Simple essence of AD

Artur Ezequiel Nelson

Universidade do Minho

24 de Maio

Index

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

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

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

$$\mathcal{D}^+ :: (a \to b) \to (a \to (b \times (a \multimap b)))$$
$$\mathcal{D}^+ \; f \; a = (f \; a, \mathcal{D} \; f \; a)$$

Corolários associados a $\mathcal{D}^+$

Corollary 1.1

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

Corollary 2.1

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

Corollary 3.1

For all linear functions f, $\mathcal{D}^+\ f = \lambda a \to (fa,\ f)$.

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:

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

Funtor: 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 id ($\in \mathcal{U}$) = id ($\in \mathcal{V}$)
- F (f $\circ$ g) = F f $\circ$ F g

Adaptação de definições

Definição de $\mathcal{D}$

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

Adapted definition for $\mathcal{D}$ type

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

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

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

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

Passo 3

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

Para definir a composição generalziamos a condição:

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

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

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

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

Instancia deduzida para a categoria

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

$$
\begin{array}{l}
\textbf{class } \textit{Category } k \Rightarrow \textit{Monoidal } k \textbf{ where} \\
\quad (\times) :: (a \;{}'\mathtt{k}{}'\; c) \to (b \;{}'\mathtt{k}{}'\; d) \to ((a \times b) \;{}'\mathtt{k}{}'\; (c \times d))
\end{array}
$$

Instancia deduzida para a categoria monoidal

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

$$
\begin{array}{l}
\textbf{class } \mathit{Monoidal}\ k \Rightarrow \mathit{Cartesian}\ k \textbf{ where} \\
\quad \mathit{exl} :: (a, b)\ \texttt{'k'}\ a \\
\quad \mathit{exr} :: (a, b)\ \texttt{'k'}\ b \\
\quad \mathit{dup} :: a\ \texttt{'k'}\ (a, a)
\end{array}
$$

Instancia deduzida para a categoria cartesiana

$$
\begin{array}{l}
\textbf{instance } \mathit{Cartesian}\ D \textbf{ where} \\
\quad \mathit{exl} = \mathit{linearD}\ \mathit{exl} \\
\quad \mathit{exr} = \mathit{linearD}\ \mathit{exr} \\
\quad \mathit{dup} = \mathit{linearD}\ \mathit{dup}
\end{array}
$$

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

Instâncias que deduzimos - caso $(\rightarrow^{+})$

$$
\begin{array}{l}
\textbf{newtype}\ a \rightarrow^{+} b = \mathit{AddFun}\ (a \rightarrow b) \\
\textbf{instance}\ \mathit{Category}\ (\rightarrow^{+})\ \textbf{where} \\
\quad \textbf{type}\ \mathit{Obj}\ (\rightarrow^{+}) = \mathit{Additive} \\
\quad \mathit{id} = \mathit{AddFun}\ \mathit{id} \\
\quad \mathit{AddFun}\ g \circ \mathit{AddFun}\ f = \mathit{AddFun}\ (g \circ f) \\
\textbf{instance}\ \mathit{Monoidal}\ (\rightarrow^{+})\ \textbf{where} \\
\quad \mathit{AddFun}\ f \times \mathit{AddFun}\ g = \mathit{AddFun}\ (f \times g) \\
\textbf{instance}\ \mathit{Cartesian}\ (\rightarrow^{+})\ \textbf{where} \\
\quad \mathit{exl} = \mathit{AddFun}\ \mathit{exl} \\
\quad \mathit{exr} = \mathit{AddFun}\ \mathit{exr} \\
\quad \mathit{dup} = \mathit{AddFun}\ \mathit{dup}
\end{array}
$$

Instâncias que deduzimos - caso $(\rightarrow^{+})$

$$
\begin{aligned}
&\textbf{instance}\ \mathit{Cocartesian}\ (\rightarrow^{+})\ \textbf{where} \\
&\quad \mathit{inl} = \mathit{AddFun}\ \mathit{inlF} \\
&\quad \mathit{inr} = \mathit{AddFun}\ \mathit{inrF} \\
&\quad \mathit{jam} = \mathit{AddFun}\ \mathit{jamF}
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{inlF} :: \mathit{Additive}\ b \Rightarrow a \rightarrow a \times b \\
&\mathit{inrF} :: \mathit{Additive}\ a \Rightarrow b \rightarrow a \times b \\
&\mathit{jamF} :: \mathit{Additive}\ a \Rightarrow a \times a \rightarrow a
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{inlF} = \lambda a \rightarrow (a, 0) \\
&\mathit{inrF} = \lambda b \rightarrow (0, b) \\
&\mathit{jamF} = \lambda (a, b) \rightarrow a + b
\end{aligned}
$$

