SNM: Stochastic Newton Method for Optimization of Discrete Choice Models

Gael Lederrey, Virginie Lurkin and Michel Bierlaire

Submitted at IEEE ITSC 2018

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

TRANSP-OR

STRC | 18th Swiss Transport Research Conference
Monte Verità / Ascona, May 16-18, 2018

Optimization of Discrete Choice Models?

DCMs Softwares

Virginie

- Larch (Newman *et al.*, 2018)
- MNLogit (`statsmodel`, Python)
- Biogeme (Bierlaire, 2003)

No Stochasticity

Motivation

Twitter users sent
456,000 tweets

Youtube users
uploaded
400 hours of video

Every minute of the Day (2017)

BIG DATA

Instagram users
shared **46,740**
pictures

Google received over
3,600,000 search queries

What can we do?

1. ~~Avoid using more data~~
   (Do we really need more data?)

2. ~~Use more powerful computers~~
   (Do you know about Moore's law?)

3. ~~Stop using DCMs and use ML~~
   (Basically, it's the same thing, no?)

4. Actually do something about DCMs

Where to get inspiration?

- Machine Learning is the obvious choice!
  - Emerging since 1950's
  - Lot's of work on Optimization thanks to Neural Networks
  - They make use of data (data-driven)
    => they know how to deal with data!

ML is actually "close" to DCMs

DCMs	v.s.	ML
Model-driven		Data-driven
Main goal: Understand behavior		Main goal: Prediction
Can also predict		Can also help to understand behavior
Likelihood as objective function		Likelihood (possible) as objective function
Optimization is very important!		Optimization is very important!

