common lisp で自動微分アルゴリズム

Yuichiro Honda

2016/09/16

問題意識

計算機上で微分（勾配ベクトル）を速く，正確に計算したい
(性質のいい) 関数が与えられたとき，その微分も一意的に定まるはずだが，現状多くの場合微分の関数も手書きで別に与えなければならない
微分の用途：

- 最適化アルゴリズム
- 感度分析
- 物理モデリング
- 機械学習 etc...

微分は極限を扱うが，計算機には「無限」が理解できない．
→ 近似の必要性

定式化

入力：関数 f，点 x
出力：入力関数の点 x での微分値

近似

よく見る微分の姿：

- 数値微分:

$$\frac{df(x)}{dx} = \lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$$

- 数式微分:

$$\frac{d(u(x)+v(x))}{dx} = \frac{du(x)}{dx} + \frac{dv(x)}{dx}$$

$$\frac{du(x)v(x)}{dx} = u(x)\frac{dv(x)}{dx} + \frac{du(x)}{dx}v(x)$$

近似

数値微分の難点：

- 一般的な近似: $\frac{df(x)}{dx} = \frac{f(x+h)-f(x)}{h}$ (h は適当に小さい値)
- 差分法を用いることによって生じる離散化誤差

数式微分の難点：

- $\frac{d(make-point(x,y))}{dx}$ など，別の場所で定義された関数への操作が難しい.
- $\frac{d(u_1(x)u_2(x)...u_{1000}(x))}{dx}$ とかだと全部バラす前にスタックオーバーフローしないか心配...

自動微分

自動微分の原理

$$f(x+\epsilon) = f(x) + f'(x)\epsilon$$

f: 点 x で微分可能な任意の関数

ϵ: 二重数の複零元

二重数

二重数：複零元 (二乗すると 0 になる数)ϵ を用いた実数の拡張
任意の二重数は $a + b\epsilon$ の形で表せる

二重数の基本演算

$$
\begin{aligned}
\text{加算：} & (a + b\epsilon) + (c + d\epsilon) = (a + c) + (b + d)\epsilon \\
\text{乗算：} & (a + b\epsilon)(c + d\epsilon) = ac + (ad + bc)\epsilon \\
\text{除算：} & \frac{a + b\epsilon}{c + d\epsilon} = \frac{(a + b\epsilon)(c - d\epsilon)}{(c + d\epsilon)(c - d\epsilon)} = \frac{ac + (bc - ad)\epsilon}{c^2}
\end{aligned}
$$

二重数

二重数と関数

$$
\begin{aligned}
f(a+b\epsilon) &= \sum_{n=0}^{\infty} \frac{f^{(n)}(a)(b\epsilon)^n}{n!} \qquad \text{（テイラー展開）} \\
&= f(a) + \frac{f'(a)b\epsilon}{1!} + \frac{f''(a)(b\epsilon)^2}{2!} + \ldots \\
&= f(a) + bf'(a)\epsilon
\end{aligned}
$$

二重数

二重数の基本演算 (初等関数)

$$e^{a+b\epsilon} = e^a \sum_{n=0}^{\infty} \frac{(b\epsilon)^n}{n!} = e^a(1 + b\epsilon)$$

$$\sin(a + b\epsilon) = \sum_{n=0}^{\infty} \frac{\sin^{(n)} a(b\epsilon)^n}{n!} = \sin a + b\epsilon \cos a$$

$$\cos(a + b\epsilon) = \sum_{n=0}^{\infty} \frac{\sin^{(n)} a(b\epsilon)^n}{n!} = \cos a - b\epsilon \sin a$$

実装

実装の手順

1. 二重数 (dual-number) を構造体によって定義
2. number に適用できる標準的なメソッドを generic に再定義し，dual-number に適用できるようにする
3. 入力関数 f，点 a に対して $f(a+\epsilon)$ を計算
4. 3. で計算した ϵ の係数を取り出す

できたこと

- 加減乗除，初等関数 ($sinx$，$cosx$，e^x，$\sqrt{x}$) の組み合わせで表される関数の微分を実現した
- lisp で package の作り方を学習した
- 二重数の代数的理解が深まった

できなかったこと

- 数多あるメソッドの再定義 (大小比較，log，mod とかは入れたかった)
- 零除算などの例外処理
- マクロをうまくつかう
- 高速化

感想

common lisp には演算子オーバーロードというか演算子が無いのでメソッドを generic に再定義する手法を取ったが，再定義したメソッドに mapcar などをかけられないという制約が痛かった．マクロなどをうまく使えば解決できるかもしれないし，そもそももっといい方法があるかもしれない．