Optimization of Discrete Choice Models

Gael Lederrey, Virginie Lurkin and Michel Bierlaire

Work in progress

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

TRANSP-OR

WORKSHOP ON DISCRETE CHOICE MODELS 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

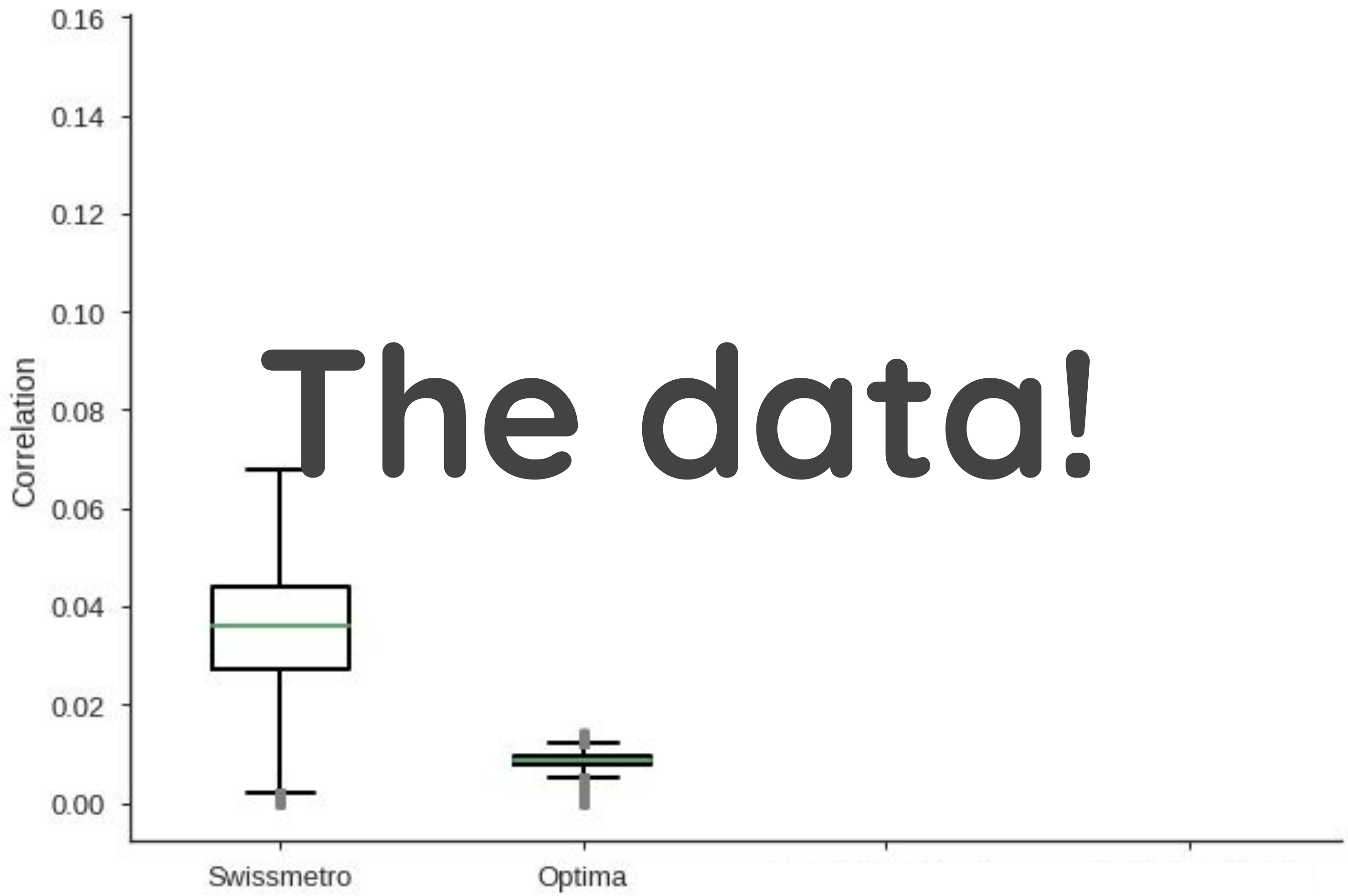
Where to get inspiration?

- 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!

One fundamental difference



The data!

