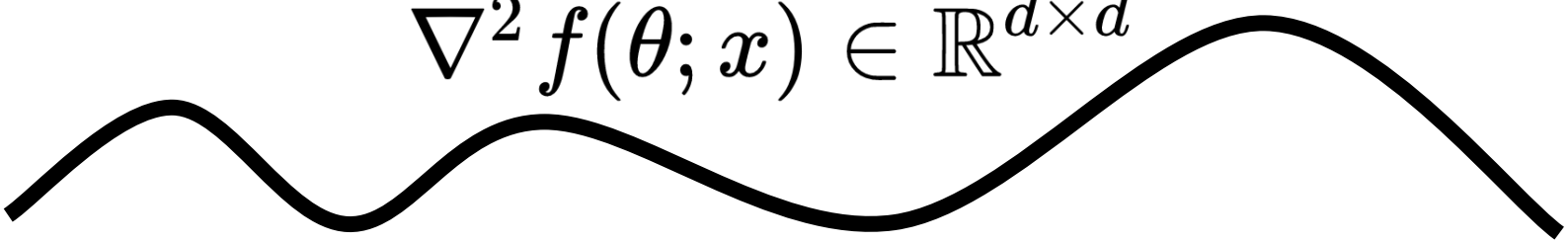
First-Order vs Second-Order

- Gradient is pretty cheap to compute

$$\nabla_\theta f(\theta; x) \in \mathbb{R}^d$$

- Computation of Hessian is difficult/impossible

$$\nabla^2 f(\theta; x) \in \mathbb{R}^{d \times d}$$



- Recently, more work on quasi-Newton methods.

  Gower *et al.* (2018), Martens (2010), Bordes *et al.* (2009-2010)

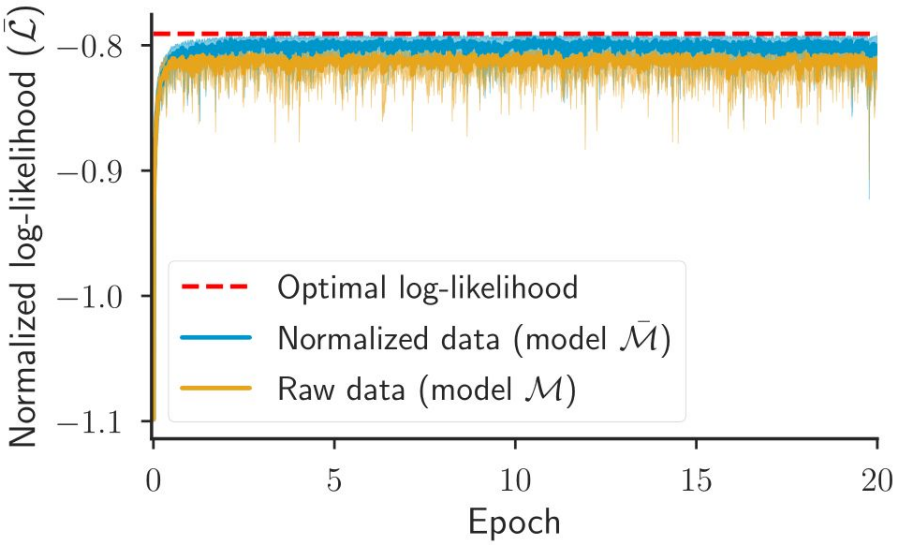
Optimization of DCMs

Simple MNL

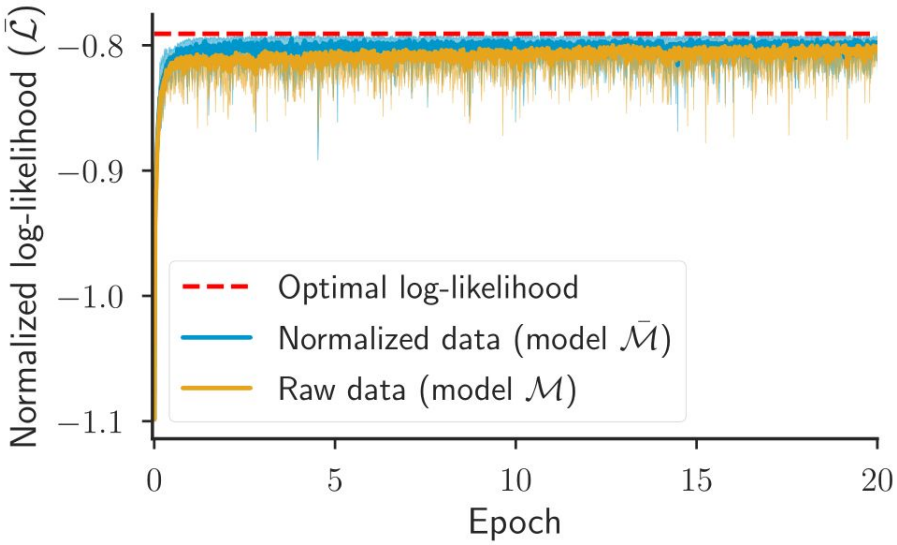
- Logit Model with 10 parameters (1 fixed)
- Normalized and raw observations (#9,036)
- **Normalized LL: –0.7908**