Optimization of ML - The Basics

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

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

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

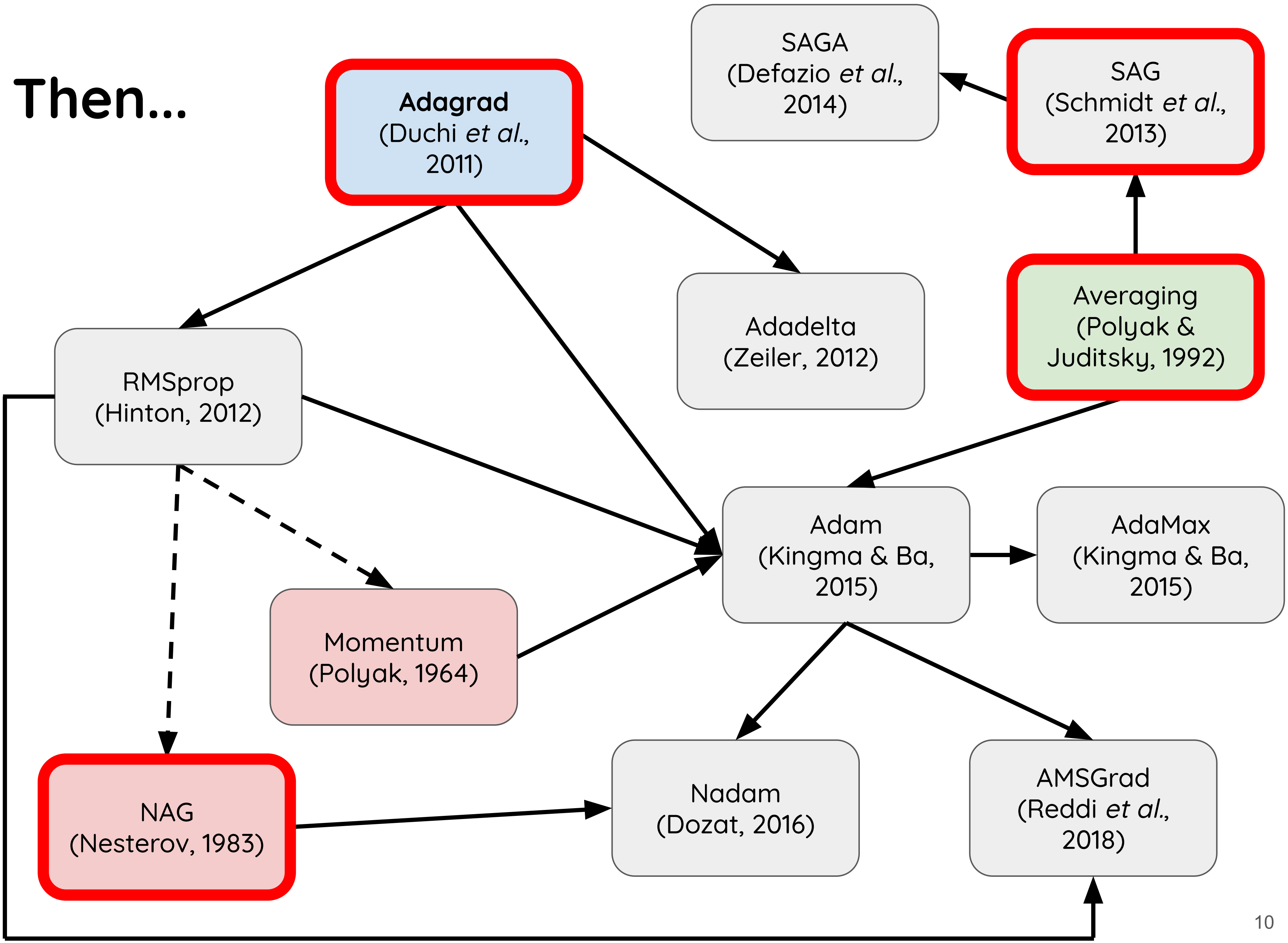
x : Data, $x \in \mathbb{R}^n$

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

Challenges

- Choosing a proper step size
- Same step size applies to all parameter updates
- **Avoid getting trapped in a local minima!**
  (For non-convex functions)

Then...


Adagrad
(Duchi *et al.*, 2011)
SAGA
(Defazio *et al.*, 2014)
SAG
(Schmidt *et al.*, 2013)
RMSprop
(Hinton, 2012)
Adadelta
(Zeiler, 2012)
Averaging
(Polyak & Juditsky, 1992)
Momentum
(Polyak, 1964)
Adam
(Kingma & Ba, 2015)
AdaMax
(Kingma & Ba, 2015)
NAG
(Nesterov, 1983)
Nadam
(Dozat, 2016)
AMSGrad
(Reddi *et al.*, 2018)