Optimization of ML - The Basics

GD

(Cauchy, 1847)

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

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

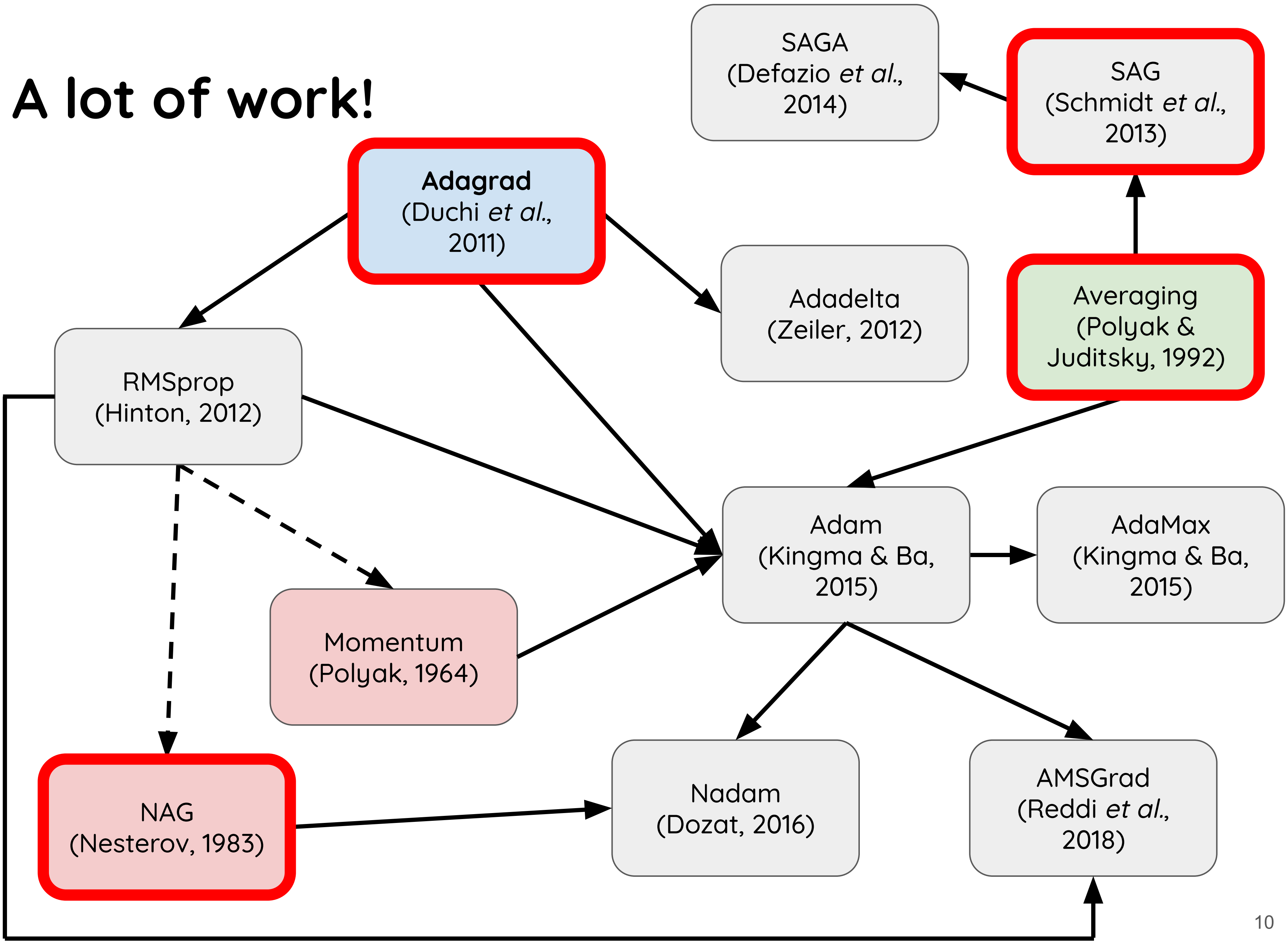
f : Function, $f \in C^1(\mathbb{R}^\mathrm{n})$

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

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

First-order methods

A lot of work!


