Backpropagation and Nonsmooth Optimization for Machine Learning

Andrea Walther
Institut für Mathematik
Humboldt-Universität zu Berlin

CWI Workshop – Scientific Machine Learning

Amsterdam, December 7, 2023

Outline

Retail part: Joint work with Aswin Kannan and Timo Kreimeier, Humboldt-Universität zu Berlin

Where are Derivatives Needed?

- Optimization:

$$
\begin{array}{lll}
\text{unbounded:} & \min f(x), & f:\mathbb{R}^n \to \mathbb{R} \\
\text{bounded:} & \min f(x), & f:\mathbb{R}^n \to \mathbb{R} \\
 & c(x) = 0, & c:\mathbb{R}^n \to \mathbb{R}^m \\
 & h(x) \le 0, & h:\mathbb{R}^n \to \mathbb{R}^l
\end{array}
$$

Where are Derivatives Needed?

- Optimization:

$$
\begin{array}{llll}
\text{unbounded:} & \min f(x), & & f:\mathbb{R}^n \to \mathbb{R} \\
\text{bounded:} & \min f(x), & & f:\mathbb{R}^n \to \mathbb{R} \\
 & c(x) = 0, & & c:\mathbb{R}^n \to \mathbb{R}^m \\
 & h(x) \leq 0, & & h:\mathbb{R}^n \to \mathbb{R}^l
\end{array}
$$

- Solution of nonlinear equation systems

$$F(x) = 0, \quad F : \mathbb{R}^n \to \mathbb{R}^n$$

Newton method requires $F'(x) \in \mathbb{R}^{n\times n}$

Where are Derivatives Needed?

- Optimization:

$$\begin{array}{lll} \text{unbounded:} & \min f(x), & f:\mathbb{R}^n \to \mathbb{R} \\ \text{bounded:} & \min f(x), & f:\mathbb{R}^n \to \mathbb{R} \\ & c(x) \;=\; 0, & c:\mathbb{R}^n \to \mathbb{R}^m \\ & h(x) \leq 0, & h:\mathbb{R}^n \to \mathbb{R}^l \end{array}$$

- Solution of nonlinear equation systems

$$F(x) \;=\; 0, \quad F \;:\; \mathbb{R}^n \;\to\; \mathbb{R}^n$$

  Newton method requires $F'(x) \in \mathbb{R}^{n \times n}$

- Simulation of complex system
  - definition
  - integration of differential equations using implicit methods

Where are Derivatives Needed?

- Optimization:

$$\begin{array}{lll} \text{unbounded:} & \min f(x), & f:\mathbb{R}^n \to \mathbb{R} \\ \text{bounded:} & \min f(x), & f:\mathbb{R}^n \to \mathbb{R} \\ & c(x) \;=\; 0, & c:\mathbb{R}^n \to \mathbb{R}^m \\ & h(x) \leq 0, & h:\mathbb{R}^n \to \mathbb{R}^l \end{array}$$

- Solution of nonlinear equation systems

  $$F(x) = 0, \quad F : \mathbb{R}^n \to \mathbb{R}^n$$

  Newton method requires $F'(x) \in \mathbb{R}^{n\times n}$

- Simulation of complex system
  - definition
  - integration of differential equations using implicit methods
- Sensitivity analysis
- Real-time control

Where are Derivatives Needed?

- Optimization:

$$\begin{array}{lll} \text{unbounded:} & \min f(x), & f:\mathbb{R}^n \to \mathbb{R} \\ \text{bounded:} & \min f(x), & f:\mathbb{R}^n \to \mathbb{R} \\ & c(x) = 0, & c:\mathbb{R}^n \to \mathbb{R}^m \\ & h(x) \le 0, & h:\mathbb{R}^n \to \mathbb{R}^l \end{array}$$

- Solution of nonlinear equation systems

$$F(x) = 0, \quad F : \mathbb{R}^n \to \mathbb{R}^n$$

  Newton method requires $F'(x) \in \mathbb{R}^{n \times n}$

- Simulation of complex system
  - definition
  - integration of differential equations using implicit methods
- Sensitivity analysis
- Real-time control
- ML, e.g., Stochastic Gradient Descent, Adam, ...
  target functions quite often nonsmooth!

Computing Derivatives

Given:

Description of functional relation as

- formula $y = F(x)$ $\Rightarrow$ explicit expression $y' = F'(x)$
- computer program $\Rightarrow$?

Computing Derivatives

Given:

Description of functional relation as

- formula $y = F(x)$ $\Rightarrow$ explicit expression $y' = F'(x)$
- computer program $\Rightarrow$?

Task:

Computation of derivatives taking

- requirements on exactness
- computational effort

into account

Algorithmic Differentiation (AD)

aka Automatic Differentiation

$=$ Differentiation of computer programs implementing $F : \mathbb{R}^n \mapsto \mathbb{R}^m$

Algorithmic Differentiation (AD)

aka Automatic Differentiation

= Differentiation of computer programs implementing $F : \mathbb{R}^n \mapsto \mathbb{R}^m$

Main Products:

- Quantitative dependence information (local):
  - Weighted and directed partial derivatives
  - Error and condition number estimates ...
  - Lipschitz constants, interval enclosures ...

Algorithmic Differentiation (AD)

aka Automatic Differentiation

$=$ Differentiation of computer programs implementing $F : \mathbb{R}^n \mapsto \mathbb{R}^m$

Main Products:

- Quantitative dependence information (local):
  - Weighted and directed partial derivatives
  - Error and condition number estimates ...
  - Lipschitz constants, interval enclosures ...
- Qualitative dependence information (regional):
  - Sparsity structures, degrees of polynomials
  - Ranks, eigenvalue multiplicities ...

Algorithmic Differentiation (AD)

aka Automatic Differentiation

$=$ Differentiation of computer programs implementing $F : \mathbb{R}^n \mapsto \mathbb{R}^m$

Main Products:

- Quantitative dependence information (local):
  - Weighted and directed partial derivatives
  - Error and condition number estimates ...
  - Lipschitz constants, interval enclosures ...
- Qualitative dependence information (regional):
  - Sparsity structures, degrees of polynomials
  - Ranks, eigenvalue multiplicities ...

Assumption:
F differentiable at least in a neighbourhood of current argument x

Historical Development of AD

J. Nolan	1953	$\rightarrow$	J. M. Thames et al.	1975	$\rightarrow$
L. M. Beda et al.	1959	$\rightarrow$	D. D. Warner	1975	$\rightarrow$
A. Gibbons	1960	$\rightarrow$			
J. W. Hanson et al.	1962	$\rightarrow$	J. Joss	1980	$\rightarrow$
R. E. Wengert	1964	$\rightarrow$			
R. D. Wilkins	1964	$\rightarrow$			
G. Wanner	1965	$\rightarrow$	L. B. Rall	1980	$\rightarrow$
R. Bellman et al.	1965	$\rightarrow$			
Y. F. Chang	1967	$\rightarrow$	R. Kalaba et al.	1983	$\rightarrow$
D. Barton et al.	1971	$\rightarrow$			
R. E. Pugh	1972	$\rightarrow$			
			L. C. W. Dixon et al.	1986	$\rightarrow$
			...		

Historical Development of AD

J. Nolan	1953	$\rightarrow$	J. M. Thames et al.	1975	$\rightarrow$
L. M. Beda et al.	1959	$\rightarrow$	D. D. Warner	1975	$\rightarrow$
A. Gibbons	1960	$\rightarrow$	W. Miller	1975	$\leftarrow$
J. W. Hanson et al.	1962	$\rightarrow$	J. Joss	1980	$\rightarrow$
R. E. Wengert	1964	$\rightarrow$	G. Kedem	1980	$\leftarrow$
R. D. Wilkins	1964	$\rightarrow$	B. Speelpenning	1980	$\leftarrow$
G. Wanner	1965	$\rightarrow$	L. B. Rall	1980	$\rightarrow$
R. Bellman et al.	1965	$\rightarrow$	W. Baur, V. Strassen	1983	$\leftarrow$
Y. F. Chang	1967	$\rightarrow$	R. Kalaba et al.	1983	$\rightarrow$
S. Linnainma	1970	$\leftarrow$	M. Iri et al.	1984	$\leftarrow$
D. Barton et al.	1971	$\rightarrow$	K. W. Kim et al.	1984	$\leftarrow$
G. M. Ostrowski	1971	$\leftarrow$	J. W. Sawyer	1984	$\leftarrow$
R. E. Pugh	1972	$\rightarrow$			
W. Stacey	1973	$\leftarrow$	L. C. W. Dixon et al.	1986	$\rightarrow$
P. Werbos	1974	$\leftarrow$	...		

Historical Development of AD

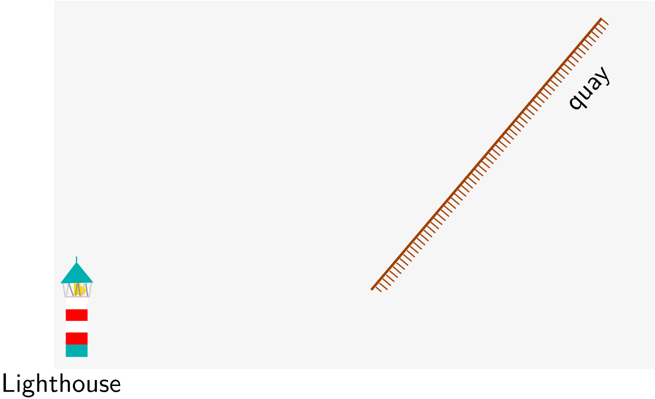
J. Nolan	1953	$\rightarrow$	J. M. Thames et al.	1975	$\rightarrow$
L. M. Beda et al.	1959	$\rightarrow$	D. D. Warner	1975	$\rightarrow$
A. Gibbons	1960	$\rightarrow$	W. Miller	1975	$\leftarrow$
J. W. Hanson et al.	1962	$\rightarrow$	J. Joss	1980	$\rightarrow$
R. E. Wengert	1964	$\rightarrow$	G. Kedem	1980	$\leftarrow$
R. D. Wilkins	1964	$\rightarrow$	B. Speelpenning	1980	$\leftarrow$
G. Wanner	1965	$\rightarrow$	L. B. Rall	1980	$\rightarrow$
R. Bellman et al.	1965	$\rightarrow$	W. Baur, V. Strassen	1983	$\leftarrow$
Y. F. Chang	1967	$\rightarrow$	R. Kalaba et al.	1983	$\rightarrow$
S. Linnainma	1970	$\leftarrow$	M. Iri et al.	1984	$\leftarrow$
D. Barton et al.	1971	$\rightarrow$	K. W. Kim et al.	1984	$\leftarrow$
G. M. Ostrowski	1971	$\leftarrow$	J. W. Sawyer	1984	$\leftarrow$
R. E. Pugh	1972	$\rightarrow$	E. M. Oblow et al.	1985	$\leftrightarrow$
W. Stacey	1973	$\leftarrow$	L. C. W. Dixon et al.	1986	$\rightarrow$
P. Werbos	1974	$\leftarrow$	...		