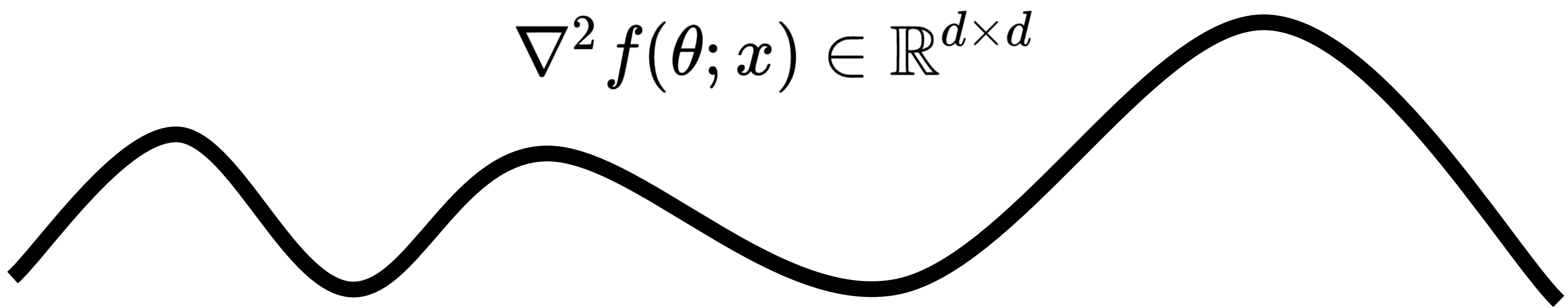
First-Order vs Second-Order

- Gradient is pretty cheap to compute

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

- Computation of Hessian is difficult/impossible

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

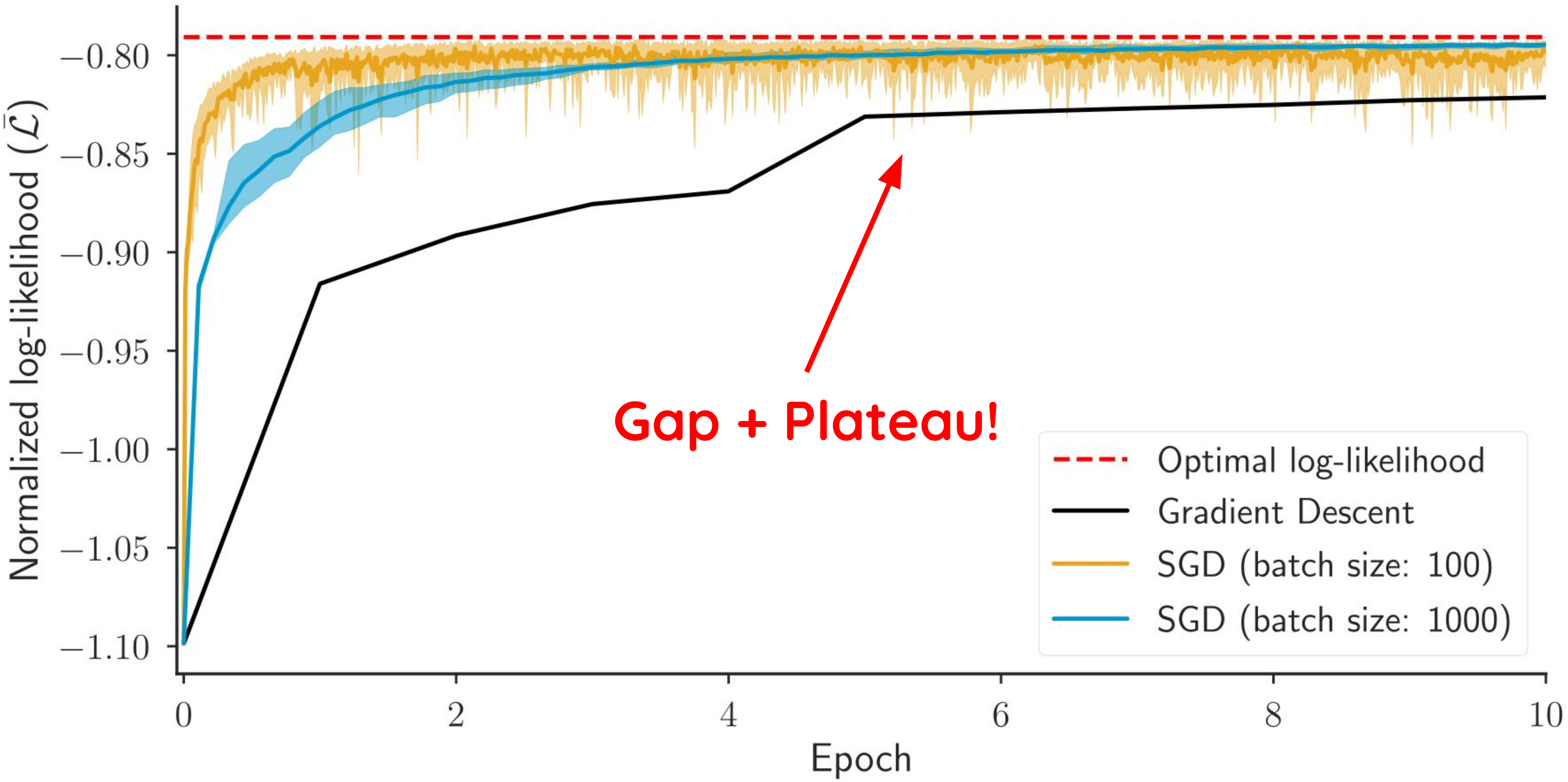


- Recently, more work on quasi-Newton methods.