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

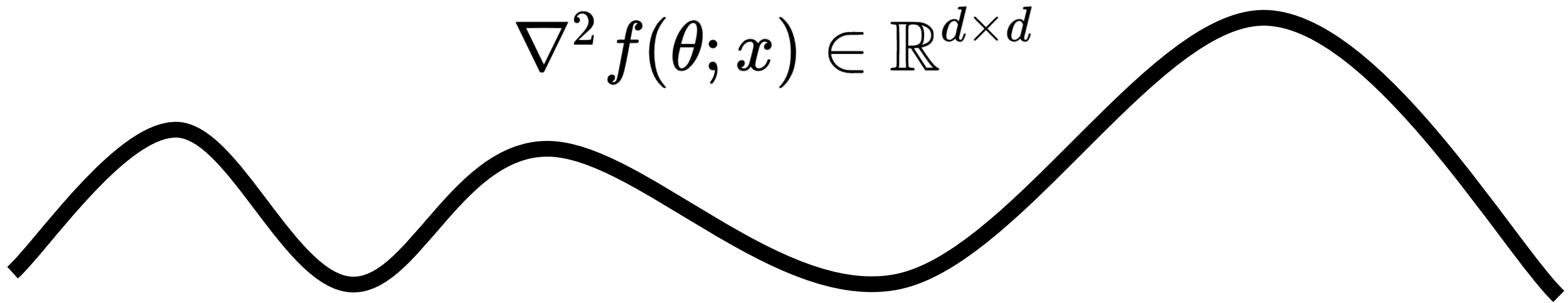
First-Order vs Second-Order

- 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?

Newton Method

- Update step:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha[\nabla^2 f(\mathbf{x}_n)]^{-1} \nabla f(\mathbf{x}_n)$$

- Use Conjugate Gradient to find the step direction

$$\nabla^2 f(\mathbf{x}_n) \Delta \mathbf{x} = -\nabla f(\mathbf{x}_n)$$

- Use Line Search for proper step size
  - Wolfe 1, Wolfe 2, Armijo, etc.

Trust Region

- Define a trust region around current point: $\mathbf{x}_n$
- Use Taylor to approximate $f(\mathbf{x}_n)$

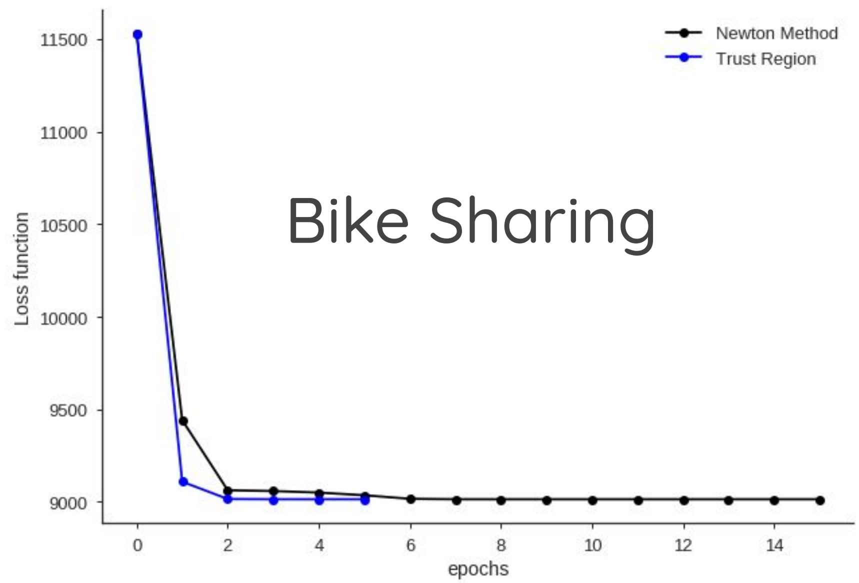
$$f(\mathbf{x}_n + \Delta\mathbf{x}) = f(\mathbf{x}_n) + \nabla f(\mathbf{x}_n)\Delta\mathbf{x} + \frac{1}{2}\nabla^2 f(\mathbf{x}_n)(\Delta\mathbf{x})^2$$

