Optimization of DCMs using first-order methods

Gael Lederrey
PhD student @ TRANSP-OR, EPFL

05.09.2018
Outline

1. Motivation
2. State-of-the-art
3. Optimization of DCMs
4. Beyond Optimization
5. Conclusion
Motivation
DCMs in a nutshell

- Well established theory with many success stories!
Why is a solution required?

Every minute of the Day (2017)

- Twitter users sent 456,000 tweets
- Youtube users uploaded 300 hours of video
- Instagram users shared 46,740 pictures
- Google received over 3,600,000 search queries

Why is a solution required?

Twitter users sent 100,000 tweets

Google received over 2,000,000 search queries

BIG DATA

Ref: Domo + Internet
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
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
- \(\theta\): Parameters
- \(\alpha\): 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
- \(\theta\): Parameters
- \(\alpha\): 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
- \(\theta\): Parameters
- \(\alpha\): Step size
- \(f\): Function, \(f \in C^1(\mathbb{R}^n)\)
- \(x\): 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)
First-Order optimization

Adagrad (Duchi et al., 2011)

SAGA (Defazio et al., 2014)

SAG (Schmidt et al., 2013)

Adadelta (Zeiler, 2012)

Averaging (Polyak & Juditsky, 1992)

RMSprop (Tieleman & Hinton, 2012)

Momentum (Polyak, 1964)

Adam (Kingma & Ba, 2014)

AdaMax (Kingma & Ba, 2014)

NAD (Nesterov, 1983)

Nadam (Dozat, 2016)

AMSGGrad (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.
  
  Gower et al. (2018), Martens (2010), Bordes et al. (2009-2010)
Optimization of Discrete Choice Models
Motivation

- Data is growing everyday!

- DCMs (will) have to deal with these new datasets!

- => Make them faster, especially with big datasets.
First-order methods

Fail!

Lederrey et al., 2018 - hEART 2018
Second-order methods
Stochastic Newton Method

Not good enough!
Problems?

- Seems to be stuck around the optimum. => “Flat area”?
Adaptive Batch Size (ABS)

- Not enough improvement => Update the BS!
Results

Better: 10/12

(a) MNL-SM

(b) NLM-SM

(c) LogReg-BS

(d) LogReg-BS-Full
Next steps

- Test the ABS technique on bigger models/datasets
  => Danalet and Mathys (2018)

  Suggestions are welcome!

- Final step: Write an article!
Beyond Optimization
What’s left?

1. Faster optimization of DCMs.

2. Better at predicting. (⚠️ Interpretability)

3. Use data to find “better” models. (faster!)
<table>
<thead>
<tr>
<th><strong>DCMs</strong></th>
<th>v.s.</th>
<th><strong>ML</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Model-driven</td>
<td>Main goal: Understand behavior</td>
<td>Data-driven</td>
</tr>
<tr>
<td>Can also predict</td>
<td>Main goal: Prediction</td>
<td>Can also help to understand behavior</td>
</tr>
<tr>
<td>Likelihood as objective function</td>
<td>Likelihood (possible) as objective function</td>
<td></td>
</tr>
<tr>
<td>Optimization is very important!</td>
<td>Optimization is very important!</td>
<td></td>
</tr>
</tbody>
</table>
What we should not do!
What should we do then?

- Improve DCMs with ML!
- Examples:
Conclusion
Conclusion

- DCMs have to be improved!
- Better optimization process are already working!
- ML can bring a lot to the DCMs field...
- The same goes in the other direction.
How can we help each other?

Do you have trouble optimizing huge and complicated DCMs with big datasets?

Contact me:

gael.lederrey@epfl.ch
Thank you!
Back up Slides
# ABS Algorithm

**Algorithm 1 Adaptive Batch Size (ABS)**

**Input:** Current iteration index ($M$), function value at iteration $M$ ($f_M$), and batch size ($n$)

**Output:** New batch size ($n'$)

1. **function** ABS
2. Store $f_M$ in a list $\mathcal{F}$
3. Compute $WMA_{M,W}$ using $\mathcal{F}$ and store it in a list $\mathcal{A}$
4. if $M > 0$ then \(\triangleright\) We need at least two values to compute the improvement.
5. Compute $i$ the improvement as in Equation 3 using the list $\mathcal{A}$ and store it in a list $\mathcal{I}$
6. if $n < N$ then
7. \(\triangleright\) Improvement under the threshold
8. if $\mathcal{I}_M < \Delta$ then
9. \(\triangleright\) We restart the counter
10. $c = c + 1$
11. else
12. $c = 0$
13. \(\triangleright\) We will update the batch size
14. if $c == C$ then
15. \(\triangleright\) We restart the counter
16. $c = 0$
17. $n' = \tau \cdot n$
18. \(\triangleright\) The batch size is too big now
19. if $n' >= N$ then
20. \(\triangleright\) We will update the batch size
21. $n' = N$
22. else
23. return $n' n' = n$