What am I doing here?

- Build a simple MNL (11 parameters) on *Swissmetro*

First-Order stochastic algorithms



How to fix that?

Stochastic Newton Method

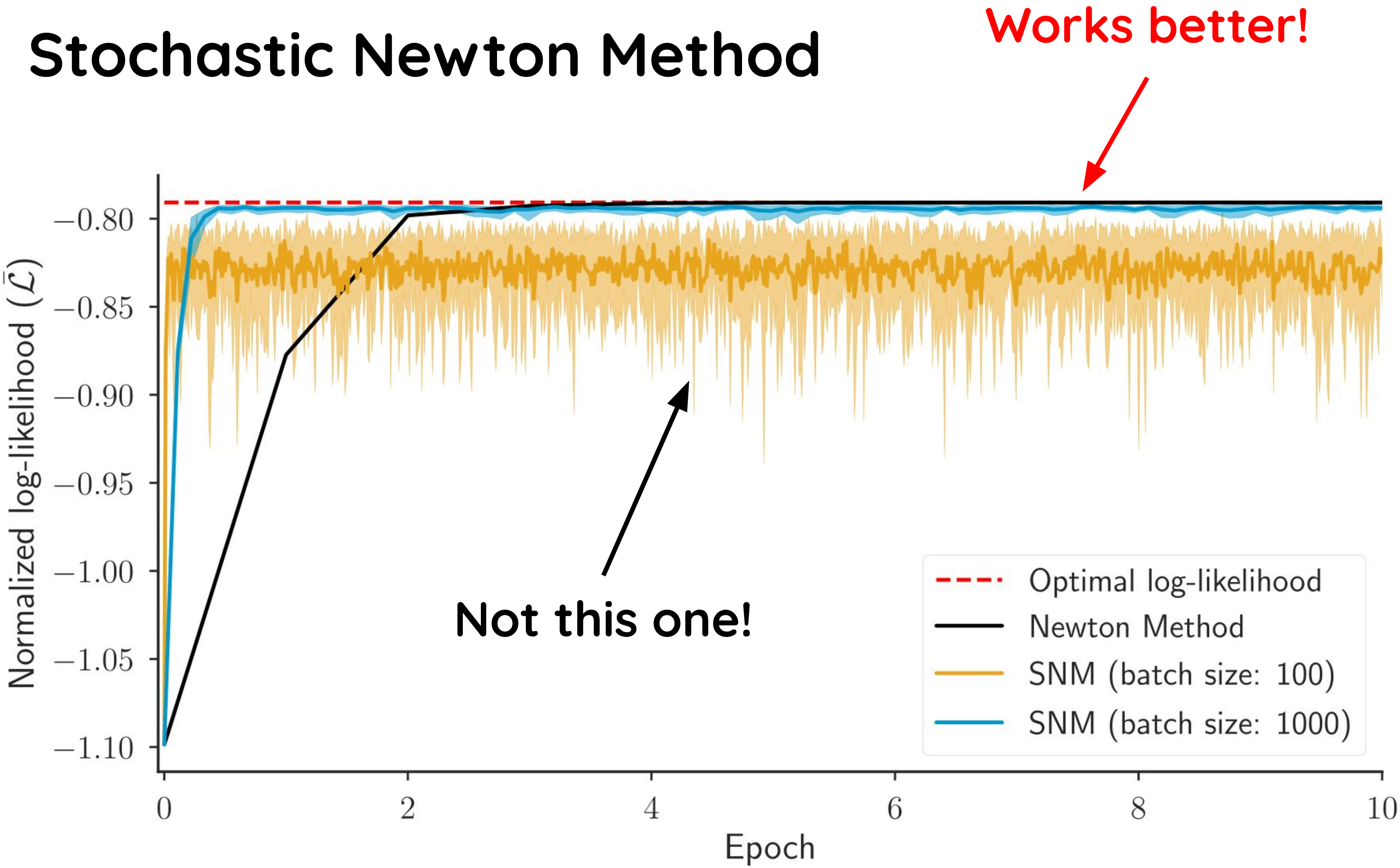
Work with batch of data

Make Newton step if possible.
(Hessian positive/negative definite)

Otherwise ...