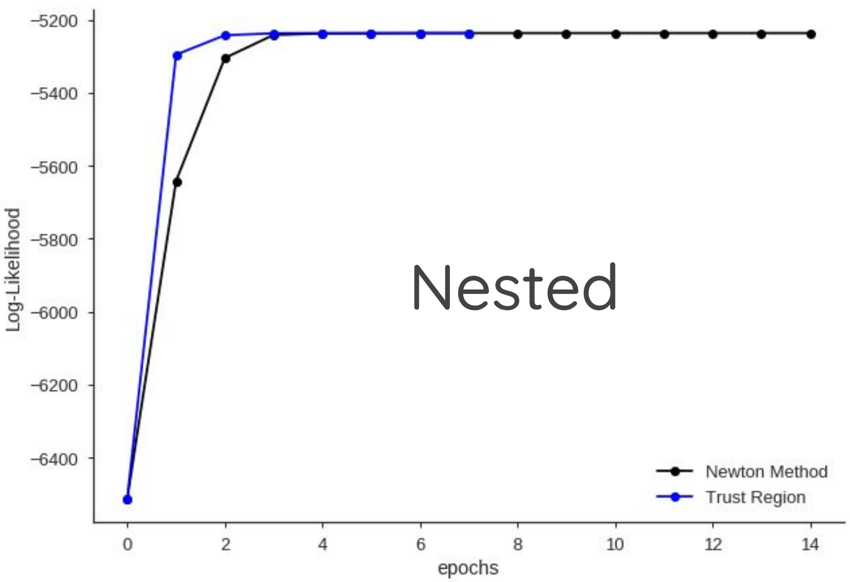
- Find $\Delta\mathbf{x}$ that minimize the Taylor approx.
- Based on $\Delta\mathbf{x}$, decide to:
  - Augment the size of the TR
  - Reduce the size of the TR
  - Do a step or try again
  - …

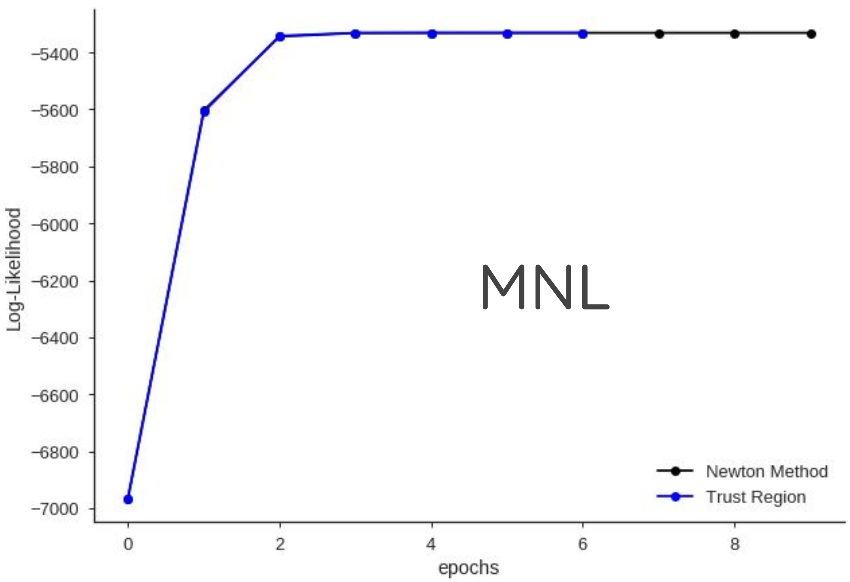
Models and datasets

- Swissmetro (~10k) and MNL
- Swissmetro (~10k) and Nested Logit Model
- Bike Sharing (~16k) and Logistic Regression

