...Machine Learning...

Artur Ezequiel Nelson

Universidade do Minho

26 de Abril

Indice

A short introdution

- We want to calculate $\mathcal{D}$.
- However, $\mathcal{D}$ is not computable.
- Solution: reimplement corollaries using category theory

A short introdution

- We want to calculate $\mathcal{D}$.
- However, $\mathcal{D}$ is not computable.
- Solution: reimplement corollaries using category theory

A short introdution

- We want to calculate $\mathcal{D}$.
- However, $\mathcal{D}$ is not computable.
- Solution: reimplement corollaries using category theory

A short introdution

Corollary 1.1

NOTA: adicionar definição do corolário 1.1 aqui

Corollary 2.1

NOTA: adicionar definição do corolário 2.1 aqui

Corollary 3.1

NOTA: adicionar definição do corolário 3.1 aqui

Categories

A category is a collection of objects(sets and types) and morphisms(operation between objects), with 2 basic operations(identity and composition) of morfisms, and 2 laws:

- (C.1) $id \circ f = id \circ f = f$
- (C.2) $f \circ (g \circ h) = (f \circ g) \circ h$

Note

For this paper, objects are data types and morfisms are functions

class *Category k* **where**
$id :: (a'k'a)$
$(\circ) :: (b'k'c) \rightarrow (a'k'b) \rightarrow (a'k'c)$

instance *Category* $(\rightarrow)$ **where**
$id = \lambda a \rightarrow a$
$g \circ f = \lambda a \rightarrow g\ (f\ a)$

Functors

A functor F between 2 categories $\mathcal{U}$ *and* $\mathcal{V}$ is such that:

- given any object t $\in \mathcal{U}$ there exists an object F t $\in \mathcal{V}$
- given any morphism m :: a $\rightarrow$ b $\in \mathcal{U}$ there exists a morphism F m :: F a $\rightarrow$ F b $\in \mathcal{V}$
- F id ($\in \mathcal{U}$) = id ($\in \mathcal{V}$)
- F (f $\circ$ g) = F f $\circ$ F g

Note

Given this papers category properties(objects are data types) functors map types to themselves

Objective

Let's start by defining a new data type:
newtype $\mathcal{D}$ a b = $\mathcal{D}$ (a $\rightarrow$ b $\times$ (a $\multimap$ b))
,and adapting $\mathcal{D}^{+}$ to use it:

Adapted definition

$$\hat{\mathcal{D}} :: (a \rightarrow b) \rightarrow \mathcal{D}\ a\ b$$
$$\hat{\mathcal{D}}\ f = \mathcal{D}\ (\mathcal{D}^{+}\ f)$$

Our objective is to deduce an instance of a Category for $\mathcal{D}$ where $\hat{\mathcal{D}}$ is a functor.

Instance deduction

Before deducing our instance we must first note that using corollaries 3.1 and 1.1 we can determine that

- (DP.1) $\mathcal{D}^+$ id = λ a -> (id a,id)
- (DP.2) $\mathcal{D}^+$(g $\circ$ f) = λ a -> let(b,f') = $\mathcal{D}^+$ f a; (c,g') = $\mathcal{D}^+$ g b in (c,g' $\circ$ f')

Saying that $\hat{\mathcal{D}}$ is a functor is equivalent to, for all f and g functions of apropriate types:

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

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

Instance deduction

Based on (DP.1) and (DP.2) we'll rewrite the above into the following definition:
id = $\mathcal{D}$ (λ a -> (id a,id))
$\hat{\mathcal{D}}$ g $\circ$ $\hat{\mathcal{D}}$ f = $\mathcal{D}$ (λ a -> let(b,f') = $\mathcal{D}^+$ f a; (c,g') = $\mathcal{D}^+$ g b in (c,g' $\circ$ f'))
The first equation shown above has a trivial solution(define id of instance as $\mathcal{D}$(λ a -> (id a,id)))
To solve the second we'll first solve a more general one:
$\mathcal{D}$ g $\circ$ $\mathcal{D}$ f = $\mathcal{D}$(λ a -> let(b,f') = f a; (c,g') = g b in(c,g' $\circ$ f'))
This condition also leads us to a trivial solution inside our instance.