Make Gradient step

Stochastic Newton Method



Conclusion

- We need more data but we can't process them.
- Works need to be done on the optimization side!
- Promising early results on a simple algorithm

Future work

- Use the same iterative strategy to develop SNM
  => Variance reduction, Accelerated, etc.

- Add Conjugate Gradient and Trust-Region methods

- Test on non-convex models (Nested logit models)

Thank you!

Article submitted at IEEE ITSC 2018

http://github.com/glederrey/IEEE2018-SNM

Code and article also available upon request:

gael.lederrey@epfl.ch

Backup Slides

Main extensions of SGD

Momentum

$$\nu_t = \gamma \nu_{t-1} + \alpha \nabla_\theta f(\theta; x)$$
$$\theta = \theta - \nu_t$$

$$\nu_t = \gamma \nu_{t-1} + \alpha \nabla_\theta f(\theta - \gamma \nu_{t-1}; x)$$
$$\theta = \theta - \nu_t$$

Averaging Gradient (SAG)

$$\theta = \theta - \frac{\alpha}{n} \sum_{i=1}^{n} y_i^k \quad \text{with} \quad y_i^k = \begin{cases} \nabla_\theta f(\theta : x_i) & \text{if } i = i_k, \\ y_i^{k-1} & \text{otherwise.} \end{cases}$$

Adaptive Step size (Adagrad)

Gradient: $$g_{t,i} = \nabla_\theta f(\theta_{t,i}; x)$$

Update Step: $$\theta_{t+1,i} = \theta_{t,i} - \frac{\alpha}{\sqrt{G_{t,ii} + \varepsilon}} \cdot g_{t,i}$$

Where $G_t \in \mathbb{R}^{d \times d}$ is a diagonal matrix where each element
Is the **sum of the squares** of the gradients up to t.

Model

$$
\begin{aligned}
V_{\text{Car}} &= \text{ASC}_{\text{Car}} + \beta_{\text{TT,Car}}\,\text{TT}_{\text{Car}} \\
&\quad + \beta_{\text{C,Car}}\,\text{C}_{\text{Car}} + \beta_{\text{Senior}}\,1_{\text{Senior}}
\end{aligned}
$$

$$
\begin{aligned}
V_{\text{SM}} &= \text{ASC}_{\text{SM}} + \beta_{\text{TT,SM}}\,\text{TT}_{\text{SM}} \\
&\quad + \beta_{\text{C,SM}}\,\text{C}_{\text{SM}} + \beta_{\text{HE}}\,\text{HE}_{\text{SM}} \\
&\quad + \beta_{\text{Senior}}\,1_{\text{Senior}}
\end{aligned}
$$

$$
\begin{aligned}
V_{\text{Train}} &= \text{ASC}_{\text{Train}} + \beta_{\text{TT,Train}}\,\text{TT}_{\text{Train}} \\
&\quad + \beta_{\text{C,Train}}\,\text{C}_{\text{Train}} + \beta_{\text{HE}}\,\text{HE}_{\text{Train}}
\end{aligned}
$$

Swissmetro (11 parameters)