Newton Method and Trust Region





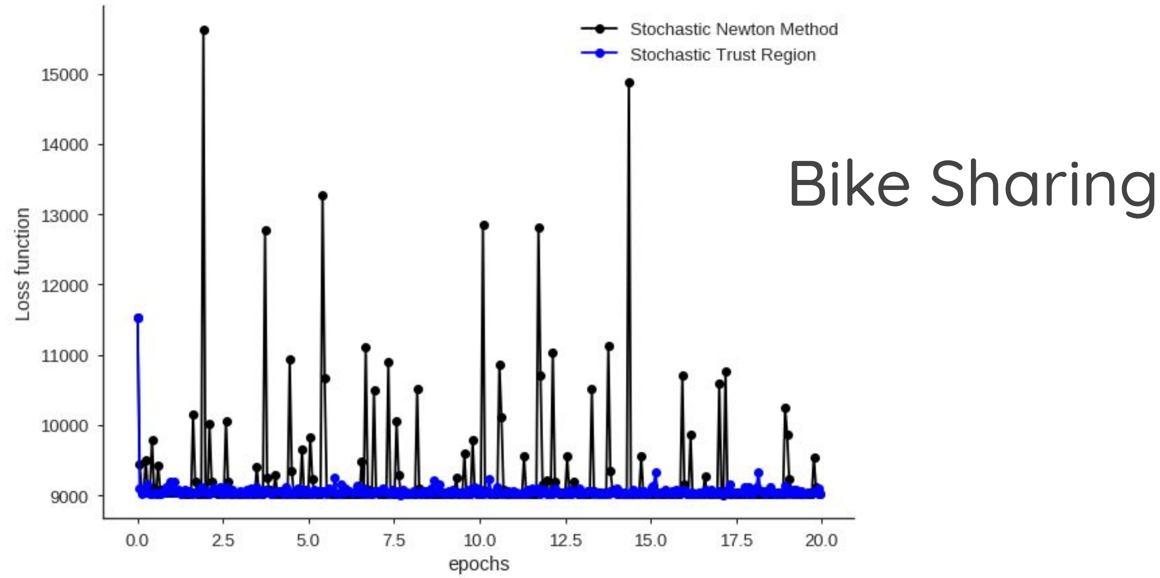


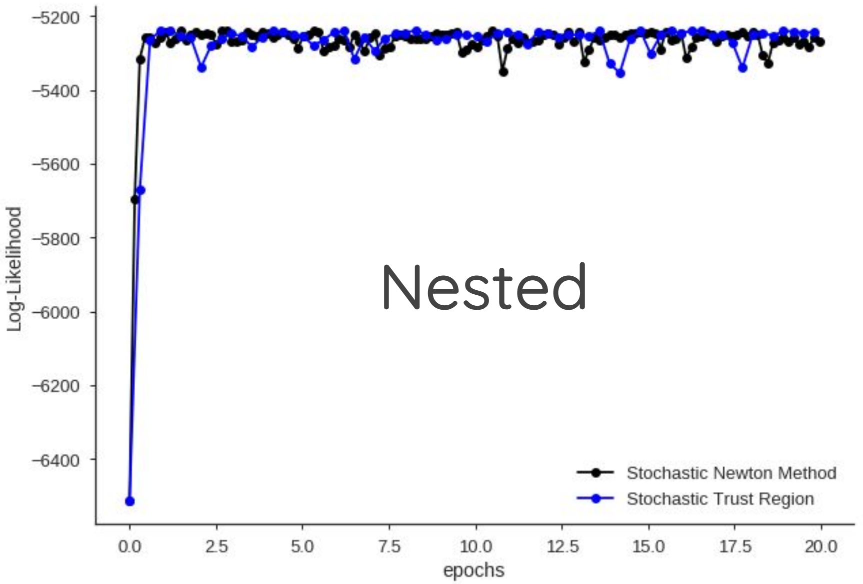
Introducing Stochasticity

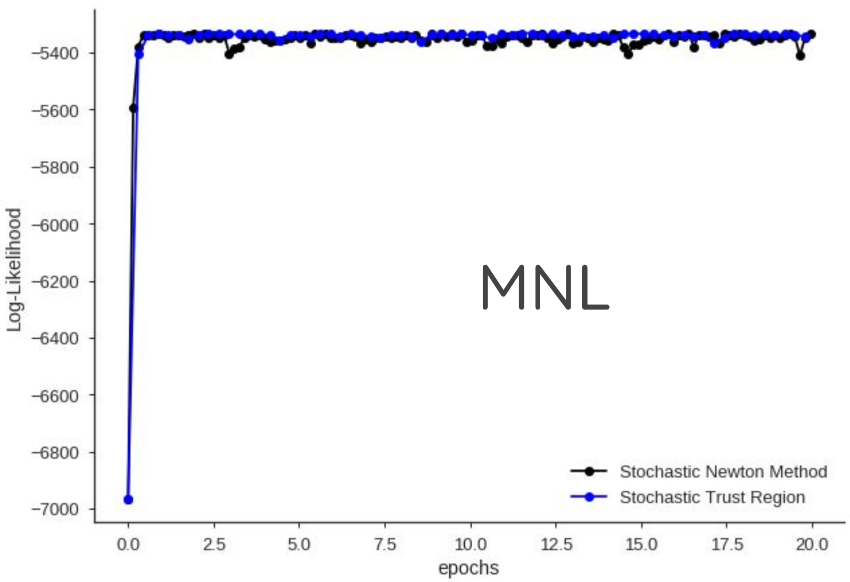
- No “Full” gradient or Hessian
- Choose a “final” batch size
- Sample the data without replacement

Is Stochasticity a good thing?