Historical Development of AD

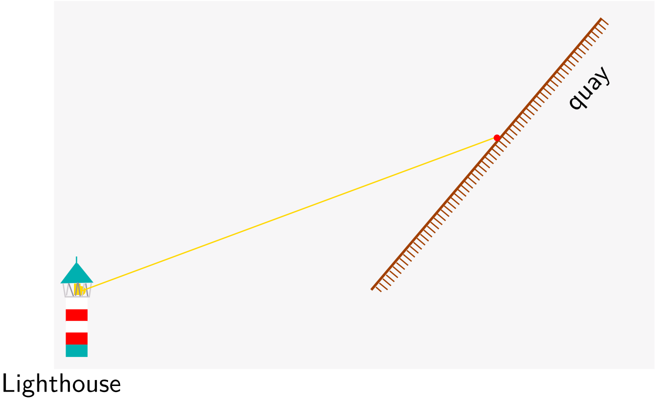
J. Nolan	1953	$\rightarrow$	J. M. Thames et al.	1975	$\rightarrow$
L. M. Beda et al.	1959	$\rightarrow$	D. D. Warner	1975	$\rightarrow$
A. Gibbons	1960	$\rightarrow$	W. Miller	1975	$\leftarrow$
J. W. Hanson et al.	1962	$\rightarrow$	J. Joss	1980	$\rightarrow$
R. E. Wengert	1964	$\rightarrow$	G. Kedem	1980	$\leftarrow$
R. D. Wilkins	1964	$\rightarrow$	B. Speelpenning	1980	$\leftarrow$
G. Wanner	1965	$\rightarrow$	L. B. Rall	1980	$\rightarrow$
R. Bellman et al.	1965	$\rightarrow$	W. Baur, V. Strassen	1983	$\leftarrow$
Y. F. Chang	1967	$\rightarrow$	R. Kalaba et al.	1983	$\rightarrow$
S. Linnainma	1970	$\leftarrow$	M. Iri et al.	1984	$\leftarrow$
D. Barton et al.	1971	$\rightarrow$	K. W. Kim et al.	1984	$\leftarrow$
G. M. Ostrowski	1971	$\leftarrow$	J. W. Sawyer	1984	$\leftarrow$
R. E. Pugh	1972	$\rightarrow$	E. M. Oblow et al.	1985	$\leftrightarrow$
W. Stacey	1973	$\leftarrow$	L. C. W. Dixon et al.	1986	$\rightarrow$
P. Werbos	**1974**	$\leftarrow$	...		

Rumelhart at al. (1986) made backpropagation famous for neural nets

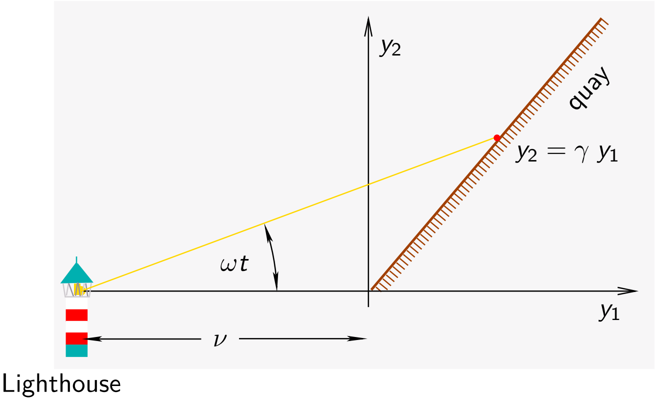
The "Hello-World"-Example of AD



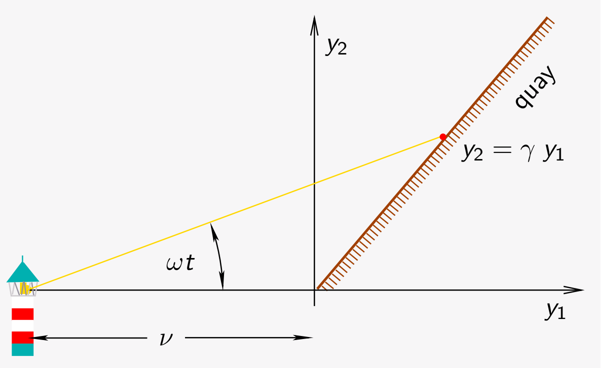
The "Hello-World"-Example of AD



The “Hello-World”-Example of AD



The "Hello-World"-Example of AD



Lighthouse

$$y_1 = \frac{\nu \tan(\omega\, t)}{\gamma - \tan(\omega\, t)} \qquad \text{and} \qquad y_2 = \frac{\gamma\, \nu \tan(\omega\, t)}{\gamma - \tan(\omega\, t)}$$

Evaluation Procedure (Lighthouse)

$$y_1 = \frac{\nu \tan(\omega t)}{\gamma - \tan(\omega t)}$$

$$y_2 = \frac{\gamma \nu \tan(\omega t)}{\gamma - \tan(\omega t)}$$

$$\Longrightarrow$$

$$\begin{array}{lll}
v_{-3} & = x_1 = \nu & \\
v_{-2} & = x_2 = \gamma & \\
v_{-1} & = x_3 = \omega & \\
v_0 & = x_4 = t & \\
\hline
v_1 & = v_{-1} * v_0 & \equiv \varphi_1(v_{-1}, v_0) \\
v_2 & = \tan(v_1) & \equiv \varphi_2(v_1) \\
v_3 & = v_{-2} - v_2 & \equiv \varphi_3(v_{-2}, v_2) \\
v_4 & = v_{-3} * v_2 & \equiv \varphi_4(v_{-3}, v_2) \\
v_5 & = v_4 / v_3 & \equiv \varphi_5(v_4, v_3) \\
v_6 & = v_5 * v_{-2} & \equiv \varphi_6(v_5, v_{-2}) \\
\hline
y_1 & = v_5 & \\
y_2 & = v_6 &
\end{array}$$

Function Evaluation in ML

Typical function evaluation (deep neural net):

Propagation of one data point:

$$
\begin{aligned}
x = x^{(1)} &\rightarrow \tilde{x}^{(1)} = W^{(1)}x^{(1)} + b^{(1)} \quad \rightarrow x^{(2)} = \rho(\tilde{x}^{(1)}) \\
&\rightarrow \tilde{x}^{(2)} = W^{(2)}x^{(2)} + b^{(2)} \quad \rightarrow x^{(3)} = \rho(\tilde{x}^{(2)}) \\
&\rightarrow \cdots \\
&\rightarrow y = W^{(k)}x^{(k)} + b^{(k)}
\end{aligned}
$$

Function Evaluation in ML

Typical function evaluation (deep neural net):

Propagation of one data point:

$$
\begin{aligned}
x = x^{(1)} &\to \tilde{x}^{(1)} = W^{(1)}x^{(1)} + b^{(1)} &&\to x^{(2)} = \rho(\tilde{x}^{(1)}) \\
&\to \tilde{x}^{(2)} = W^{(2)}x^{(2)} + b^{(2)} &&\to x^{(3)} = \rho(\tilde{x}^{(2)}) \\
&\to \cdots \\
&\to y = W^{(k)}x^{(k)} + b^{(k)}
\end{aligned}
$$

Empirical risk, loss function, ...

$$
f(x_{1\leq i\leq M}) = \tfrac{1}{M}\sum_{i=1}^{M} l(y_i(x_i), y_i^{NN})
$$

Function Evaluation in ML

Typical function evaluation (deep neural net):

Propagation of one data point:

$$\begin{aligned} x = x^{(1)} &\to \tilde{x}^{(1)} = W^{(1)}x^{(1)} + b^{(1)} && \to x^{(2)} = \rho(\tilde{x}^{(1)}) \\ &\to \tilde{x}^{(2)} = W^{(2)}x^{(2)} + b^{(2)} && \to x^{(3)} = \rho(\tilde{x}^{(2)}) \\ &\to \cdots \\ &\to y = W^{(k)}x^{(k)} + b^{(k)} \end{aligned}$$

Empirical risk, loss function, ...

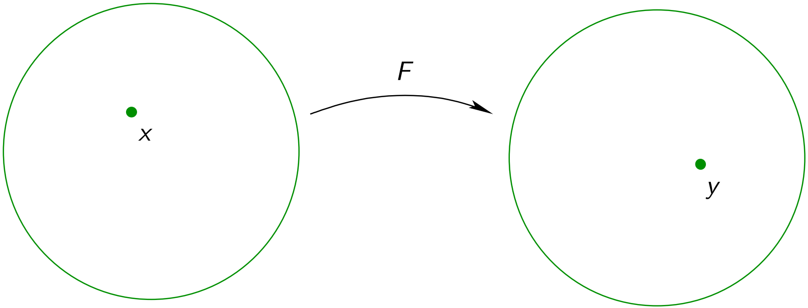
$$f(x_{1\leq i\leq M}) = \frac{1}{M}\sum_{i=1}^{M} l(y_i(x_i), y_i^{NN})$$