  (Biogeme: ~10 epochs)

First-step - First-order



SGD

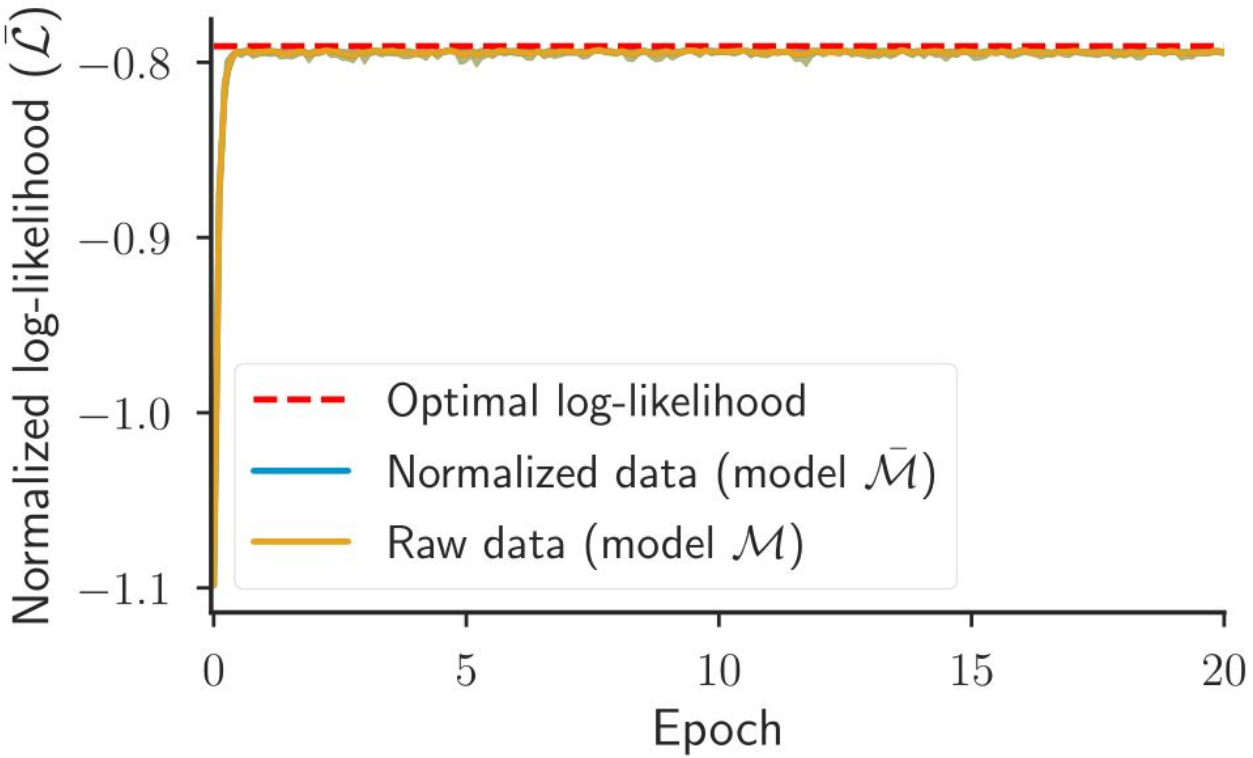


Adagrad

Why not Second-Order?

- Small number of parameters! (≠ Deep Learning)

SNM: The good, [...]


Normalized log-likelihood ($\bar{\mathcal{L}}$)
−0.8
−0.9
−1.0
−1.1
0
5
10
15
20
Epoch
Optimal log-likelihood
Normalized data (model $\bar{\mathcal{M}}$)
Raw data (model $\mathcal{M}$)

SNM: The bad, [...]