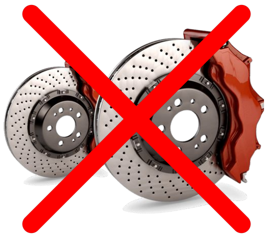
Batch size: 1000



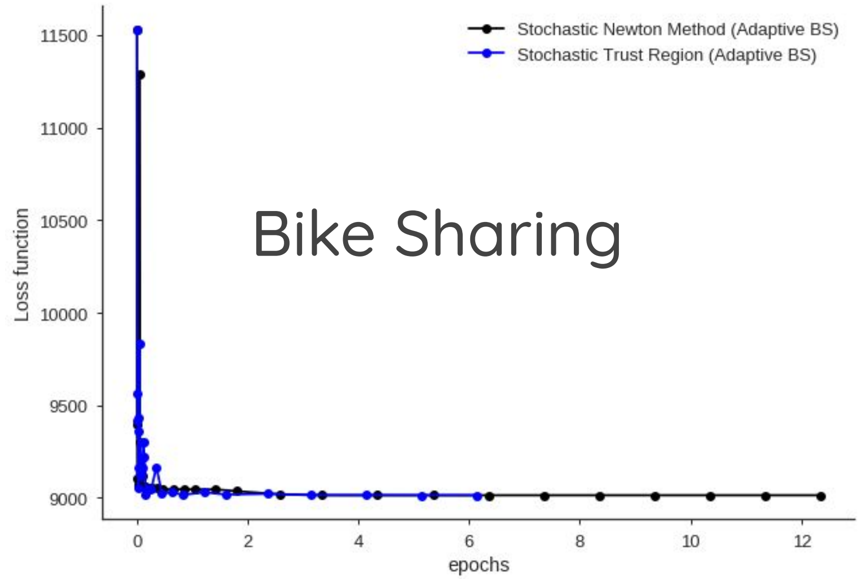


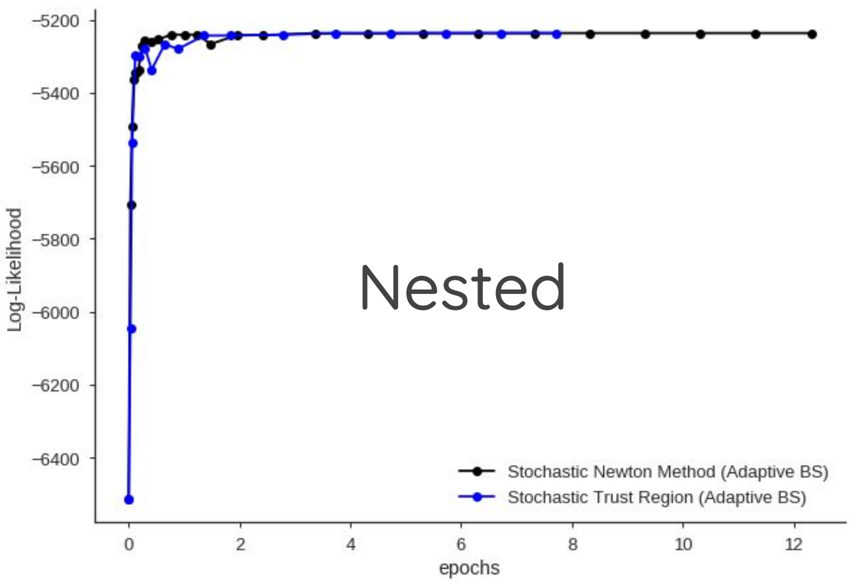


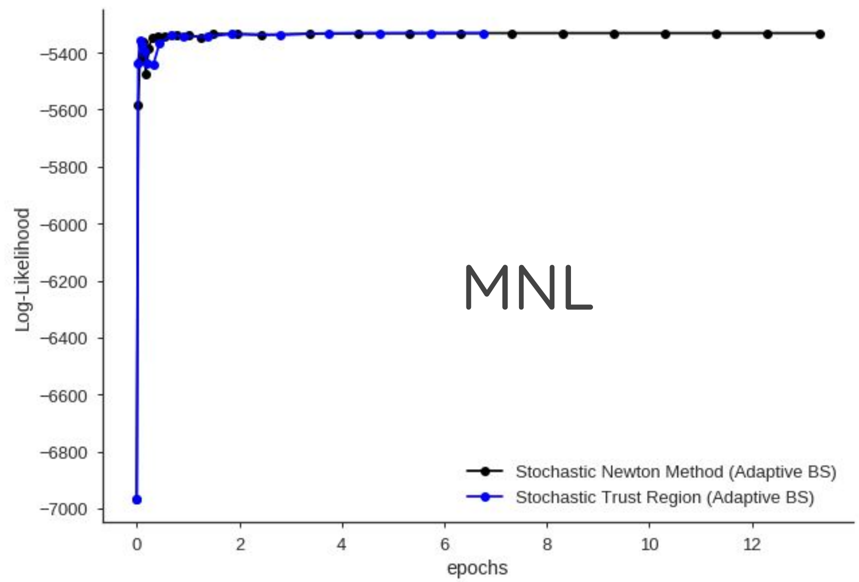
Adaptive Batch Size (ABS)