Stochastic gradient descent requires

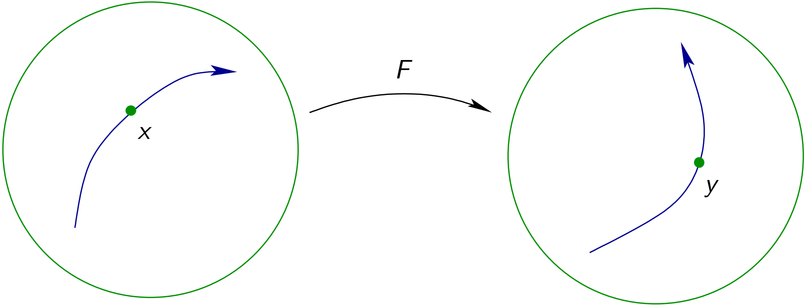
$$\nabla_{W^1,b^1,\ldots,W^k,b^k} l(y_i(x_i), y_i^{NN})$$

for one $i \in \{1, \ldots, M\}$

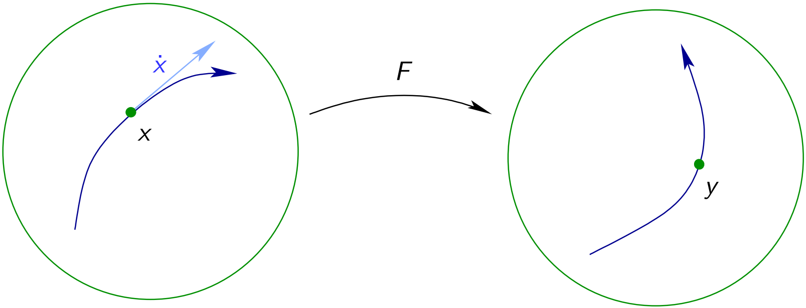
Forward mode AD = Tangents/Sensitivities



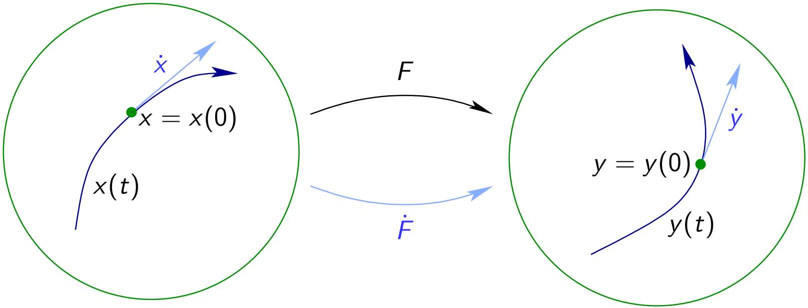
Forward mode AD = Tangents/Sensitivities



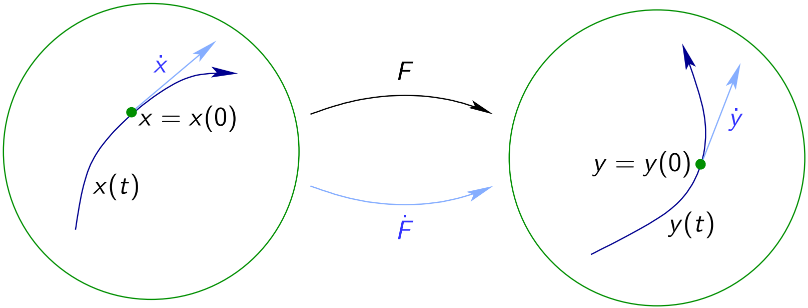
Forward mode AD = Tangents/Sensitivities



Forward mode AD = Tangents/Sensitivities



Forward mode AD = Tangents/Sensitivities



$$\dot{y}(t) \;=\; \frac{\partial}{\partial t}F(x(t)) \;=\; F'(x(t))\,\dot{x}(t) \equiv \; \dot{F}(x,\dot{x})$$

Forward Mode (Lighthouse)

$$
\begin{array}{rcl}
v_{-3} & = & x_1 = \nu \\
v_{-2} & = & x_2 = \gamma \\
v_{-1} & = & x_3 = \omega \\
v_0 & = & x_4 = t \\
\hline
v_1 & = & v_{-1} * v_0 \\
v_2 & = & \tan(v_1) \\
v_3 & = & v_{-2} - v_2 \\
v_4 & = & v_{-3} * v_2 \\
v_5 & = & v_4 / v_3 \\
v_6 & = & v_5 * v_{-2} \\
\hline
y_1 & = & v_5 \\
y_2 & = & v_6
\end{array}
$$

Forward Mode (Lighthouse)

$$
\begin{array}{lcl|lcl}
v_{-3} & = & x_1 = \nu & \dot{v}_{-3} & = & \dot{x}_1 \\
v_{-2} & = & x_2 = \gamma & \dot{v}_{-2} & = & \dot{x}_2 \\
v_{-1} & = & x_3 = \omega & \dot{v}_{-1} & = & \dot{x}_3 \\
v_0 & = & x_4 = t & \dot{v}_0 & = & \dot{x}_4 \\
\hline
v_1 & = & v_{-1} * v_0 & & & \\
v_2 & = & \tan(v_1) & & & \\
v_3 & = & v_{-2} - v_2 & & & \\
v_4 & = & v_{-3} * v_2 & & & \\
v_5 & = & v_4 / v_3 & & & \\
v_6 & = & v_5 * v_{-2} & & & \\
\hline
y_1 & = & v_5 & & & \\
y_2 & = & v_6 & & &
\end{array}
$$

Forward Mode (Lighthouse)

$$
\begin{array}{lll|lll}
v_{-3} & = & x_1 = \nu & \dot{v}_{-3} & = & \dot{x}_1 \\
v_{-2} & = & x_2 = \gamma & \dot{v}_{-2} & = & \dot{x}_2 \\
v_{-1} & = & x_3 = \omega & \dot{v}_{-1} & = & \dot{x}_3 \\
v_0 & = & x_4 = t & \dot{v}_0 & = & \dot{x}_4 \\
\hline
v_1 & = & v_{-1} * v_0 & \dot{v}_1 & = & \dot{v}_{-1} * v_0 + v_{-1} * \dot{v}_0 \\
v_2 & = & \tan(v_1) & & & \\
v_3 & = & v_{-2} - v_2 & & & \\
v_4 & = & v_{-3} * v_2 & & & \\
v_5 & = & v_4 / v_3 & & & \\
v_6 & = & v_5 * v_{-2} & & & \\
\hline
y_1 & = & v_5 & & & \\
y_2 & = & v_6 & & &
\end{array}
$$

Forward Mode (Lighthouse)

$$
\begin{array}{lll|lll}
v_{-3} & = & x_1 = \nu & \dot{v}_{-3} & = & \dot{x}_1 \\
v_{-2} & = & x_2 = \gamma & \dot{v}_{-2} & = & \dot{x}_2 \\
v_{-1} & = & x_3 = \omega & \dot{v}_{-1} & = & \dot{x}_3 \\
v_0 & = & x_4 = t & \dot{v}_0 & = & \dot{x}_4 \\
\hline
v_1 & = & v_{-1} * v_0 & \dot{v}_1 & = & \dot{v}_{-1} * v_0 + v_{-1} * \dot{v}_0 \\
v_2 & = & \tan(v_1) & \dot{v}_2 & = & \dot{v}_1 / \cos(v_1)^2 \\
v_3 & = & v_{-2} - v_2 & & & \\
v_4 & = & v_{-3} * v_2 & & & \\
v_5 & = & v_4 / v_3 & & & \\
v_6 & = & v_5 * v_{-2} & & & \\
\hline
y_1 & = & v_5 & & & \\
y_2 & = & v_6 & & &
\end{array}
$$

Forward Mode (Lighthouse)

$$
\begin{array}{lcl|lcl}
v_{-3} & = & x_1 = \nu & \dot{v}_{-3} & = & \dot{x}_1 \\
v_{-2} & = & x_2 = \gamma & \dot{v}_{-2} & = & \dot{x}_2 \\
v_{-1} & = & x_3 = \omega & \dot{v}_{-1} & = & \dot{x}_3 \\
v_0 & = & x_4 = t & \dot{v}_0 & = & \dot{x}_4 \\
\hline
v_1 & = & v_{-1} * v_0 & \dot{v}_1 & = & \dot{v}_{-1} * v_0 + v_{-1} * \dot{v}_0 \\
v_2 & = & \tan(v_1) & \dot{v}_2 & = & \dot{v}_1 / \cos(v_1)^2 \\
v_3 & = & v_{-2} - v_2 & \dot{v}_3 & = & \dot{v}_{-2} - \dot{v}_2 \\
v_4 & = & v_{-3} * v_2 & & & \\
v_5 & = & v_4 / v_3 & & & \\
v_6 & = & v_5 * v_{-2} & & & \\
\hline
y_1 & = & v_5 & & & \\
y_2 & = & v_6 & & &
\end{array}
$$

Forward Mode (Lighthouse)

$$
\begin{array}{lcl|lcl}
v_{-3} & = & x_1 = \nu & \dot{v}_{-3} & = & \dot{x}_1 \\
v_{-2} & = & x_2 = \gamma & \dot{v}_{-2} & = & \dot{x}_2 \\
v_{-1} & = & x_3 = \omega & \dot{v}_{-1} & = & \dot{x}_3 \\
v_0 & = & x_4 = t & \dot{v}_0 & = & \dot{x}_4 \\
\hline
v_1 & = & v_{-1} * v_0 & \dot{v}_1 & = & \dot{v}_{-1} * v_0 + v_{-1} * \dot{v}_0 \\
v_2 & = & \tan(v_1) & \dot{v}_2 & = & \dot{v}_1 / \cos(v_1)^2 \\
v_3 & = & v_{-2} - v_2 & \dot{v}_3 & = & \dot{v}_{-2} - \dot{v}_2 \\
v_4 & = & v_{-3} * v_2 & \dot{v}_4 & = & \dot{v}_{-3} * v_2 + v_{-3} * \dot{v}_2 \\
v_5 & = & v_4 / v_3 & & & \\
v_6 & = & v_5 * v_{-2} & & & \\
\hline
y_1 & = & v_5 & & & \\
y_2 & = & v_6 & & &
\end{array}
$$