- Shift between Newton step and Gradient step
  => Conjugate Gradient

SNM: and the ugly!

- Still not able to converge!

~~Future~~ work

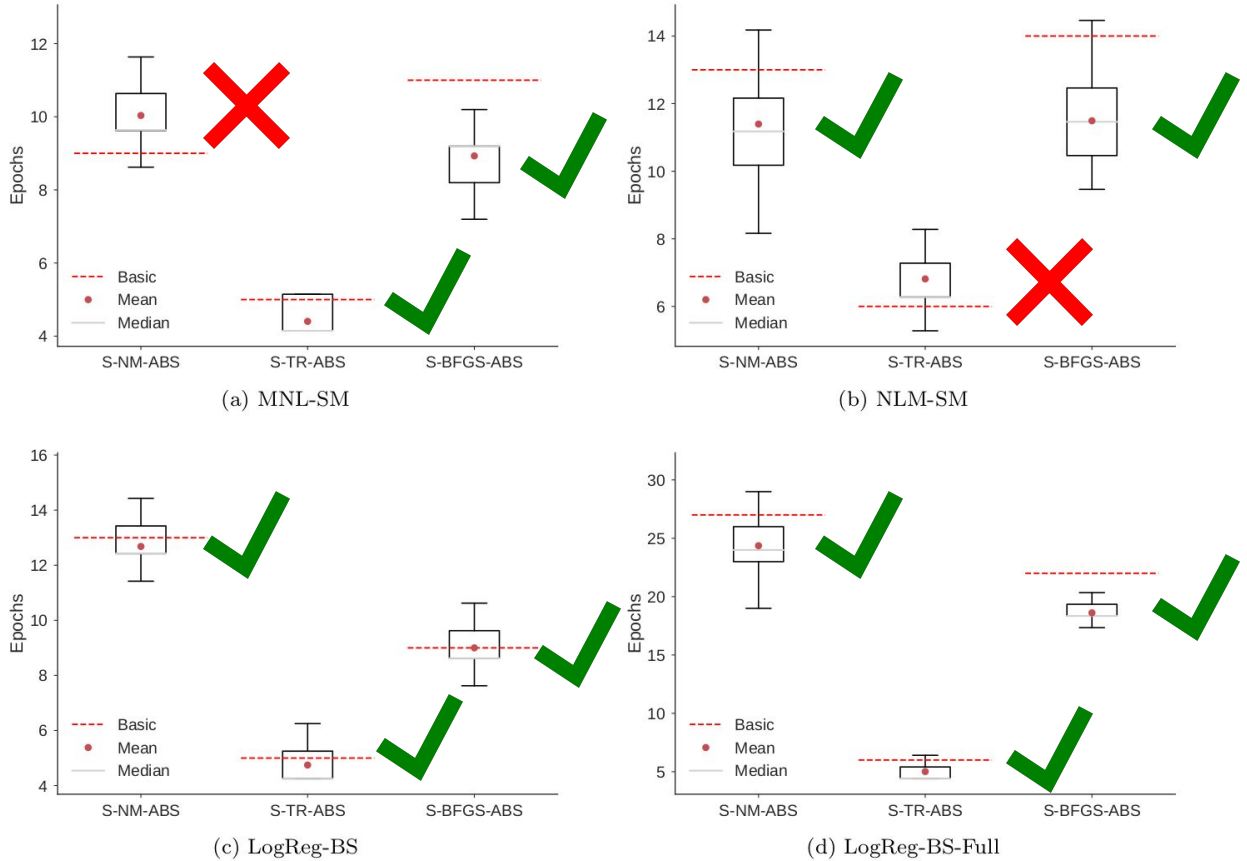
Current

Adaptive Batch Size (ABS)

coming soon!