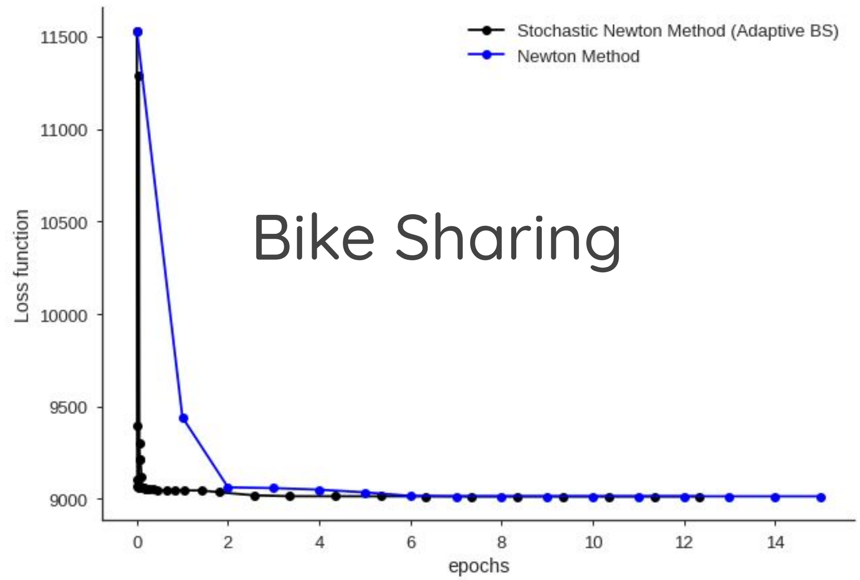
ABS at work!


Bike Sharing
Stochastic Newton Method (Adaptive BS)
Stochastic Trust Region (Adaptive BS)
Loss function
11500
11000
10500
10000
9500
9000
0
2
4
6
8
10
12
epochs