Forward Mode (Lighthouse)

$$
\begin{array}{lll|lll}
v_{-3} & = & x_1 = \nu & \dot{v}_{-3} & = & \dot{x}_1 \\
v_{-2} & = & x_2 = \gamma & \dot{v}_{-2} & = & \dot{x}_2 \\
v_{-1} & = & x_3 = \omega & \dot{v}_{-1} & = & \dot{x}_3 \\
v_0 & = & x_4 = t & \dot{v}_0 & = & \dot{x}_4 \\
\hline
v_1 & = & v_{-1} * v_0 & \dot{v}_1 & = & \dot{v}_{-1} * v_0 + v_{-1} * \dot{v}_0 \\
v_2 & = & \tan(v_1) & \dot{v}_2 & = & \dot{v}_1 / \cos(v_1)^2 \\
v_3 & = & v_{-2} - v_2 & \dot{v}_3 & = & \dot{v}_{-2} - \dot{v}_2 \\
v_4 & = & v_{-3} * v_2 & \dot{v}_4 & = & \dot{v}_{-3} * v_2 + v_{-3} * \dot{v}_2 \\
v_5 & = & v_4 / v_3 & \dot{v}_5 & = & (\dot{v}_4 - \dot{v}_3 * v_5) * (1/v_3) \\
v_6 & = & v_5 * v_{-2} & & & \\
\hline
y_1 & = & v_5 & & & \\
y_2 & = & v_6 & & &
\end{array}
$$

Forward Mode (Lighthouse)

$$
\begin{array}{lll|lll}
v_{-3} & = & x_1 = \nu & \dot{v}_{-3} & = & \dot{x}_1 \\
v_{-2} & = & x_2 = \gamma & \dot{v}_{-2} & = & \dot{x}_2 \\
v_{-1} & = & x_3 = \omega & \dot{v}_{-1} & = & \dot{x}_3 \\
v_0 & = & x_4 = t & \dot{v}_0 & = & \dot{x}_4 \\
\hline
v_1 & = & v_{-1} * v_0 & \dot{v}_1 & = & \dot{v}_{-1} * v_0 + v_{-1} * \dot{v}_0 \\
v_2 & = & \tan(v_1) & \dot{v}_2 & = & \dot{v}_1 / \cos(v_1)^2 \\
v_3 & = & v_{-2} - v_2 & \dot{v}_3 & = & \dot{v}_{-2} - \dot{v}_2 \\
v_4 & = & v_{-3} * v_2 & \dot{v}_4 & = & \dot{v}_{-3} * v_2 + v_{-3} * \dot{v}_2 \\
v_5 & = & v_4 / v_3 & \dot{v}_5 & = & (\dot{v}_4 - \dot{v}_3 * v_5) * (1/v_3) \\
v_6 & = & v_5 * v_{-2} & \dot{v}_6 & = & \dot{v}_5 * v_{-2} + v_5 * \dot{v}_{-2} \\
\hline
y_1 & = & v_5 & & & \\
y_2 & = & v_6 & & &
\end{array}
$$

Forward Mode (Lighthouse)

$$
\begin{array}{lcl|lcl}
v_{-3} & = & x_1 = \nu & \dot{v}_{-3} & = & \dot{x}_1 \\
v_{-2} & = & x_2 = \gamma & \dot{v}_{-2} & = & \dot{x}_2 \\
v_{-1} & = & x_3 = \omega & \dot{v}_{-1} & = & \dot{x}_3 \\
v_0 & = & x_4 = t & \dot{v}_0 & = & \dot{x}_4 \\
\hline
v_1 & = & v_{-1} * v_0 & \dot{v}_1 & = & \dot{v}_{-1} * v_0 + v_{-1} * \dot{v}_0 \\
v_2 & = & \tan(v_1) & \dot{v}_2 & = & \dot{v}_1 / \cos(v_1)^2 \\
v_3 & = & v_{-2} - v_2 & \dot{v}_3 & = & \dot{v}_{-2} - \dot{v}_2 \\
v_4 & = & v_{-3} * v_2 & \dot{v}_4 & = & \dot{v}_{-3} * v_2 + v_{-3} * \dot{v}_2 \\
v_5 & = & v_4 / v_3 & \dot{v}_5 & = & (\dot{v}_4 - \dot{v}_3 * v_5) * (1/v_3) \\
v_6 & = & v_5 * v_{-2} & \dot{v}_6 & = & \dot{v}_5 * v_{-2} + v_5 * \dot{v}_{-2} \\
\hline
y_1 & = & v_5 & \dot{y}_1 & = & \dot{v}_5 \\
y_2 & = & v_6 & \dot{y}_2 & = & \dot{v}_6
\end{array}
$$

Complexity (Forward Mode)

tang	c	$\pm$	$*$	ψ
MOVES	$1+1$	$3+3$	$3+3$	$2+2$
ADDS	0	$1+1$	$0+1$	$0+0$
MULTS	0	0	$1+2$	$0+1$
NLOPS	0	0	0	$1+1$

Complexity (Forward Mode)

tang	c	$\pm$	$*$	ψ
MOVES	$1+1$	$3+3$	$3+3$	$2+2$
ADDS	0	$1+1$	$0+1$	$0+0$
MULTS	0	0	$1+2$	$0+1$
NLOPS	0	0	0	$1+1$

$$\Rightarrow \quad \boxed{\mathrm{OPS}(F'(x)\dot{x}) \quad \leq \quad c\,\mathrm{OPS}(F(x))}$$

with $c \in [2, 5/2]$ platform dependent

Forward Mode AD for ML

Typical function evaluation (deep neural net):

$$x = x^{(1)} \to \tilde{x}^{(1)} = W^{(1)}x^{(1)} + b^{(1)} \quad \to x^{(2)} = \rho(\tilde{x}^{(1)})$$

Attention: Optimization variables W and $b \Rightarrow \dot{W}$ and $\dot{b}$!

Forward Mode AD for ML

Typical function evaluation (deep neural net):

$$x = x^{(1)} \to \tilde{x}^{(1)} = W^{(1)}x^{(1)} + b^{(1)} \quad \to x^{(2)} = \rho(\tilde{x}^{(1)})$$

Attention: Optimization variables W and $b \Rightarrow \dot{W}$ and $\dot{b}$!

$$\begin{aligned} x = x^{(1)} \to \tilde{x}^{(1)} &= W^{(1)}x^{(1)} + b^{(1)} &&\to x^{(2)} = \rho(\tilde{x}^{(1)}) \\ \dot{\tilde{x}}^{(1)} &= \dot{W}^{(1)}x^{(1)} + \dot{b}^{(1)} &&\to \dot{x}^{(2)} = \rho'(\tilde{x}^{(1)})\dot{\tilde{x}}^{(1)} \end{aligned}$$

Forward Mode AD for ML

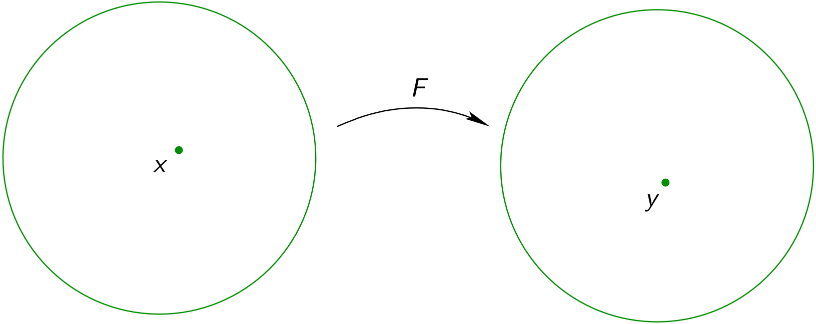
Typical function evaluation (deep neural net):

$$x = x^{(1)} \to \tilde{x}^{(1)} = W^{(1)}x^{(1)} + b^{(1)} \quad \to x^{(2)} = \rho(\tilde{x}^{(1)})$$

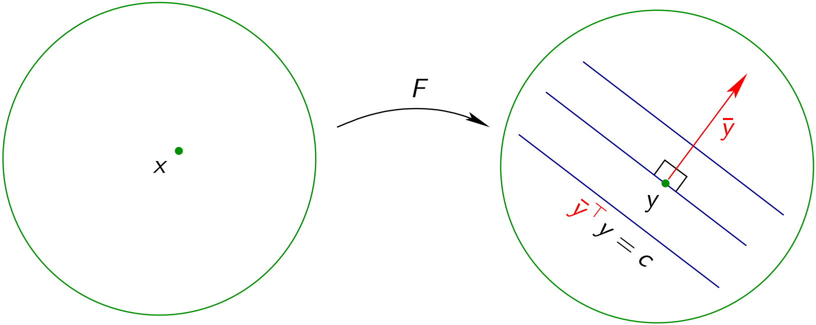
Attention: Optimization variables W and $b \Rightarrow \dot{W}$ and $\dot{b}$!

$$\begin{aligned}
x = x^{(1)} \to & \tilde{x}^{(1)} = W^{(1)}x^{(1)} + b^{(1)} \quad \to x^{(2)} = \rho(\tilde{x}^{(1)}) \\
& \dot{\tilde{x}}^{(1)} = \dot{W}^{(1)}x^{(1)} + \dot{b}^{(1)} \quad \to \dot{x}^{(2)} = \rho'(\tilde{x}^{(1)})\dot{\tilde{x}}^{(1)} \\
\to & \tilde{x}^{(2)} = W^{(2)}x^{(2)} + b^{(2)} \quad \to x^{(3)} = \rho(\tilde{x}^{(2)}) \\
& \dot{\tilde{x}}^{(2)} = \dot{W}^{(2)}x^{(2)} + W^{(2)}\dot{x}^{(2)} + \dot{b}^{(2)} \quad \to \dot{x}^{(3)} = \rho'(\tilde{x}^{(3)})\dot{\tilde{x}}^{(3)} \\
\to & \cdots \\
\to & y = W^{(k)}x^{(k)} + b^{(k)} \\
\to & \dot{y} = \dot{W}^{(2)}x^{(2)} + W^{(2)}\dot{x}^{(2)} + \dot{b}^{(2)}
\end{aligned}$$