Model trained with Biogeme

Log-likelihood: **-7145.721**

Name	Value	Std err	t-test	p-value
ASC_{Car}	0	-	-	-
ASC_{SM}	$7.86 \cdot 10^{-1}$	$6.93 \cdot 10^{-2}$	11.35	0.00
$\text{ASC}_{\text{Train}}$	$9.83 \cdot 10^{-1}$	$1.31 \cdot 10^{-1}$	7.48	0.00
$\beta_{\text{TT,Car}}$	$-1.05 \cdot 10^{-2}$	$7.89 \cdot 10^{-4}$	-8.32	0.00
$\beta_{\text{TT,SM}}$	$-1.44 \cdot 10^{-2}$	$6.36 \cdot 10^{-4}$	-21.29	0.00
$\beta_{\text{TT,Train}}$	$-1.80 \cdot 10^{-2}$	$8.65 \cdot 10^{-4}$	-20.78	0.00
$\beta_{\text{C,Car}}$	$-6.56 \cdot 10^{-3}$	$7.89 \cdot 10^{-4}$	-8.32	0.00
$\beta_{\text{C,SM}}$	$-8.00 \cdot 10^{-3}$	$3.76 \cdot 10^{-4}$	-21.29	0.00
$\beta_{\text{C,Train}}$	$-1.46 \cdot 10^{-2}$	$9.65 \cdot 10^{-4}$	-15.09	0.00
β_{Senior}	-1.06	$1.16 \cdot 10^{-1}$	-9.11	0.00
β_{HE}	$-6.88 \cdot 10^{-3}$	$1.03 \cdot 10^{-3}$	-6.69	0.00

Stochastic Newton Method

Algorithm 1 Stochastic Newton Method (SNM)

Input: Starting parameter value (θ_0), data ($\mathcal{D}$), function (f), gradient (∇f), Hessian ($\nabla^2 f$), number of epochs (n_{ep}), batch size (n_{batch})

Output: Epochs (e), parameters (θ), 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, θ and f_v. Set $\theta[0] \leftarrow \theta_0$
    for $i = 0 \dots n_{iter}$ **do**
        $e[i] \leftarrow i \cdot n_{batch} / n_{\mathcal{D}}$ ▷ Store the epoch
        $f_v[i] \leftarrow f(\theta[i])$ ▷ Store the function value
        `idx` $\leftarrow n_{batch}$ values from $\mathcal{U}(0, n_{\mathcal{D}})$ without replacement
        `grad` $\leftarrow \nabla f_{\texttt{idx}}(\theta[i])$ ▷ Gradient on the samples from `idx`
        `hess` $\leftarrow \nabla^2 f_{\texttt{idx}}(\theta[i])$ ▷ Hessian on the samples from `idx`
        if `hess` is non singular **then**
            `inv_hess` $\leftarrow$ `hess`$^{-1}$
            `step` $\leftarrow -$`grad` $\cdot$ `inv_hess`
        else
            `step` $\leftarrow$ `grad`
        $\alpha \leftarrow$ Backtracking Line Search with `step` on the subset of data with indices from `idx`
        $\theta[i+1] \leftarrow \theta[i] + \alpha \cdot$ `step`
    $e[n_{iter}] \leftarrow n_{iter} \cdot n_{batch} / n_{\mathcal{D}}$
    $f_v[n_{iter}] \leftarrow f(\theta[n_{iter}])$
    return e, θ and f_v

