CppAD's Abs-normal Representation

Bradley M. Bell

Applied Physics Laboratory and
Institute for Health Metrics and Evaluation,
University of Washington,
bradbell@uw.edu

July 2, 2018

Non-Smooth Functions

$f(x)$

$f : \mathbf{R}^n \to \mathbf{R}^m$ where the only non-smooth nodes in its computational graph are $|\cdot|$.

Non-Smooth Functions

$f(x)$

$f : \mathbf{R}^n \to \mathbf{R}^m$ where the only non-smooth nodes in its computational graph are $|\cdot|$.

$a(x)$

Let s be number of $|\cdot|$ in f. We define $a : \mathbf{R}^n \to \mathbf{R}^s$ where $a_i(x)$ is the result for the i-th absolute value.

Smooth Functions

$z(x, u)$

There is a smooth $z : \mathbf{R}^{n+s} \to \mathbf{R}^s$ where $z_i(x, u)$ is argument to i-th absolute value when $u_j = a_j(x)$ for $j < i$.

Smooth Functions

$z(x, u)$

There is a smooth $z : \mathbf{R}^{n+s} \to \mathbf{R}^s$ where $z_i(x, u)$ is argument to i-th absolute value when $u_j = a_j(x)$ for $j < i$.

$y(x, u)$

There is a smooth $y : \mathbf{R}^{n+s} \to \mathbf{R}^m$ where $y(x, u) = f(x)$ when $u = a(x)$ for all i.

Smooth Functions

$z(x, u)$

There is a smooth $z : \mathbf{R}^{n+s} \to \mathbf{R}^s$ where $z_i(x, u)$ is argument to i-th absolute value when $u_j = a_j(x)$ for $j < i$.

$y(x, u)$

There is a smooth $y : \mathbf{R}^{n+s} \to \mathbf{R}^m$ where $y(x, u) = f(x)$ when $u = a(x)$ for all i.

$g(x, u)$

The function $g : \mathbf{R}^{n+s} \to \mathbf{R}^{m+s}$ is defined by

$$g(x, u) = \left[\begin{array}{c} y(x, u) \\ z(x, y) \end{array} \right]$$

Approximating $a(x)$

$$z[\hat{x}](x,u) = z(\hat{x}, a(\hat{x})) + \partial_x z(\hat{x}, a(\hat{x}))(x - \hat{x}) + \partial_u z(\hat{x}, a(\hat{x}))(u - a(\hat{x}))$$

Approximating $a(x)$

$$z[\hat{x}](x,u) = z(\hat{x},a(\hat{x})) + \partial_x z(\hat{x},a(\hat{x}))(x-\hat{x}) + \partial_u z(\hat{x},a(\hat{x}))(u-a(\hat{x}))$$

Note that $z_0(x,u)$ does not depend on u:

$$a_0[\hat{x}](x) = |z_0(\hat{x},a(\hat{x})) + \partial_x z_0(\hat{x},a(\hat{x}))(x-\hat{x})|$$

Approximating $a(x)$

$$z[\hat{x}](x,u) = z(\hat{x}, a(\hat{x})) + \partial_x z(\hat{x}, a(\hat{x}))(x-\hat{x}) + \partial_u z(\hat{x}, a(\hat{x}))(u-a(\hat{x}))$$

Note that $z_0(x,u)$ does not depend on u:

$$a_0[\hat{x}](x) = |z_0(\hat{x}, a(\hat{x})) + \partial_x z_0(\hat{x}, a(\hat{x}))(x-\hat{x})|$$

$$\begin{aligned} a_i[\hat{x}](x) &= \Bigg| z_i(\hat{x}, a(\hat{x})) + \partial_x z_i(\hat{x}, a(\hat{x}))(x-\hat{x}) \\ &+ \sum_{j<i} \partial_{u(j)} z_i(\hat{x}, a(\hat{x}))(a_j[\hat{x}](x) - a_j(\hat{x})) \Bigg| \end{aligned}$$

Approximating $a(x)$

$$z[\hat{x}](x,u) = z(\hat{x},a(\hat{x})) + \partial_x z(\hat{x},a(\hat{x}))(x-\hat{x}) + \partial_u z(\hat{x},a(\hat{x}))(u-a(\hat{x}))$$

Note that $z_0(x,u)$ does not depend on u:

$$a_0[\hat{x}](x) = |z_0(\hat{x},a(\hat{x})) + \partial_x z_0(\hat{x},a(\hat{x}))(x-\hat{x})|$$

$$\begin{aligned} a_i[\hat{x}](x) &= \Bigg| z_i(\hat{x},a(\hat{x})) + \partial_x z_i(\hat{x},a(\hat{x}))(x-\hat{x}) \\ &+ \sum_{j<i} \partial_{u(j)} z_i(\hat{x},a(\hat{x}))(a_j[\hat{x}](x) - a_j(\hat{x})) \Bigg| \end{aligned}$$

$$a(x) = a[\hat{x}](x) + o(x-\hat{x})$$

Representation

`f.abs_normal_fun(g, a)`

Given the `ADFun<Base>` object `f` for $f(x)$, this creates the two `ADFun<Base>` objects `g`, `a` for $g(x, u)$ and $a(x)$ respectively.