Instance deduction

$\hat{\mathcal{D}}$ definition for linear functions

$$\mathit{linearD} :: (a \to b) \to \mathcal{D}\ a\ b$$
$$\mathit{linearD}\ f = \mathcal{D}\ (\lambda a \to (f\ a, f))$$

Categorical instance we've deduced

$$\textbf{instance}\ \mathit{Category}\ \mathcal{D}\ \textbf{where}$$
$$\quad \mathit{id} = \mathit{linearD}\ \mathit{id}$$
$$\quad \mathcal{D}\ g \circ \mathcal{D}\ f =$$
$$\quad \mathcal{D}\ (\lambda a \to \textbf{let}\ \{(b, f') = f\ a; (c, g') = g\ b\}\ \textbf{in}\ (c, g' \circ f'))$$

Instance proof

In order to prove that the instance is correct we must check if it follows laws (C.1) and (C.2).
First we must make a concession: that we only use morfisms arising from $\mathcal{D}^+$(we can force this by transforming $\mathcal{D}$ into an abstract type). If we do, then $\mathcal{D}^+$ is a functor.

(C.1) proof

id $\circ$ $\hat{\mathcal{D}}$
= $\hat{\mathcal{D}}$ id $\circ$ $\hat{\mathcal{D}}$ f - functor law for id (specification of $\hat{\mathcal{D}}$)
= $\hat{\mathcal{D}}$ (id $\circ$ f) - functor law for ($\circ$)
= $\hat{\mathcal{D}}$ f - categorical law

Instance proof

(C.2) proof

$\hat{\mathcal{D}}$ h ∘ ($\hat{\mathcal{D}}$ g ∘ $\hat{\mathcal{D}}$ f)
= $\hat{\mathcal{D}}$ h ∘ $\hat{\mathcal{D}}$ (g ∘ f) - functor law for (∘)
= $\hat{\mathcal{D}}$ (h ∘ (g ∘ f)) - functor law for (∘)
= $\hat{\mathcal{D}}$ ((h ∘ g) ∘ f) - categorical law
= $\hat{\mathcal{D}}$ (h ∘ g) ∘ $\hat{\mathcal{D}}$ f - functor law for (∘)
= ($\hat{\mathcal{D}}$ h ∘ $\hat{\mathcal{D}}$ g) ∘ $\hat{\mathcal{D}}$ f - functor law for (∘)

Note

This proofs don't require anything from $\mathcal{D}$ *and* $\hat{\mathcal{D}}$ aside from functor laws. As such, all other instances of categories created from a functor won't require further proving like this onr did.

Monoidal categories and functors

Generalized parallel composition shall be defined using a monoidal category:

class *Category k* $\Rightarrow$ *Monoidal k* **where**
$(x) :: (a\ '\mathtt{k}'\ c) \rightarrow (b\ '\mathtt{k}'\ d) \rightarrow ((a \; x \; b)\ '\mathtt{k}'\ (c \; x \; d))$

instance *Monoidal* $(\rightarrow)$ **where**
$f \; x \; g = \lambda(a, b) \rightarrow (f \; a, g \; b)$

Monoidal Functor definition

A monoidal functor F between categories $\mathcal{U}$ *and* $\mathcal{V}$ is such that:

- F is a functor
- F (f $\times$ g) = F f $\times$ F g

Instance deduction

From corollary 2.1 we can deduce that:
$\mathcal{D}^+$ (f $\times$ g) = λ (a,b) -> let(c,f')=$\mathcal{D}^+$ f a;(d,g')= $\mathcal{D}^+$ g b
in ((c,d),f' $\times$ g')
Deriving F from $\hat{\mathcal{D}}$ leaves us with the following definition:
$\mathcal{D}$ ($\mathcal{D}^+$ f) $\times$ $\mathcal{D}$ ($\mathcal{D}^+$ g) = $\mathcal{D}$ ($\mathcal{D}^+$ (f $\times$ g))
Using the same method as before, we replace $\mathcal{D}^+$ with it's definition and generalize the condition:
$\mathcal{D}$ f $\times$ $\mathcal{D}$ g =
$\mathcal{D}$ (λ (a,b) -> let(c,f') = f a; (d,g') = g b in ((c,d),f' $\times$ g'))
and this is enough for our new instance.

Instance deduction

Categorical instance we've deduced

$$
\begin{array}{l}
\textbf{instance}\ \mathit{Monoidal}\ \mathcal{D}\ \textbf{where} \\
\quad \mathcal{D}\ f\ x\ \mathcal{D}\ g = \mathcal{D}\ (\lambda(a,b) \rightarrow \textbf{let}\ \{(c,f') = f\ a; (d,g') = g\ b\} \\
\qquad \textbf{in}\ ((c,d), f'\ x\ g'))
\end{array}
$$

Cartesian categories and functors

class *Monoidal* $k \Rightarrow$ *Cartesian* k **where**
$exl :: (a, b)\ \texttt{'k'}\ a$
$exr :: (a, b)\ \texttt{'k'}\ b$
$dup :: a' k'\ (a, a)$

instance *Cartesian* $(\rightarrow)$ **where**
$exl = \lambda(a, b) \rightarrow a$
$exr = \lambda(a, b) \rightarrow b$
$dup = \lambda a \rightarrow (a, a)$

A cartesian functor F between categories $\mathcal{U}$ *and* $\mathcal{V}$ is such that:

- F is a monoidal functor
- F exl = exl
- F exp = exp
- F dup = dup

Instance deduction

From corollary 3.1 and from exl,exr and dup being linear functions we can deduce that:

$\mathcal{D}^+$ exl = $\lambda p \rightarrow$ (exl p,exl)

$\mathcal{D}^+$ exr = $\lambda p \rightarrow$ (exr p,exr)

$\mathcal{D}^+$ dup = $\lambda p \rightarrow$ (dup a,dup)

With this in mind we can arrive at our instance:

exl = $\mathcal{D}(\mathcal{D}^+$ exl)

exr = $\mathcal{D}(\mathcal{D}^+$ exr)

dup = $\mathcal{D}(\mathcal{D}^+$ dup)

Instance deduction

Replacing $\mathcal{D}^+$ with it's definition and remembering linearD's definition we can obtain:

exl = linearD exl

exr = linearD exr

dup = linearD dup

and convert this directly into a new instance:

Categorical instance we've deduced

$$
\begin{aligned}
&\textbf{instance } \textit{Cartesian D } \textbf{where}\\
&\quad \textit{exl} = \textit{linearD exl}\\
&\quad \textit{exr} = \textit{linearD exr}\\
&\quad \textit{dup} = \textit{linearD dup}
\end{aligned}
$$

Cocartesian category

This type of categories is the dual of the cartesian type of categories.

Note

In this paper coproducts are categorical products, i.e., biproducts

Definition

$$
\begin{array}{l}
\textbf{class } \mathit{Category}\ k \Rightarrow \mathit{Cocartesian}\ k \textbf{ where}: \\
\quad \mathit{inl} :: a' k' \ (a, b) \\
\quad \mathit{inr} :: b' k' \ (a, b) \\
\quad \mathit{jam} :: (a, a)\ '\mathtt{k}'\ \ a
\end{array}
$$

Cocartesian functors

Cocartesian functor definition

A cocartesian functor F between categories $\mathcal{U}$ *and* $\mathcal{V}$ is such that:

- F is a functor
- F inl = inl
- F inr = inr
- F jam = jam