Introduction to Algorithmic Differentiation

AD by Hand (Case Study)

Uwe Naumann

Informatik 12:
Software and Tools for Computational Engineering (STCE)

RWTH Aachen

Contents

Outline

Recall

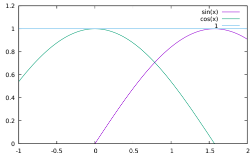
- Sigmoid
- Newton

Newton on Sigmoid

- Primal
- Tangent
- Adjoint

Let $f : \boldsymbol{R}^2 \rightarrow \boldsymbol{R}$ be defined as

$$f(x,p) = \begin{cases} f_1(x) & x < p \\ f_2(x) & x \geq p\ . \end{cases}$$



with differentiable univariate scalar f_1 and f_2.

Depending on the choice of f_1 and f_2 the function f can be nondifferentiable or even discontinuous at $x = p$.

Examples:

- $f_1 = \cos,\ f_2 = \sin \Rightarrow$ discontinuous at $x = p = 1$
- $f_1 = \cos,\ f_2 = \sin \Rightarrow$ nondifferentiable at $x = p = \frac{\pi}{4}$
- $f_1 = 1,\ f_2 = \cos \Rightarrow$ differentiable at $x = p = 0$

Sigmoid

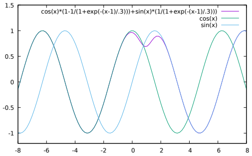
Definition

Sigmoidal smoothing replaces f with $\tilde{f} : \mathbf{R}^3 \rightarrow \mathbf{R}$ defined as

$$\tilde{f}(x, p, w) = (1 - \sigma(x, p, w)) \cdot f_1(x) + \sigma(x, p, w) \cdot f_2(x) \; ,$$

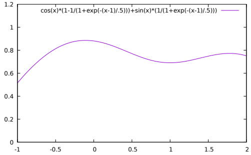
where

$$\sigma(x, p, w) = \frac{1}{1 + e^{-\frac{x-p}{w}}} \; .$$

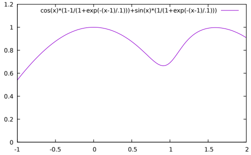


$w = 0.3$

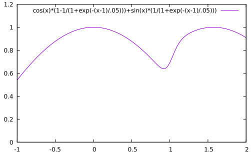
Example: $f_1 = \cos$, $f_2 = \sin$ at $x = p = 1$



$w = 0.5$



$w = 0.1$



$w = 0.05$

Sigmoid

Implementation

```
template<typename T>
void f1(const T &x, T &y);

template<typename T>
void f2(const T &x, T &y);

template<typename T, typename PT>
void sigmoid(T &x, const PT &p, const PT &w) {
  T a; f1(x,a);
  T b; f2(x,b);
  x=1/(1+exp(-(x-p)/w));
  x=a*(1-x)+b*x;
}
```

Consider a nonlinear equation $y = f(x) = 0$ at some (starting) point x.

Building on the assumption that $f(x + \Delta x) \approx f(x) + f'(x) \cdot \Delta x$ the root finding problem for f can be replaced locally by the root finding problem for the linearization

$$\bar{f}(\Delta x) = f(x) + f'(x) \cdot \Delta x \; .$$

The right-hand side is a straight line intersecting the y-axis in $(\Delta x = 0, \bar{f}(\Delta x) = f(x))$.

Solution of

$$\bar{f}(\Delta x) = f(x) + f'(x) \cdot \Delta x = 0$$

for Δx yields

$$\Delta x = -\frac{f(x)}{f'(x)}$$

implying $f(x + \Delta x) \approx 0$.

If the new iterate is not close enough to the root of the nonlinear function, i.e, $|f(x+\Delta x)| > \epsilon$ for some measure of accuracy of the numerical approximation $\epsilon > 0$, then it becomes the starting point for the next iteration yielding the recurrence

$$x = x - \frac{f(x)}{f'(x)}$$

Convergence of this method is not guaranteed in general. Damping of the magnitude of the next step may help.

$$x = x - \alpha \cdot \frac{f(x)}{f'(x)} \quad \text{for } 0 < \alpha \leq 1 \, .$$

The damping parameter α is often determined by line search (e.g, recursive bisection yielding $\alpha = 1, \ 0.5, \ 0.25, \ \ldots$) such that decrease in absolute function value is ensured.

