

Simple essence of AD

Artur Ezequiel Nelson

Universidade do Minho

24 de Maio

Index

Relembrar

Conceitos base

Objetivo

Introdução breve a categorias e funtores

Adaptação de definições

Dedução de instâncias em categorias

Instâncias deduzidas

Algoritmo AD generalizado

RAD e FAD generalizados

Scaling up

Bibliografia

Relembrar

Documento estudado: "The simple essence of automatic differentiation"

Objetivo: criar uma formalização genérica do conceito de ML para aprendizagem supervisionada

Definição de \mathcal{D} e conversão em \mathcal{D}^+

\mathcal{D} - aproximação linear de uma função

Definition

Let $f :: a \rightarrow b$ be a function, where a and b are vectorial spaces that share a common underlying field. The first derivative definition is the following:

$$\mathcal{D} :: (a \rightarrow b) \rightarrow (a \rightarrow (a \multimap b))$$

\mathcal{D}^+ - versão de \mathcal{D} com composição eficiente

$$\mathcal{D}^+ :: (a \rightarrow b) \rightarrow (a \rightarrow (b \times (a \multimap b)))$$

$$\mathcal{D}^+ f a = (f a, \mathcal{D} f a)$$

Corolários associados a \mathcal{D}^+

Corolário 1.1

$$\mathcal{D}^+ (g \circ f) a = \mathbf{let} \{ (b, f') = \mathcal{D}^+ f a; (c, g') = \mathcal{D}^+ g b \}$$

$$\mathbf{in} (c, g' \circ f')$$

Corolário 2.1

$$\mathcal{D}^+ (f \times g) (a, b) = \mathbf{let} \{ (c, f') = \mathcal{D}^+ f \ a; (d, g') = \mathcal{D}^+ g \ b \}$$

$$\mathbf{in} ((c, d), f' \times g')$$

Corolário 3.1

Para todas as funções lineares f , $\mathcal{D}^+ f = \lambda a \rightarrow (fa, f)$.

Objetivo

Criar uma implementação de \mathcal{D}^+ através da transcrição dos seus corolários para teoria de categorias de modo a obter um algoritmo generalizado para AD.

Categorias e funtores

Categoria: conjunto de objetos e morfismos com duas operações base(id e composição) e 2 regras:

- $id \circ f = f \circ id = f$
- $f \circ (g \circ h) = (f \circ g) \circ h$

Functor: mapeia uma categoria noutra, preservando a estrutura

- Dado um objeto $t \in \mathcal{U}$ existe um objeto correspondente $F t \in \mathcal{V}$
- Dado um morfismo $m :: a \rightarrow b \in \mathcal{U}$ existe um morfismo correspondente $F m :: F a \rightarrow F b \in \mathcal{V}$
- $F \text{id} (\in \mathcal{U}) = \text{id} (\in \mathcal{V})$
- $F (f \circ g) = F f \circ F g$

Adaptação de definições

Definição de tipo \mathcal{D}

newtype $\mathcal{D} \ a \ b = \mathcal{D} \ (a \rightarrow b \times (a \multimap b))$

Definição de $\hat{\mathcal{D}}$

$$\hat{\mathcal{D}} :: (a \rightarrow b) \rightarrow \mathcal{D} \ a \ b$$

$$\hat{\mathcal{D}} f = \mathcal{D} (\mathcal{D}^+ f)$$

Definição de $\hat{\mathcal{D}}$ para funções lineares

$$linearD :: (a \rightarrow b) \rightarrow \mathcal{D} \ a \ b$$

$$\text{linearD } f = \mathcal{D} (\lambda a \rightarrow (f \ a, f))$$

Passos para obter a instância a partir da definição do funtor

- Passo 1 - Assumir que $\hat{\mathcal{D}}$ é funtor de uma instância de \mathcal{D} a determinar
- Passo 2 - Substituir pelo que determinamos nos corolários
- Passo 3 - Generalizar condições se necessário para obtermos instância

Este processo é o mesmo para vários tipos de categorias.