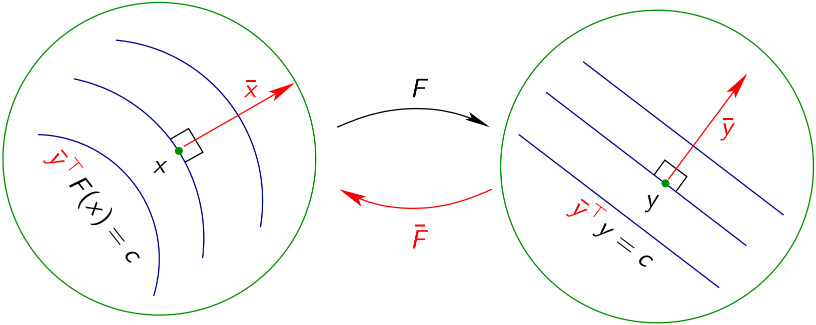
Reverse Mode AD = Discrete Adjoints



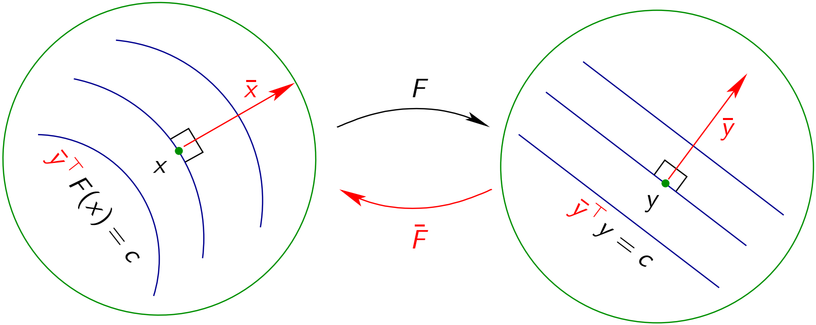
Reverse Mode AD = Discrete Adjoints



Reverse Mode AD = Discrete Adjoints



Reverse Mode AD = Discrete Adjoints



$$\bar{x} \;\equiv\; \bar{y}^\top F'(x) \;=\; \nabla_x \langle\, \bar{y}^\top F(x) \,\rangle \;\equiv\; \bar{F}(x, \bar{y})$$

Reverse Mode (Lighthouse)

$$
\begin{array}{l}
v_{-3} = x_1; \quad v_{-2} = x_2; \quad v_{-1} = x_3; \quad v_0 = x_4;\\
\qquad v_1 = v_{-1} * v_0;\\
\qquad\qquad v_2 = \tan(v_1);\\
\qquad\qquad\qquad v_3 = v_{-2} - v_2;\\
\qquad\qquad\qquad\qquad v_4 = v_{-3} * v_2;\\
\qquad\qquad\qquad\qquad\qquad v_5 = v_4 / v_3;\\
\qquad\qquad\qquad\qquad\qquad\qquad v_6 = v_5 * v_{-2};\\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad y_1 = v_5; \quad y_2 = v_6;\\
\hline
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \bar{v}_5 = \bar{y}_1; \quad \bar{v}_6 = \bar{y}_2;\\
\qquad\qquad\qquad\qquad\qquad\qquad \bar{v}_5 \mathrel{+}= \bar{v}_6 * v_{-2}; \quad \bar{v}_{-2} \mathrel{+}= \bar{v}_6 * v_5;\\
\qquad\qquad\qquad\qquad\qquad \bar{v}_4 \mathrel{+}= \bar{v}_5 / v_3; \quad \bar{v}_3 \mathrel{-}= \bar{v}_5 * v_5 / v_3;\\
\qquad\qquad\qquad\qquad \bar{v}_{-3} \mathrel{+}= \bar{v}_4 * v_2; \quad \bar{v}_2 \mathrel{+}= \bar{v}_4 * v_{-3};\\
\qquad\qquad\qquad \bar{v}_{-2} \mathrel{+}= \bar{v}_3; \ \bar{v}_2 \mathrel{-}= \bar{v}_3;\\
\qquad\qquad \bar{v}_1 \mathrel{+}= \bar{v}_2 / \cos^2(v_1);\\
\qquad \bar{v}_{-1} \mathrel{+}= \bar{v}_1 * v_0; \ \bar{v}_0 \mathrel{+}= \bar{v}_1 * v_{-1};\\
\bar{x}_4 = \bar{v}_0; \quad \bar{x}_3 = \bar{v}_{-1}; \quad \bar{x}_2 = \bar{v}_{-2}; \quad \bar{x}_1 = \bar{v}_{-3};
\end{array}
$$

Complexity (Reverse Mode)

grad	c	$\pm$	$*$	ψ
MOVES	$1+1$	$3+6$	$3+8$	$2+5$
ADDS	0	$1+2$	$0+2$	$0+1$
MULTS	0	0	$1+2$	$0+1$
NLOPS	0	0	0	$1+1$

$$\boxed{\mathrm{OPS}(\bar{y}^\top F'(x)) \leq c\,\mathrm{OPS}(F(x)),\ \mathrm{MEM}(\bar{y}^\top F'(x)) \sim \mathrm{OPS}(F(x))}$$

with $c \in [3,4]$ platform dependent

Complexity (Reverse Mode)

grad	c	$\pm$	$*$	ψ
MOVES	$1+1$	$3+6$	$3+8$	$2+5$
ADDS	0	$1+2$	$0+2$	$0+1$
MULTS	0	0	$1+2$	$0+1$
NLOPS	0	0	0	$1+1$

$$\Longrightarrow \boxed{\mathrm{OPS}(\bar{y}^\top F'(x)) \leq c\,\mathrm{OPS}(F(x)),\ \mathrm{MEM}(\bar{y}^\top F'(x)) \sim \mathrm{OPS}(F(x))}$$

with $c \in [3,4]$ platform dependent

Remarks:

- Cost for gradient calculation independent of n
- Memory requirement may cause problem! $\Rightarrow$ Checkpointing

Reverse Mode AD for ML

Typical function evaluation (deep neural net):

$$
\begin{aligned}
x = x^{(1)} &\rightarrow \tilde{x}^{(1)} = W^{(1)}x^{(1)} + b^{(1)} &&\rightarrow x^{(2)} = \rho(\tilde{x}^{(1)}) \\
&\rightarrow \tilde{x}^{(2)} = W^{(2)}x^{(2)} + b^{(2)} &&\rightarrow x^{(3)} = \rho(\tilde{x}^{(2)}) \\
&\rightarrow \cdots \\
&\rightarrow y = W^{(k)}x^{(k)} + b^{(k)}
\end{aligned}
$$

Reverse Mode AD for ML

Typical function evaluation (deep neural net):

$$\begin{aligned} x = x^{(1)} &\to \tilde{x}^{(1)} = W^{(1)}x^{(1)} + b^{(1)} &&\to x^{(2)} = \rho(\tilde{x}^{(1)}) \\ &\to \tilde{x}^{(2)} = W^{(2)}x^{(2)} + b^{(2)} &&\to x^{(3)} = \rho(\tilde{x}^{(2)}) \\ &\to \cdots \\ &\to y = W^{(k)}x^{(k)} + b^{(k)} \end{aligned}$$

With $\bar{y} = 1$ one obtains

$$\bar{W}^{(k)} = [x^{(k)}], \quad \bar{x}^{(k)} = W^{(k)}, \quad \bar{b}^{(k)} = \mathbb{1}$$

Reverse Mode AD for ML

Typical function evaluation (deep neural net):

$$
\begin{aligned}
x = x^{(1)} &\to \tilde{x}^{(1)} = W^{(1)}x^{(1)} + b^{(1)} \quad \to x^{(2)} = \rho(\tilde{x}^{(1)}) \\
&\to \tilde{x}^{(2)} = W^{(2)}x^{(2)} + b^{(2)} \quad \to x^{(3)} = \rho(\tilde{x}^{(2)}) \\
&\to \cdots \\
&\to y = W^{(k)}x^{(k)} + b^{(k)}
\end{aligned}
$$

With $\bar{y} = 1$ one obtains

$$
\begin{aligned}
&\bar{W}^{(k)} = [x^{(k)}], \quad \bar{x}^{(k)} = W^{(k)}, \quad \bar{b}^{(k)} = \mathbb{1} \\
&\bar{\tilde{x}}^{(2)} = \rho'(x^{(2)}) * \bar{x}^{(3)}, \quad \bar{W}^{(2)} = x^{(2)} * \bar{\tilde{x}}^{(2)}, \bar{x}^{(2)} = W^{(2)} * \bar{\tilde{x}}^{(2)}, \quad \bar{b}^{(2)} = \bar{\tilde{x}}^{(2)} \\
&\bar{\tilde{x}}^{(1)} = \rho'(x^{(1)}) * \bar{x}^{(2)}, \quad \bar{W}^{(1)} = x^{(1)} * \bar{\tilde{x}}^{(1)}, \bar{x}^{(1)} = W^{(1)} * \bar{\tilde{x}}^{(1)}, \quad \bar{b}^{(1)} = \bar{\tilde{x}}^{(1)}
\end{aligned}
$$

Berlin Mathematics Research Center MATH+

Reverse Mode AD for ML

Typical function evaluation (deep neural net):

$$\begin{aligned} x = x^{(1)} &\to \tilde{x}^{(1)} = W^{(1)}x^{(1)} + b^{(1)} &&\to x^{(2)} = \rho(\tilde{x}^{(1)}) \\ &\to \tilde{x}^{(2)} = W^{(2)}x^{(2)} + b^{(2)} &&\to x^{(3)} = \rho(\tilde{x}^{(2)}) \\ &\to \cdots \\ &\to y = W^{(k)}x^{(k)} + b^{(k)} \end{aligned}$$

With $\bar{y} = 1$ one obtains

$$\bar{W}^{(k)} = [x^{(k)}], \quad \bar{x}^{(k)} = W^{(k)}, \quad \bar{b}^{(k)} = \mathbb{1}$$

$$\bar{\tilde{x}}^{(2)} = \rho'(x^{(2)}) * \bar{x}^{(3)}, \quad \bar{W}^{(2)} = x^{(2)} * \bar{\tilde{x}}^{(2)}, \bar{x}^{(2)} = W^{(2)} * \bar{\tilde{x}}^{(2)}, \quad \bar{b}^{(2)} = \bar{\tilde{x}}^{(2)}$$

$$\bar{\tilde{x}}^{(1)} = \rho'(x^{(1)}) * \bar{x}^{(2)}, \quad \bar{W}^{(1)} = x^{(1)} * \bar{\tilde{x}}^{(1)}, \bar{x}^{(1)} = W^{(1)} * \bar{\tilde{x}}^{(1)}, \quad \bar{b}^{(1)} = \bar{\tilde{x}}^{(1)}$$

very simple to implement!