Results

Better: 10/12



(a) MNL-SM (b) NLM-SM

(c) LogReg-BS (d) LogReg-BS-Full

Conclusion

Conclusion

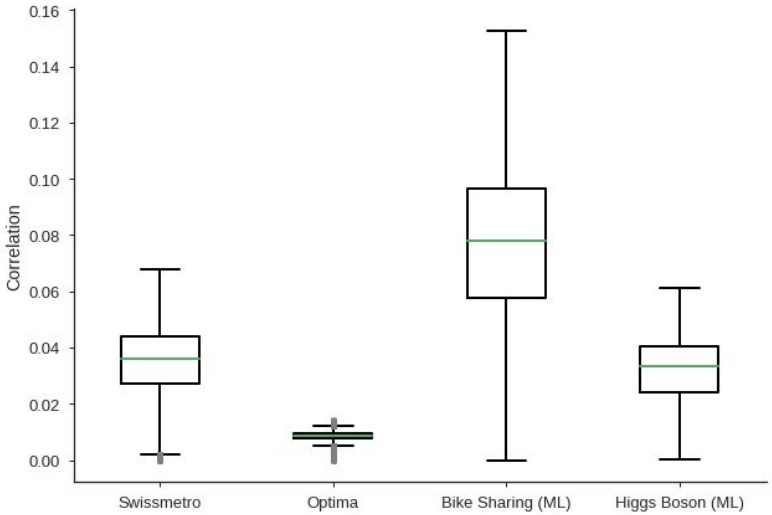
- Estimation of DCMs has to be improved!
- Optimization is more tricky than expected!
- But it’s working better everyday!

Mahalo

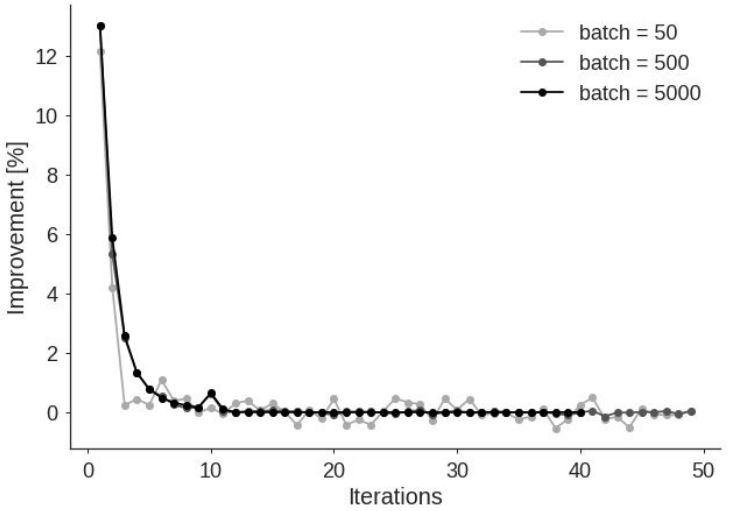
Back up Slides

SNM not converging

- Data is not redundant enough!



DCM ML



Improvement

SNM

Algorithm 1 Stochastic Newton Method (SNM)

Input: Initial parameter value (β_0), data ($\mathcal{D}$), function (f), gradient (∇f), Hessian ($\nabla^2 f$), #epochs (N_{ep}), batch size (N_{batch})
Output: Epochs (E), parameters (β), function values (f_v)

function SNM
  $(N_{\mathcal{D}}, M) = |\mathcal{D}|$ ▷ Number of samples and parameters
  $N_{iter} \leftarrow \lceil N_{ep} N_{\mathcal{D}} / N_{batch} \rceil$ ▷ Number of iterations
  Initialize E, β and f_v. Set $\beta[0] \leftarrow \beta_0$
  for $i = 0 \dots N_{iter}$ **do**
    $E[i] \leftarrow i \cdot N_{batch} / N_{\mathcal{D}}$ ▷ Store the epoch
    $f_v[i] \leftarrow f(\beta[i])$ ▷ Store the function value
    `idx` $\leftarrow N_{batch}$ indices from $\mathcal{U}(0, N_{\mathcal{D}})$ without replacement
    `grad` $\leftarrow \nabla f_{\texttt{idx}}(\beta[i])$ ▷ Gradient on the samples from `idx`
    `hess` $\leftarrow \nabla^2 f_{\texttt{idx}}(\beta[i])$ ▷ Hessian on the samples from `idx`
    if `hess` is negative definite **then**
      Solve `hess` $\cdot$ `step` $= -$`grad` to get `step` ▷ Newton step
    else
      `step` $\leftarrow$ `grad` ▷ Gradient step
    $\alpha \leftarrow$ Backtracking Line Search with `step` on the subset of data with indices from `idx`
    $\beta[i+1] \leftarrow \beta[i] + \alpha \cdot$ `step`
  $E[n_{iter}] \leftarrow N_{iter} \cdot N_{batch} / N_{\mathcal{D}}$
  $f_v[N_{iter}] \leftarrow f(\theta[N_{iter}])$
  return E, β and f_v