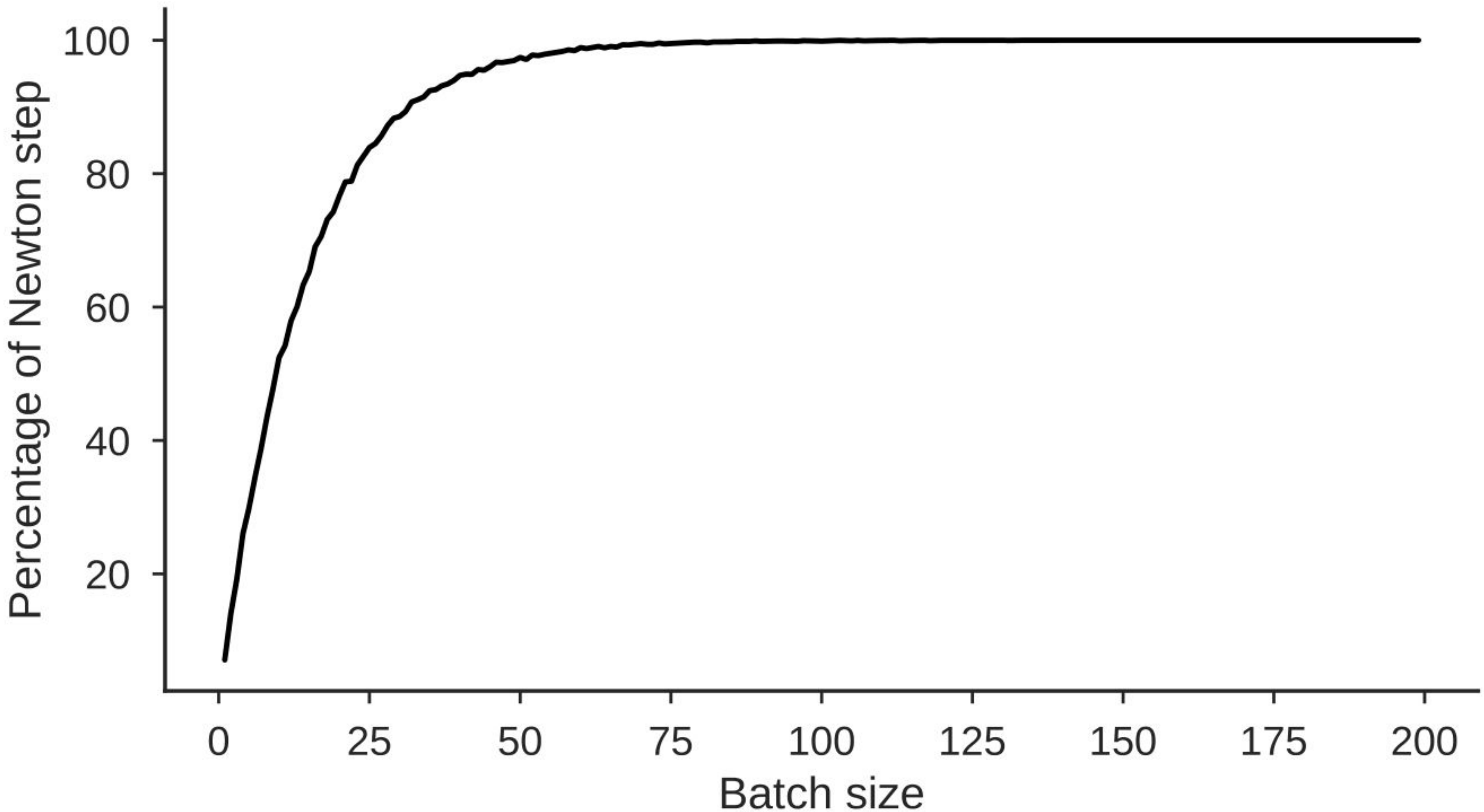
Armijo Backtracking Line Search

- Let $\mathbf{p}$ be the search direction and α_0 the maximum step size
- Define the local slope as $m = \mathbf{p}^T \nabla_\theta f(\theta; x)$
- Choose two parameters: $c \in (0, 1)$ and $\tau \in (0, 1)$

Algorithm:

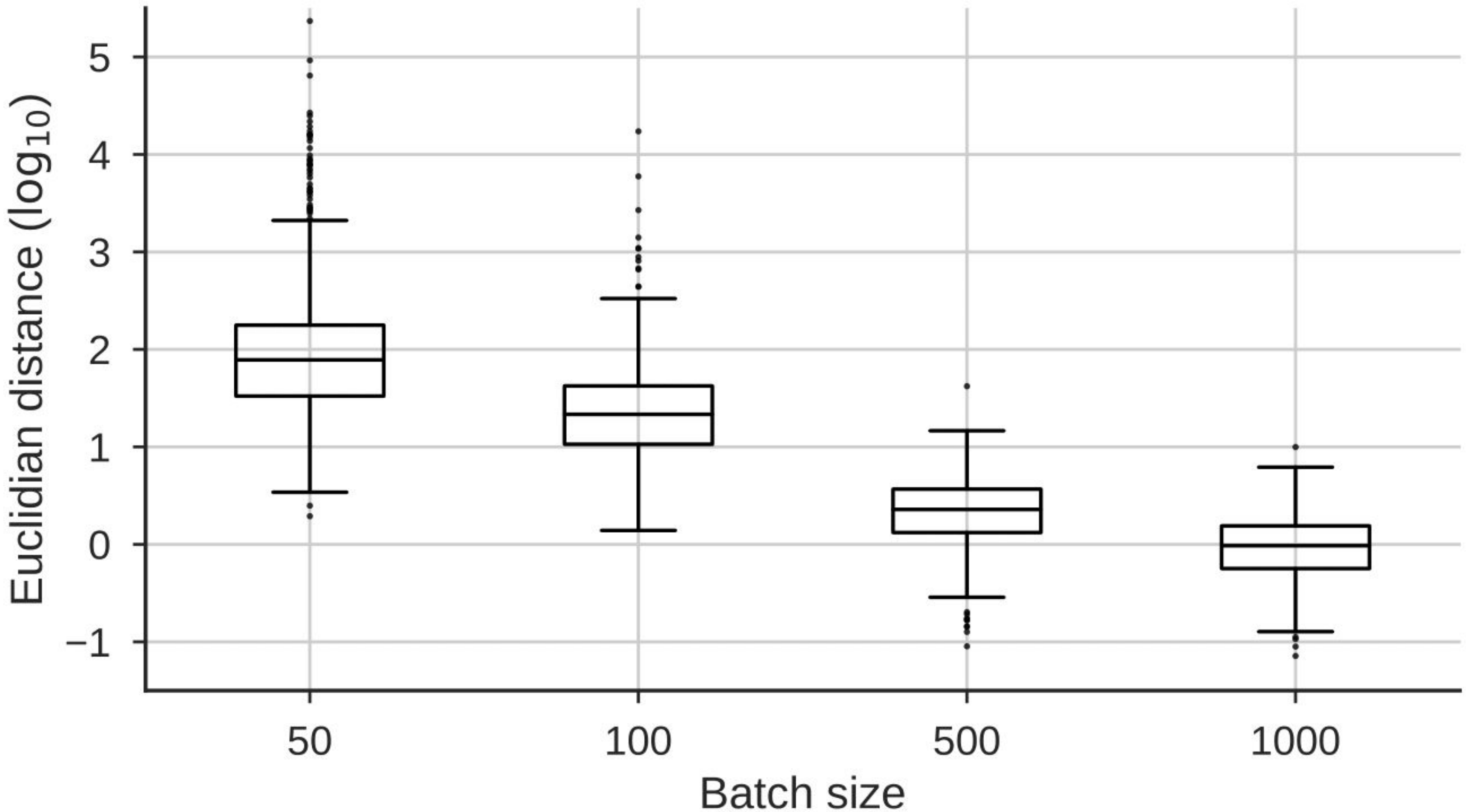
1. Set $t = -cm$
2. While $f(\theta; x) - f(\theta + \alpha_j \mathbf{p}) \leq \alpha_j t$
3. $\quad \alpha_{j+1} = \tau \alpha_j$
4. Return α_j

Not enough Newton steps?



Theoretical percentage of Newton step in function of the batch size

Bad Direction?



Euclidian distance between optimal parameters obtained on the full dataset and optimal parameters found on batches of the data

Future extensions

$$P(X, i|\mathcal{C}) = \prod_{n=1}^{N} P_n(x_n|\beta_n, \xi)$$

$$P_n(x_n|\beta_n, \xi) = \sum_{r=1}^{R} P_n(x_n|\beta_n, \xi_r)$$

$$P_n(x_n|\beta_n, \xi_r) = \prod_{t=1}^{T} P_n(x_n|\beta_n, \xi_r, t)$$