Overview AD Theory and Tools

- Differentiation of computer programmes with working accuracy (Griewank, Kulshreshtha, Walther 2012)

Overview AD Theory and Tools

- Differentiation of computer programmes with working accuracy (Griewank, Kulshreshtha, Walther 2012)
- $$\text{Forward mode:} \quad \mathrm{OPS}(F'(x)\dot{x}) \quad \leq \quad c\,\mathrm{OPS}(F), \quad c \in [2, 5/2]$$

Overview AD Theory and Tools

- Differentiation of computer programmes with working accuracy (Griewank, Kulshreshtha, Walther 2012)
- $$\text{Forward mode:} \quad \mathrm{OPS}(F'(x)\dot{x}) \quad \leq \quad c\,\mathrm{OPS}(F), \quad c \in [2, 5/2]$$

  $=$ discrete analogon to sensitivity equation

Overview AD Theory and Tools

- Differentiation of computer programmes with working accuracy (Griewank, Kulshreshtha, Walther 2012)
-

$$
\begin{array}{llcll}
\text{Forward mode:} & \mathrm{OPS}(F'(x)\dot{x}) & \leq & c\,\mathrm{OPS}(F), & c \in [2, 5/2] \\
\text{Reverse mode:} & \mathrm{OPS}(\bar{y}^\top F'(x)) & \leq & c\,\mathrm{OPS}(F), & c \in [3, 4] \\
 & \mathrm{MEM}(\bar{y}^\top F'(x)) & \sim & \mathrm{OPS}(F), &
\end{array}
$$

Overview AD Theory and Tools

- Differentiation of computer programmes with working accuracy (Griewank, Kulshreshtha, Walther 2012)
-
$$\begin{array}{llcll} \text{Forward mode:} & \mathrm{OPS}(F'(x)\dot{x}) & \leq & c\,\mathrm{OPS}(F), & c \in [2, 5/2] \\ \text{Reverse mode:} & \mathrm{OPS}(\bar{y}^\top F'(x)) & \leq & c\,\mathrm{OPS}(F), & c \in [3, 4] \\ & \mathrm{MEM}(\bar{y}^\top F'(x)) & \sim & \mathrm{OPS}(F), & \end{array}$$

$$= \text{discrete analogon to adjoint equation}$$

see also talk of Benjamin Sanderse

Overview AD Theory and Tools

- Differentiation of computer programmes with working accuracy (Griewank, Kulshreshtha, Walther 2012)
-

$$\begin{array}{llcll}
\text{Forward mode:} & \mathrm{OPS}(F'(x)\dot{x}) & \leq & c\,\mathrm{OPS}(F), & c \in [2, 5/2] \\
\text{Reverse mode:} & \mathrm{OPS}(\bar{y}^\top F'(x)) & \leq & c\,\mathrm{OPS}(F), & c \in [3, 4] \\
 & \mathrm{MEM}(\bar{y}^\top F'(x)) & \sim & \mathrm{OPS}(F), &
\end{array}$$

$\Longrightarrow$ Gradients are cheap $\sim$ Function costs!!

Overview AD Theory and Tools

- Differentiation of computer programmes with working accuracy (Griewank, Kulshreshtha, Walther 2012)
- $$\begin{array}{llcll} \text{Forward mode:} & \mathrm{OPS}(F'(x)\dot{x}) & \leq & c\,\mathrm{OPS}(F), & c \in [2, 5/2] \\ \text{Reverse mode:} & \mathrm{OPS}(\bar{y}^\top F'(x)) & \leq & c\,\mathrm{OPS}(F), & c \in [3, 4] \\ & \mathrm{MEM}(\bar{y}^\top F'(x)) & \sim & \mathrm{OPS}(F), & \end{array}$$

  $\Longrightarrow$ Gradients are cheap $\sim$ Function costs!!
- Combination: $\mathrm{OPS}(\bar{y}^\top F''(x)\dot{x}) \leq c\,\mathrm{OPS}(F), \;\; c \in [7, 10]$
- Consistent derivative information!

Overview AD Theory and Tools

- Differentiation of computer programmes with working accuracy (Griewank, Kulshreshtha, Walther 2012)
-
$$\begin{array}{llcll} \text{Forward mode:} & \mathrm{OPS}({\color{blue}F'(x)\dot{x}}) & \leq & c\,\mathrm{OPS}(F), & c \in [2, 5/2] \\ \text{Reverse mode:} & \mathrm{OPS}({\color{red}\bar{y}^\top F'(x)}) & \leq & c\,\mathrm{OPS}(F), & c \in [3, 4] \\ & \mathrm{MEM}({\color{red}\bar{y}^\top F'(x)}) & \sim & \mathrm{OPS}(F), & \end{array}$$

$$\Longrightarrow \quad \text{Gradients are cheap} \sim \text{Function costs!!}$$

- Combination: $\mathrm{OPS}({\color{green}\bar{y}^\top F''(x)\dot{x}}) \leq c\,\mathrm{OPS}(F), \; c \in [7, 10]$
- Consistent derivative information!
- Structure exploitation indispensable
- AD in real-life applications, e.g., backpropagation for ML

Overview AD Theory and Tools

- Differentiation of computer programmes with working accuracy (Griewank, Kulshreshtha, Walther 2012)
- $$\begin{array}{llcll} \text{Forward mode:} & \mathrm{OPS}(F'(x)\dot{x}) & \leq & c\,\mathrm{OPS}(F), & c \in [2, 5/2] \\ \text{Reverse mode:} & \mathrm{OPS}(\bar{y}^\top F'(x)) & \leq & c\,\mathrm{OPS}(F), & c \in [3, 4] \\ & \mathrm{MEM}(\bar{y}^\top F'(x)) & \sim & \mathrm{OPS}(F), & \end{array}$$

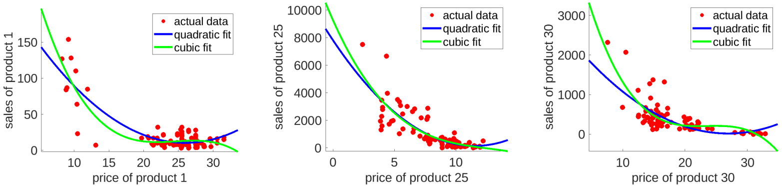
  $\Longrightarrow$ Gradients are cheap $\sim$ Function costs!!
- Combination: $\mathrm{OPS}(\bar{y}^\top F''(x)\dot{x}) \leq c\,\mathrm{OPS}(F), \quad c \in [7, 10]$
- Consistent derivative information!
- Structure exploitation indispensable
- AD in real-life applications, e.g., backpropagation for ML

(Griewank, Walther 2008), (Naumann 2012), www.autodiff.org

Automatic Differentiation by OverLoading in C++

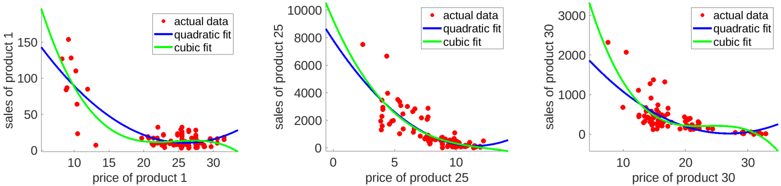
- ADOL-C version 2.7, available at COIN-OR since 2009
  open source (GPL or ECL)
- based on operator overloading, trace as internal representation
- general-purpose AD tool with focus on functionalities
- interfaces to ColPack (Purdue University) and Ipopt (COIN-OR)
- current developments
  - exploitation of fixed-point structure for second-order derivatives
  - generalized derivatives for nonsmooth functions

Finding the Demand Function



data from Cohen, Perakis and Pindyck, Pricing with Limited Knowledge of Demand, 2016

Finding the Demand Function



data from Cohen, Perakis and Pindyck, Pricing with Limited Knowledge of Demand, 2016

Common approach: Use piecewise linear demand functions (PLF)

1. $\max(a_1 - b_1 p, a_2 - b_2 p)$ $\Rightarrow$ non-convex formulation
2. $\min(a_3 - b_3 p, a_4 - b_4 p)$ $\Rightarrow$ convex formulation
3. $\max(a_5 - b_5 p, 0)$ $\Rightarrow$ non-convex formulation

The Piecewise Linear Regression Problem

Yields the piecewise linear problem

$$\min_{a\in\mathbb{R}^n, b\in\mathbb{R}^n} \sum_{i\in\mathcal{I}} \sum_{t\in\mathcal{T}} \left| \underbrace{f_i(a,b,p)}_{\text{PLF}} - d^t_{obs} \right|$$

such that $a, b \geq 0.$

The Piecewise Linear Regression Problem

Yields the piecewise linear problem

$$\min_{a\in\mathbb{R}^n, b\in\mathbb{R}^n} \sum_{i\in\mathcal{I}} \sum_{t\in\mathcal{T}} \left| \underbrace{f_i(a,b,p)}_{\text{PLF}} - d^t_{obs} \right|$$

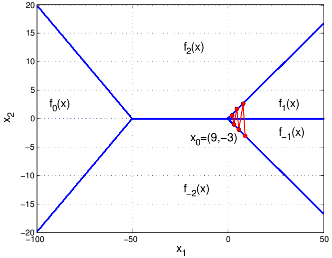
$$\text{such that } \; a, b \geq 0.$$

Calculation of solution using smoothing or heuristics! Why?

Observations

Even $\min f(x)$ with piecewise linear (PL) convex f not easy!

- Global minimization is NP-hard
- Steepest descent with exact line search may fail
- Zeno behaviour possible, i.e., solution trajectory with infinite number of direction changes in a finite amount of time



J.-B. Hiriart-Urruty, C. Lemaréchal: Convex Analysis and Minimization Algorithms I, Springer, 1993

The Revenue Maximization Problem

Based on the determined demand function, one obtains

$$\min_{p,u,I} h(p) = \sum_{i\in\mathcal{I}} \sum_{t\in\mathcal{T}} \Big(\underbrace{-p_i^t d_i^t(p_i^t)}_{\text{revenue}} + \underbrace{\phi(d_i^t)(p_i^t)}_{\text{costs}} \Big)$$

s.t. $I_i^{t+1} = I_i^t + u_i^t - d_i^t(p_i^t)$		Inventory dynamics constraint
$\rho_i^l \le p_i^t \le \rho_i^u,$	$\forall t \in \mathcal{T}_p$	Promotion constraint
$\theta_i^l \le p_i^t \le \theta_i^u,$	$\forall t \in \mathcal{T}_m$	Markdown constraint
$p_i^t - p_j^t \ge \kappa_{ij},$	$\forall \{i,j\} \in \mathcal{I}\times\mathcal{I}$	Inter-Item constraints
$u_i^t = 0,$	$\forall t \in \mathcal{T}_i$	Non-replenishment time-slots
$I_i^t, u_i^t, p_i^t \ge 0,$	$\forall i \in \mathcal{I}, t \in \mathcal{T}$	Non-negativity constraints

with price p, inventory I and replenishment u

Cohen et al.,The Impact of Linear Optimization on Promotion Planning, 2017.
Kannan et al., Computerized promotion and markdown price scheduling, 2020.