Representation

`f.abs_normal_fun(g, a)`

Given the `ADFun<Base>` object `f` for $f(x)$, this creates the two `ADFun<Base>` objects `g`, `a` for $g(x, u)$ and $a(x)$ respectively.

Advantages

Any AD operation can be computed for the smooth function `g`; e.g., any order forward and reverse mode, sparsity patterns, and sparse derivatives.

Approximating $f(x)$

$$y[\hat{x}](x,u) = y(\hat{x},a(\hat{x})) + \partial_x y(\hat{x},a(\hat{x}))(x-\hat{x}) + \partial_u y(\hat{x},a(\hat{x}))(u-a(\hat{x}))$$

Approximating $f(x)$

$$y[\hat{x}](x,u) = y(\hat{x},a(\hat{x})) + \partial_x y(\hat{x},a(\hat{x}))(x-\hat{x}) + \partial_u y(\hat{x},a(\hat{x}))(u-a(\hat{x}))$$

$$f(x) = y[\hat{x}](x, a[\hat{x}](x)) + o(x-\hat{x})$$

```
abs_eval(n, m, s, g_hat , g_jac , delta_x)
```

Evaluates $y[\hat{x}](x, a[\hat{x}](x))$

Approximating $f(x)$

$$y[\hat{x}](x,u) = y(\hat{x},a(\hat{x})) + \partial_x y(\hat{x},a(\hat{x}))(x-\hat{x}) + \partial_u y(\hat{x},a(\hat{x}))(u-a(\hat{x}))$$

$$f(x) = y[\hat{x}](x, a[\hat{x}](x)) + o(x-\hat{x})$$

```
abs_eval(n, m, s, g_hat , g_jac , delta_x)
```

Evaluates $y[\hat{x}](x, a[\hat{x}](x))$

- `g_hat` is $g[\hat{x}, a(\hat{x})]$

Approximating $f(x)$

$$y[\hat{x}](x,u) = y(\hat{x},a(\hat{x})) + \partial_x y(\hat{x},a(\hat{x}))(x-\hat{x}) + \partial_u y(\hat{x},a(\hat{x}))(u-a(\hat{x}))$$

$$f(x) = y[\hat{x}](x,a[\hat{x}](x)) + o(x-\hat{x})$$

```
abs_eval(n, m, s, g_hat , g_jac , delta_x)
```

Evaluates $y[\hat{x}](x,a[\hat{x}](x))$

- `g_hat` is $g[\hat{x},a(\hat{x})]$
- `g_jac` is $g^{(1)}[\hat{x},a(\hat{x})]$

Approximating $f(x)$

$$y[\hat{x}](x,u) = y(\hat{x},a(\hat{x})) + \partial_x y(\hat{x},a(\hat{x}))(x-\hat{x}) + \partial_u y(\hat{x},a(\hat{x}))(u-a(\hat{x}))$$

$$f(x) = y[\hat{x}](x, a[\hat{x}](x)) + o(x-\hat{x})$$

`abs_eval(n, m, s, g_hat , g_jac , delta_x)`

Evaluates $y[\hat{x}](x, a[\hat{x}](x))$

- `g_hat` is $g[\hat{x}, a(\hat{x})]$
- `g_jac` is $g^{(1)}[\hat{x}, a(\hat{x})]$
- `delta_x` is $x - \hat{x}$

abs_min_linear

Problem

minimize $\tilde{f}(x) = y[\hat{x}](x, a(\hat{x}))$ w.r.t x subject to $-b \leq x \leq b$ using the assumption that $\tilde{f}(x)$ is convex.

`abs_min_linear`

Problem

minimize $\tilde{f}(x) = y[\hat{x}](x, a(\hat{x}))$ w.r.t x subject to $-b \leq x \leq b$ using the assumption that $\tilde{f}(x)$ is convex.

Algorithm

1. Start at with point $x = \hat{x}$ and C an empty set of cutting planes.

abs_min_linear

Problem

minimize $\tilde{f}(x) = y[\hat{x}](x, a(\hat{x}))$ w.r.t x subject to $-b \leq x \leq b$ using the assumption that $\tilde{f}(x)$ is convex.

Algorithm

1. Start at with point $x = \hat{x}$ and C an empty set of cutting planes.
2. Add affine apprimation for $\tilde{f}(x)$ at x to C.

abs_min_linear

Problem

minimize $\tilde{f}(x) = y[\hat{x}](x, a(\hat{x}))$ w.r.t x subject to $-b \leq x \leq b$ using the assumption that $\tilde{f}(x)$ is convex.

Algorithm

1. Start at with point $x = \hat{x}$ and C an empty set of cutting planes.
2. Add affine apprimation for $\tilde{f}(x)$ at x to C.
3. Minimize w.r.t x the maximum of the affine functions in C subject to $-b \leq x \leq b$ (this is an LP).

`abs_min_linear`

Problem

minimize $\tilde{f}(x) = y[\hat{x}](x, a(\hat{x}))$ w.r.t x subject to $-b \leq x \leq b$ using the assumption that $\tilde{f}(x)$ is convex.

Algorithm

1. Start at with point $x = \hat{x}$ and C an empty set of cutting planes.
2. Add affine apprimation for $\tilde{f}(x)$ at x to C.
3. Minimize w.r.t x the maximum of the affine functions in C subject to $-b \leq x \leq b$ (this is an LP).
4. If change in x for this this iteration is small, return x as solution. Otherwise, goto step 2.