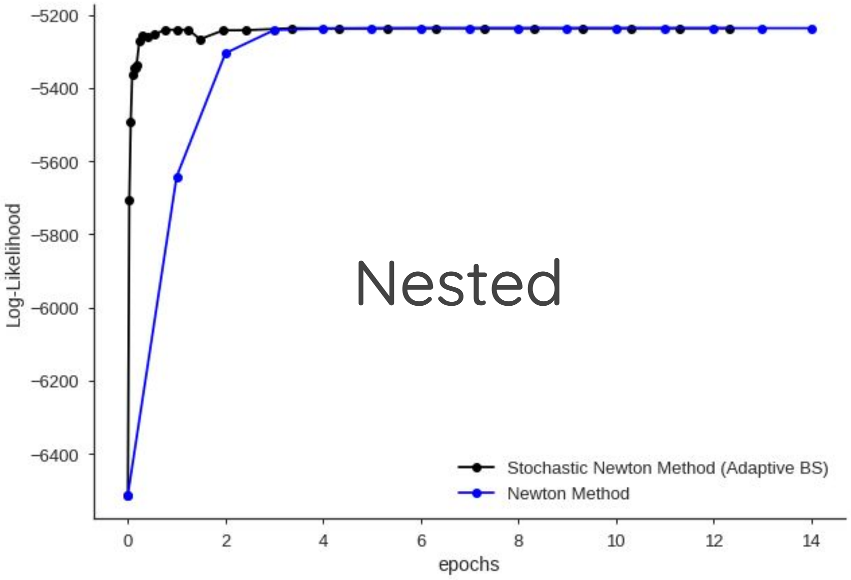

Nested
Stochastic Newton Method (Adaptive BS)
Stochastic Trust Region (Adaptive BS)
Log-Likelihood
−5200
−5400
−5600
−5800
−6000
−6200
−6400
0
2
4
6
8
10
12
epochs


MNL
Stochastic Newton Method (Adaptive BS)
Stochastic Trust Region (Adaptive BS)
Log-Likelihood
−5400
−5600
−5800
−6000
−6200
−6400
−6600
−6800
−7000
0
2
4
6
8
10
12
epochs

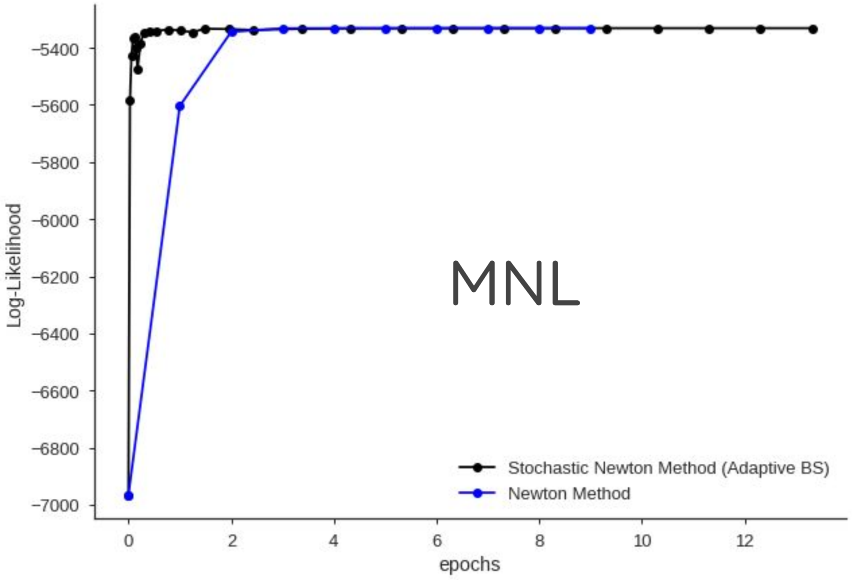
Comparison



Stochastic Newton Method (Adaptive BS)
Newton Method
Bike Sharing
Loss function
11500
11000
10500
10000
9500
9000
0
2
4
6
8
10
12
14
epochs



Nested
Log-Likelihood
−5200
−5400
−5600
−5800
−6000
−6200
−6400
Stochastic Newton Method (Adaptive BS)
Newton Method
0
2
4
6
8
10
12
14
epochs



MNL
Log-Likelihood
−5400
−5600
−5800
−6000
−6200
−6400
−6600
−6800
−7000
Stochastic Newton Method (Adaptive BS)
Newton Method
0
2
4
6
8
10
12
epochs

Wrap-up

		Bike Sharing	MNL	Nested
Newton Method	Basic			
	Stocha			
	ABS			

		Bike Sharing	MNL	Nested
Trust Region	Basic			
	Stocha			
	ABS			

Conclusion

- Basic stochastic algorithms seem to struggle
- A lot of work is needed on the batch size
- **But,** Promising early results!

Future work

- Testing ABS on
  - More models: Cross Nested, Mixed Logit, etc.
  - More data: RP & SP data from 2015 microsensus

- Continue developing ideas about
  - Different batch size update
  - Quasi Newton methods?
- Candidacy!

Thank you!