Representations of PL Functions

There are many choices

Representations of PL Functions

There are many choices, e.g., (Scholtes, 2012)

Theorem (Max-Min representation of PL functions)

For each PL $f : \mathbb{R}^n \mapsto \mathbb{R}$ *with selection functions* $f_j(x) = a_j^\top x + b_j$, $1 \leq j \leq k$, *there exist index sets* $M_i \subset \{1, \ldots, k\}$, $1 \leq i \leq l$, *such that*

$$f(x) = \max_{1 \leq i \leq l} \min_{j \in M_i} a_j^\top x + b_j \ .$$

Representations of PL Functions

There are many choices, e.g., (Scholtes, 2012)

Theorem (Max-Min representation of PL functions)

For each PL $f : \mathbb{R}^n \mapsto \mathbb{R}$ with selection functions $f_j(x) = a_j^\top x + b_j$, $1 \leq j \leq k$, there exist index sets $M_i \subset \{1, \ldots, k\}$, $1 \leq i \leq l$, such that

$$f(x) = \max_{1 \leq i \leq l} \min_{j \in M_i} a_j^\top x + b_j \ .$$

However, not constructive! But:

Lemma (Abs-linear form of piecewise linear $f : \mathbb{R}^n \to \mathbb{R}$)

Each PL $f : \mathbb{R}^n \mapsto \mathbb{R}$ has an abs-linear form given by

$$\begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + \begin{bmatrix} Z & M & L \\ a & b & 0 \end{bmatrix} \begin{bmatrix} x \\ z \\ |z| \end{bmatrix} .$$

Follows by refomulation von max and min!

Representations of PL Functions

There are many choices, e.g., (Scholtes, 2012)

Theorem (Max-Min representation of PL functions)

For each PL $f : \mathbb{R}^n \mapsto \mathbb{R}$ with selection functions $f_j(x) = a_j^\top x + b_j$, $1 \leq j \leq k$, there exist index sets $M_i \subset \{1, \ldots, k\}$, $1 \leq i \leq l$, such that

$$f(x) = \max_{1 \leq i \leq l} \min_{j \in M_i} a_j^\top x + b_j \ .$$

However, not constructive! But:

Lemma (Abs-linear form of piecewise linear $f : \mathbb{R}^n \to \mathbb{R}$)

Each PL $f : \mathbb{R}^n \mapsto \mathbb{R}$ has an abs-linear form given by

$$\begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + \begin{bmatrix} Z & M & L \\ a & b & 0 \end{bmatrix} \begin{bmatrix} x \\ z \\ |z| \end{bmatrix} .$$

Follows by refomulation von max and min! Can be generated by AD!

Signature Domaines

Definition ((Extended) Signature domain)

For a fixed $\sigma \in \{-1, 0, 1\}^s$ and $f \in \mathcal{C}^d_{\text{abs}}(\mathbb{R}^n)$, we define

$$\mathcal{P}_\sigma \equiv \{x \in \mathbb{R}^n \mid \operatorname{sgn}(z(x)) = \sigma\} \subset \bar{\mathcal{P}}_\sigma \equiv \{x \in \mathbb{R}^n \mid \Sigma z(x) = |z(x)|\} \; .$$

$\mathcal{P}_\sigma$ is called *signature domain* and $\bar{\mathcal{P}}_\sigma$ *extended signature domain*.

Signature Domaines

Definition ((Extended) Signature domain)

For a fixed $\sigma \in \{-1, 0, 1\}^s$ and $f \in \mathcal{C}^d_{\text{abs}}(\mathbb{R}^n)$, we define

$$\mathcal{P}_\sigma \equiv \{x \in \mathbb{R}^n \mid \text{sgn}(z(x)) = \sigma\} \subset \bar{\mathcal{P}}_\sigma \equiv \{x \in \mathbb{R}^n \mid \Sigma z(x) = |z(x)|\}\ .$$

$\mathcal{P}_\sigma$ is called *signature domain* and $\bar{\mathcal{P}}_\sigma$ *extended signature domain*.

- the signature domains form a disjoint decomposition of $\mathbb{R}^n$
- for a PL function f
  - each signature domain $\mathcal{P}_\sigma$ is a polyhedron and
  - f is linear on $\mathcal{P}_\sigma$

Signature Domaines

Definition ((Extended) Signature domain)

For a fixed $\sigma \in \{-1, 0, 1\}^s$ and $f \in \mathcal{C}^d_{\text{abs}}(\mathbb{R}^n)$, we define

$$\mathcal{P}_\sigma \equiv \{x \in \mathbb{R}^n \mid \operatorname{sgn}(z(x)) = \sigma\} \subset \bar{\mathcal{P}}_\sigma \equiv \{x \in \mathbb{R}^n \mid \Sigma z(x) = |z(x)|\} .$$

$\mathcal{P}_\sigma$ is called *signature domain* and $\bar{\mathcal{P}}_\sigma$ *extended signature domain*.

- the signature domains form a disjoint decomposition of $\mathbb{R}^n$
- for a PL function f
  - each signature domain $\mathcal{P}_\sigma$ is a polyhedron and
  - f is linear on $\mathcal{P}_\sigma$

Algorihmic idea:
Minimize PL function on $\mathcal{P}_\sigma$

Signature Domaines

Definition ((Extended) Signature domain)

For a fixed $\sigma \in \{-1,0,1\}^s$ and $f \in \mathcal{C}^d_{\text{abs}}(\mathbb{R}^n)$, we define

$$\mathcal{P}_\sigma \equiv \{x \in \mathbb{R}^n \mid \text{sgn}(z(x)) = \sigma\} \subset \bar{\mathcal{P}}_\sigma \equiv \{x \in \mathbb{R}^n \mid \Sigma z(x) = |z(x)|\}\ .$$

$\mathcal{P}_\sigma$ is called *signature domain* and $\bar{\mathcal{P}}_\sigma$ *extended signature domain*.

- the signature domains form a disjoint decomposition of $\mathbb{R}^n$
- for a PL function f
  - each signature domain $\mathcal{P}_\sigma$ is a polyhedron and
  - f is linear on $\mathcal{P}_\sigma$

Algorihmic idea:
Minimize PL function on $\mathcal{P}_\sigma$

But 2^s signature vectors!

Signature Domaines

Definition ((Extended) Signature domain)

For a fixed $\sigma \in \{-1, 0, 1\}^s$ and $f \in \mathcal{C}^d_{\text{abs}}(\mathbb{R}^n)$, we define

$$\mathcal{P}_\sigma \equiv \{x \in \mathbb{R}^n \mid \operatorname{sgn}(z(x)) = \sigma\} \subset \bar{\mathcal{P}}_\sigma \equiv \{x \in \mathbb{R}^n \mid \Sigma z(x) = |z(x)|\} \,.$$

$\mathcal{P}_\sigma$ is called *signature domain* and $\bar{\mathcal{P}}_\sigma$ *extended signature domain*.

- the signature domains form a disjoint decomposition of $\mathbb{R}^n$
- for a PL function f
  - each signature domain $\mathcal{P}_\sigma$ is a polyhedron and
  - f is linear on $\mathcal{P}_\sigma$

Algorihmic idea:
Minimize PL function on $\mathcal{P}_\sigma$
Therefore: Choose next $\mathcal{P}_{\tilde{\sigma}}$ carefully!

But 2^s signature vectors!

Example: A Nesterov-Rosenbrock Function

The Nesterov-Rosenbrock function

$$f : \mathbb{R}^n \mapsto \mathbb{R}, \quad f(x) = \tfrac{1}{4}\,|x_1 - 1| + \sum_{i=1}^{n-1} |x_{i+1} - 2|x_i| + 1|$$

has 2^{n-1} Clarke-stationary points!

M. Gürbüzbalaban, M. Overton, On Nesterov's nonsmooth Chebyshev–Rosenbrock functions, Nonlinear Anal: Theory, 2012

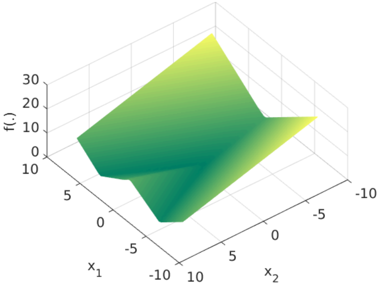
Example: A Nesterov-Rosenbrock Function

The Nesterov-Rosenbrock function

$$f : \mathbb{R}^n \mapsto \mathbb{R}, \quad f(x) = \frac{1}{4}\,|x_1 - 1| + \sum_{i=1}^{n-1} |x_{i+1} - 2|x_i| + 1|$$

has 2^{n-1} Clarke-stationary points!

M. Gürbüzbalaban, M. Overton, On Nesterov's nonsmooth Chebyshev–Rosenbrock functions, Nonlinear Anal: Theory, 2012



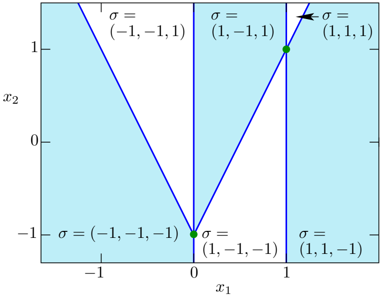
Example: A Nesterov-Rosenbrock Function

The Nesterov-Rosenbrock function

$$f : \mathbb{R}^n \mapsto \mathbb{R}, \quad f(x) = \frac{1}{4}\,|x_1 - 1| + \sum_{i=1}^{n-1} |x_{i+1} - 2|x_i| + 1|$$

has 2^{n-1} Clarke-stationary points!

M. Gürbüzbalaban, M. Overton, On Nesterov's nonsmooth Chebyshev–Rosenbrock functions, Nonlinear Anal: Theory, 2012



Active Signature Method (ASM)

= Optimization of unconstrained, piecewise linear functions

- minimization over a sequence of polyhedra
- new optimality conditions that can be verified in polynomial time
- corresponding adapted QP solver on each polyhedron
- convergence in finitely many steps

For the first time convergence to local minimizers!