Exemplo para categorias

Passo 1

$$id = \hat{\mathcal{D}} id = \mathcal{D} (\mathcal{D}^+ id)$$

$$\hat{\mathcal{D}} g \circ \hat{\mathcal{D}} f = \hat{\mathcal{D}} (g \circ f) = \mathcal{D} (\hat{\mathcal{D}} (g \circ f))$$

Passo 2

$$id = \mathcal{D} (\lambda a \rightarrow (id\ a, id))$$

$$\hat{\mathcal{D}} g \circ \hat{\mathcal{D}} f = \mathcal{D} (\lambda a \rightarrow \mathbf{let}\ \{(b, f') = \mathcal{D}^+ f\ a; (c, g') = \mathcal{D}^+ g\ b\} \mathbf{in}\ (c, g' \circ f'))$$

Passo 3

Para a nossa instância a primeira equação que determinamos serve como definição da identidade.

Para definir a composição generalizamos a condição:

$$\mathcal{D} g \circ \mathcal{D} f = \mathcal{D} (\lambda a \rightarrow \mathbf{let}\ \{(b, f') = f\ a; (c, g') = g\ b\} \mathbf{in}\ (c, g' \circ f'))$$

Definição da classe de categoria monoidal

```
class Category  $k \Rightarrow$  Monoidal  $k$  where
     $(\times) :: (a \text{ ' } k \text{ ' } c) \rightarrow (b \text{ ' } k \text{ ' } d) \rightarrow ((a \times b) \text{ ' } k \text{ ' } (c \times d))$ 
```

Instância deduzida para a categoria monoidal

```
instance Monoidal  $\mathcal{D}$  where
     $\mathcal{D} \ f \times \mathcal{D} \ g = \mathcal{D} \ (\lambda(a,b) \rightarrow \text{let } \{(c,f') = f \ a; (d,g') = g \ b\}$ 
    in  $((c,d), f' \times g'))$ 
```

Definição da classe de categoria cartesiana

```

class Monoidal  $k \Rightarrow$  Cartesian  $k$  where
    exl ::  $(a, b) \rightarrow k \rightarrow a$ 
    exr ::  $(a, b) \rightarrow k \rightarrow b$ 
    dup ::  $a \rightarrow k \rightarrow (a, a)$ 
    
```

Instância deduzida para a categoria cartesiana

```

instance Cartesian  $D$  where
    exl = linearD exl
    exr = linearD exr
    dup = linearD dup
    
```

Definição da classe de categoria cocartesiana

class *Category* $k \Rightarrow \text{Cocartesian } k$ **where**

inl :: $a \text{ ' } k \text{ ' } (a, b)$

inr :: $b \text{ ' } k \text{ ' } (a, b)$

jam :: $(a, a) \text{ ' } k \text{ ' } a$

Instâncias que deduzimos - caso (\rightarrow^+)

newtype $a \rightarrow^+ b = \text{AddFun } (a \rightarrow b)$

instance *Category* (\rightarrow^+) **where**

type *Obj* $(\rightarrow^+) = \text{Additive}$

id = *AddFun id*

AddFun g \circ *AddFun f* = *AddFun* (*g* \circ *f*)

instance *Monoidal* (\rightarrow^+) **where**

AddFun f \times *AddFun g* = *AddFun* (*f* \times *g*)

instance *Cartesian* (\rightarrow^+) **where**

exl = *AddFun exl*

exr = *AddFun exr*

dup = *AddFun dup*

Instância deduzida para RAD genérico

instance *Category* $k \Rightarrow \text{Category } \text{Cont}_k^r$ **where**

id = *Cont id*

Cont g \circ *Cont f* = *Cont (f \circ *g)**

instance *Monoidal* $k \Rightarrow \text{Monoidal } \text{Cont}_k^r$ **where**

Conf f \times *Cont g* = *Cont (join* \circ (*f* \times *g)* \circ *unjoin)*

instance *Cartesian* $k \Rightarrow \text{Cartesian } \text{Cont}_k^r$ **where**

exl = *Cont (join* \circ *inl)*; *exr* = *Cont (join* \circ *inr)*

dup = *Cont (jam* \circ *unjoin)*

instance *Cocartesian* $k \Rightarrow \text{Cocartesian } \text{Cont}_k^r$ **where**

inl = *Cont (exl* \circ *unjoin)*; *inr* = *Cont (exr* \circ *unjoin)*

jam = *Cont (join* \circ *dup)*

instance *Scalable* k *a* $\Rightarrow \text{Scalable } \text{Cont}_k^r$ *a* **where**

scale s = *Cont (scale s)*

...

■ ■ ■