Newton's Method

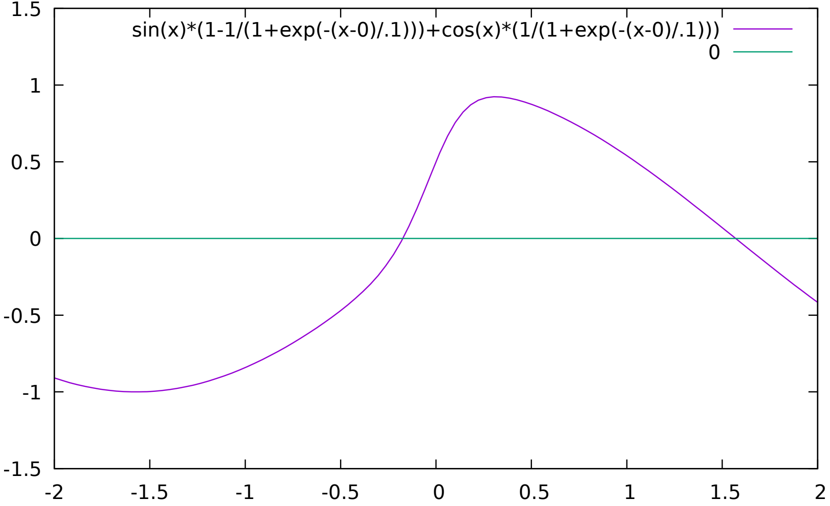
Implementation

```
template<typename T, typename PT>
T f(T &x, const PT &p);

template<typename T, typename PT>
T dfdx(T &x, const PT &p);

template<typename T, typename PT>
void newton(T &x, const T &p, const PT &eps, const unsigned int maxit) {
  unsigned int it=0;
  T y=f(x,p);
  do {
    x-=y/dfdx(x,p);
    y=f(x,p);
    if (++it==maxit) break;
  } while(fabs(y)>eps);
}
```

Outline



$f_1(x) = \sin(x);\ f_2(x) = \cos(x);\ p = 0;\ w = 0.1$

Newton on Sigmoid

Primal

```
template<typename T, typename PT>
void newton(T &x, const T &p, const T &w, const PT &eps, const unsigned int maxit) {
  unsigned int it=0;
  T y=x, dy;
  dsigmoid_dx(y,p,w,dy);
  do {
    x-=y/dy;
    y=x;
    dsigmoid_dx(y,p,w,dy);
    if (++it==maxit) break;
  } while(fabs(y)>eps);
}
```

Newton on Sigmoid

Primal (Derivative of Residual)


```
template<typename T, typename PT>
void sigmoid_t(T &x, T &x_t, const PT &p, const PT &w) {
  T a, a_t;
  f1_t(x,x_t,a,a_t);
  T b, b_t;
  f2_t(x,x_t,b,b_t);
  T c_t=-exp(-(x-p)/w)/w*x_t;
  T c=1+exp(-(x-p)/w);
  x_t=-c_t/pow(c,2);
  x=1/c;
  x_t=(1-x)*a_t+(b-a)*x_t+x*b_t;
  x=a*(1-x)+b*x;
}

template<typename T, typename PT>
void dsigmoid_dx(T &x, const PT &p, const PT &w, T &dx) {
  dx=1; sigmoid_t(x,dx,p,w);
}
```

Newton on Sigmoid

Primal (f_1 and f_2)

```
template<typename T>
void f1_t(const T &x, const T &x_t, T &y, T &y_t) {
  y_t=cos(x)*x_t;
  y=sin(x);
}

template<typename T>
void f2_t(const T &x, const T &x_t, T &y, T &y_t) {
  y_t=-sin(x)*x_t;
  y=cos(x);
}
```

Code Inspection / Discussion / Experiments

Code Inspection / Discussion / Experiments

Summary

Recall

- Sigmoid
- Newton

Newton on Sigmoid

- Primal
- Tangent
- Adjoint