Active Signature Method (ASM)

= Optimization of unconstrained, piecewise linear functions

- minimization over a sequence of polyhedra
- new optimality conditions that can be verified in polynomial time
- corresponding adapted QP solver on each polyhedron
- convergence in finitely many steps

For the first time convergence to local minimizers!

Example: Nesterov-Rosenbrock function (2^{n-1} Clarke-stationary points!)

$$f : \mathbb{R}^n \mapsto \mathbb{R}, \quad f(x) = \frac{1}{4}\,|x_1 - 1| \, + \sum_{i=1,\ldots,n-1} |x_{i+1} - 2|x_i| + 1|$$

Active Signature Method (ASM)

= Optimization of unconstrained, piecewise linear functions

- minimization over a sequence of polyhedra
- new optimality conditions that can be verified in polynomial time
- corresponding adapted QP solver on each polyhedron
- convergence in finitely many steps

For the first time convergence to local minimizers!

Example: Nesterov-Rosenbrock function (2^{n-1} Clarke-stationary points!)

$$f : \mathbb{R}^n \mapsto \mathbb{R}, \quad f(x) = \frac{1}{4}\,|x_1 - 1| \; + \sum\nolimits_{i=1,\ldots,n-1} |x_{i+1} - 2|x_i| + 1|$$

Iterations numbers:

n	1	2	3	4	5	6	7	8	9	10
ASM+QP	2	4	8	16	32	64	128	256	512	1024
HANSO	3	61	494*	1341*	2521*	329*	357*	326*	307*	515*
MPBNGC	3	52	9859	9978*	3561*	4166*	2547*	1959*	9420*	9807*

* = stop at non-optimal, stationary point

A. Griewank, A. Walther: Finite convergence of an active signature method to local minima of piecewise linear functions. OMS, 2019

A Constrained Case

Add PL constraints, i.e.,

$$
\begin{aligned}
&\min_{x\in\mathbb{R}^n, z\in\mathbb{R}^s} \quad a^\top x + b^\top z \\
\text{s.t.} \quad & 0 = g + Ax + Bz + C|z| \,, \\
& 0 \geq h + Dx + Ez + F|z| \,, \\
& z = c + Zx + Mz + L|z| \,,
\end{aligned}
$$

Hence, target function might be unbounded.

A Constrained Case

Add PL constraints, i.e.,

$$
\begin{aligned}
&\min_{x\in\mathbb{R}^n, z\in\mathbb{R}^s} \quad a^\top x + b^\top z \\
\text{s.t.}\quad & 0 = g + Ax + Bz + C|z| \ , \\
& 0 \geq h + Dx + Ez + F|z| \ , \\
& z = c + Zx + Mz + L|z| \ ,
\end{aligned}
$$

Hence, target function might be unbounded.

- generalization of LIKQ and optimality conditions possible yields Constrained Active Signature Method (CASM)
- same convergence results

PhD thesis of T. Kreimeier

Paper with algorithm and convergence analysis in preparation

Solving the PL Regression Problem

$$\min_{a,b\in\mathbb{R}^{|\mathcal{I}|}} \quad \sum_{i\in\mathcal{I}} \sum_{t\in\mathcal{T}} \left|\max(a_i - b_i p_i^t, 0) - d_i^t\right| \quad \text{s.t.} \quad a, b \geq 0 \qquad (1)$$

Solving the PL Regression Problem

$$\min_{a,b \in \mathbb{R}^{|\mathcal{I}|}} \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} \left| \max(a_i - b_i p_i^t, 0) - d_i^t \right| \quad \text{s.t.} \quad a, b \geq 0 \tag{1}$$

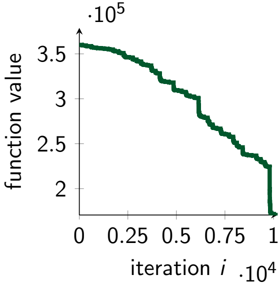
$$\min_{a_i, b_i \in \mathbb{R}} \sum_{t \in \mathcal{T}} \left| \max(a_i - b_i p_i^t, 0) - d_i^t \right| \quad \text{s.t.} \quad a_i, b_i \geq 0 \;\; \forall i \in \mathcal{I} \tag{2}$$

Berlin Mathematics Research Center

Solving the PL Regression Problem

$$\min_{a,b\in\mathbb{R}^{|\mathcal{I}|}} \sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}} \left|\max(a_i - b_i p_i^t, 0) - d_i^t\right| \quad \text{s.t.} \quad a, b \geq 0 \tag{1}$$

$$\min_{a_i,b_i\in\mathbb{R}} \sum_{t\in\mathcal{T}} \left|\max(a_i - b_i p_i^t, 0) - d_i^t\right| \quad \text{s.t.} \quad a_i, b_i \geq 0 \ \ \forall i \in \mathcal{I} \tag{2}$$



optimization problem	(1)	(2)
variables n	88	2
equal. const. m	0	0
inequal. const. p	88	2
switching variables s	8625	197
rows/columns of saddle point sys.	17426	398
iterations	10215	10303
runtime (sec.)	765	27

Comparison of Different Demand Functions

data set	mean absolute error (scaled)		
	$\max(.,0)$	$\max(.,.)$	$\min(.,.)$
Cohen	46.4562	42.7222	47.8897
UCI	19.9365	6.1840	8.6981
Logit	5.6640	5.6637	0.7258

Comparison of different piecewise linear functions

Quadratic Constrained ASM (QCASM)

- based on the idea of CASM

Quadratic Constrained ASM (QCASM)

- based on the idea of CASM
- solves problems of the form

$$
\begin{aligned}
\min_{x\in\mathbb{R}^n, z\in\mathbb{R}^s} \quad & x^\top Q\, x + \qquad\qquad a^\top x + b^\top z + d \\
\text{s.t.} \quad & 0 = g + Ax + Bz + C|z| \\
& 0 \geq h + Dx + Ez + F|z| \\
& z = c + Zx + Mz + L|z|
\end{aligned}
$$

Quadratic Constrained ASM (QCASM)

- based on the idea of CASM
- solves problems of the form

$$
\begin{aligned}
\min_{x\in\mathbb{R}^n, z\in\mathbb{R}^s} \quad & x^\top Q_1 x + x^\top Q_2 z + z^\top Q_3 z + a^\top x + b^\top z + d \\
\text{s.t.} \quad & 0 = g + Ax + Bz + C|z| \\
& 0 \geq h + Dx + Ez + F|z| \\
& z = c + Zx + Mz + L|z|
\end{aligned}
$$

Quadratic Constrained ASM (QCASM)

- based on the idea of CASM
- solves problems of the form

$$\begin{aligned} \min_{x\in\mathbb{R}^n, z\in\mathbb{R}^s} \quad & x^\top Q_1 x + x^\top Q_2 z + z^\top Q_3 z + a^\top x + b^\top z + d \\ \text{s.t.} \quad & 0 = g + Ax + Bz + C|z| \\ & 0 \geq h + Dx + Ez + F|z| \\ & z = c + Zx + Mz + L|z| \end{aligned}$$

- also optimality condition to determine next neighboring polyhedron
- descent and finite convergence ensured

Results for Cohen's Data Set

For 44 products, we obtained:

iteration	revenue (in multiples of 10^6)			
	product 4	product 7	product 8	total
1	0.07529	0.11084	1.14629	24.4017
20	0.07734	0.13073	1.17985	26.8673
50	0.08075	0.16388	1.23578	30.9767
80	0.08121	0.19703	1.29170	32.8250
100	0.08121	0.21913	1.32899	33.8253

Progress of QCASM for Cohen's problem

Demand function: $\max(a_1 - b_1 p, a_2 - b_2 p)$

Results for UCI Repository (2900+ Products)

For 2900+ products, we obtained:

iteration	revenue			
	P-377	P-780	P-1060	total
1	257.92	1939.60	515.84	5857376.16
20	635.20	3709.92	1325.10	9987238.74
50	1264.01	6660.46	2673.87	16803284.78
70	1347.85	7053.87	2853.70	17766752.17
150	1347.85	7053.87	2853.70	17839027.50

Progress of QCASM for UCI's problem

Demand function: $\max(a - bp, 0)$

Comparison to Other Price Options

	revenue (in multiples of 10^6)			
data set	base prices	random prices	mid-selection	QCASM
Cohen	2.44	2.97	2.84	4.77
UCI	5.85	15.80	12.73	17.84
Logit	34.26	88.75	76.55	96.12

Comparison of different choices of prices

Paper with algorithm and results will be submitted this year

Summary and Outlook

- AD as differentiation of computer programmes
  - www.autodiff.org, (Griewank, Walther 2008), (Naumann 2012)
  - with working accuracy (Griewank, Kulshreshtha, Walther 2012)
  - reverse mode of AD known as backpropagation

Summary and Outlook

- AD as differentiation of computer programmes
  - www.autodiff.org, (Griewank, Walther 2008), (Naumann 2012)
  - with working accuracy (Griewank, Kulshreshtha, Walther 2012)
  - reverse mode of AD known as backpropagation

- optimization of PL functions also with PL constraints
  - algorithmic idea
  - convergence results
  - numerical results

  serves as work horse for nonsmooth optimization

Summary and Outlook

- AD as differentiation of computer programmes
  - www.autodiff.org, (Griewank, Walther 2008), (Naumann 2012)
  - with working accuracy (Griewank, Kulshreshtha, Walther 2012)
  - reverse mode of AD known as backpropagation
- optimization of PL functions also with PL constraints
  - algorithmic idea
  - convergence results
  - numerical results

  serves as work horse for nonsmooth optimization
- two data-driven applications from retail
  - determining the demand function
  - maximizing retail

Summary and Outlook

- AD as differentiation of computer programmes
  - www.autodiff.org, (Griewank, Walther 2008), (Naumann 2012)
  - with working accuracy (Griewank, Kulshreshtha, Walther 2012)
  - reverse mode of AD known as backpropagation

- optimization of PL functions also with PL constraints
  - algorithmic idea
  - convergence results
  - numerical results

  serves as work horse for nonsmooth optimization

- two data-driven applications from retail
  - determining the demand function
  - maximizing retail

  Future work: Take fairness into account!