Instancia deduzida para AD genérico

$$\textbf{newtype}\ D_k\ a\ b = D\ (a \to b \times (a\ `k`\ b))$$

$$linearD :: (a \to b) \to (a\ `k`\ b) \to D_k\ a\ b$$
$$linearD\ f\ f' = D\ (\lambda a \to (f\ a, f'))$$

$$\textbf{instance}\ Category\ k \Rightarrow Category\ D_k\ \textbf{where}$$
$$\quad \textbf{type}\ Obj\ D_k = Additive \wedge Obj\ k\ ...$$

$$\textbf{instance}\ Monoidal\ k \Rightarrow Monoidal\ D_k\ \textbf{where}\ ...$$

$$\textbf{instance}\ Cartesian\ k \Rightarrow Cartesian\ D_k\ \textbf{where}\ ...$$

$$\textbf{instance}\ Cocartesian\ k \Rightarrow Cocartesian\ D_k\ \textbf{where}$$
$$\quad inl = linearD\ inlF\ inl$$
$$\quad inr = linearD\ inrF\ inr$$
$$\quad jam = linearD\ jamF\ jam$$

Generalização de AD

$$
\begin{array}{l}
\textbf{instance } \mathit{Scalable}\ k\ s \Rightarrow \mathit{NumCat}\ D_k\ s \textbf{ where} \\
\quad \mathit{negateC} = \mathit{linearD}\ \mathit{negateC}\ \mathit{negateC} \\
\quad \mathit{addC} = \mathit{linearD}\ \mathit{addC}\ \mathit{addC} \\
\quad \mathit{mulC} = D\ (\lambda(a,b) \rightarrow (a * b, \mathit{scale}\ b\ \triangledown\ \mathit{scale}\ a))
\end{array}
$$

...

Generalizando RAD e FAD

Obter FAD e RAD de algoritmo AD genérico: forçar a direção da composição de morfismos

Conversão da escrita de funções

$f :: a \ `\mathtt{k}` \ b \Rightarrow (\circ f) :: (b \ `\mathtt{k}` \ r) \rightarrow (a \ `\mathtt{k}` \ r)$ where r is any object of k.

Definição de novo tipo

$$\textbf{newtype}\ Cont_k^r\ a\ b = Cont\ ((b \ `\mathtt{k}` \ r) \rightarrow (a \ `\mathtt{k}` \ r))$$

Funtor derivado dele

$$\begin{array}{l} cont :: Category\ k \Rightarrow (a \ `\mathtt{k}` \ b) \rightarrow Cont_k^r\ a\ b \\ cont\ f = Cont\ (\circ f) \end{array}$$

Instancia deduzida para RAD genérico

$$
\begin{array}{l}
\textbf{instance } \mathit{Category}\ k \Rightarrow \mathit{Category}\ \mathit{Cont}_k^r \textbf{ where} \\
\quad \mathit{id} = \mathit{Cont}\ \mathit{id} \\
\quad \mathit{Cont}\ g \circ \mathit{Cont}\ f = \mathit{Cont}\ (f \circ g) \\
\textbf{instance } \mathit{Monoidal}\ k \Rightarrow \mathit{Monoidal}\ \mathit{Cont}_k^r \textbf{ where} \\
\quad \mathit{Conf}\ f \times \mathit{Cont}\ g = \mathit{Cont}\ (\mathit{join} \circ (f \times g) \circ \mathit{unjoin}) \\
\textbf{instance } \mathit{Cartesian}\ k \Rightarrow \mathit{Cartesian}\ \mathit{Cont}_k^r \textbf{ where} \\
\quad \mathit{exl} = \mathit{Cont}\ (\mathit{join} \circ \mathit{inl}); \mathit{exr} = \mathit{Cont}\ (\mathit{join} \circ \mathit{inr}) \\
\quad \mathit{dup} = \mathit{Cont}\ (\mathit{jam} \circ \mathit{unjoin}) \\
\textbf{instance } \mathit{Cocartesian}\ k \Rightarrow \mathit{Cocartesian}\ \mathit{Cont}_k^r \textbf{ where} \\
\quad \mathit{inl} = \mathit{Cont}\ (\mathit{exl} \circ \mathit{unjoin}); \mathit{inr} = \mathit{Cont}\ (\mathit{exr} \circ \mathit{unjoin}) \\
\quad \mathit{jam} = \mathit{Cont}\ (\mathit{join} \circ \mathit{dup}) \\
\textbf{instance } \mathit{Scalable}\ k\ a \Rightarrow \mathit{Scalable}\ \mathit{Cont}_k^r\ a \textbf{ where} \\
\quad \mathit{scale}\ s = \mathit{Cont}\ (\mathit{scale}\ s)
\end{array}
$$

...

...