逆微分圏に基づいたアローハンドラによるニューラルネットワークプログラミング言語に向けて

眞田 嵩大 [1]　平井 謙信 [2]　星野 恵佑 [2]　勝股 審也 [3]

[1] 福井県立大学 情報センター

[2] 京都大学 数理解析研究所

[3] 京都産業大学

日本ソフトウェア科学会 第 42 回大会

ニューラルネットワークプログラミングの現状

数学的構造に基づいて言語やライブラリが設計されているわけではない

- Python 上のニューラルネットワークライブラリ
  - PyTorch
  - TensorFlow
- ニューラルネットワークの数学的構造を無視している

それどころか，ニューラルネットワークの数学的な構造に関する研究は，いくつかの試みはあるものの，決定的なものはない．

大目標：ニューラルネットワークの数学的構造に基づくプログラミング言語

ニューラルネットワークの数学的構造を反映した言語が必要

- Python は型安全でない
- では他の型安全な言語で実装すればよい？
- ニューラルネットワークの数学的構造を反映した言語が必要
  - ニューラルネットワークの設計
  - 誤差逆伝播，推論などの振る舞い
  - 重みパラメータの更新
- そもそもニューラルネットワークの数学的構造は何か？

本研究：ニューラルネットワークの新しい圏論的・代数的意味論

問い：ニューラルネットワークの数学的構造は何か？それを用いてニューラルネットワークのためのプログラミング言語を作れるか？

逆微分圏上の次数付き強プロモナド $\mathcal{A}$ の代数と準同型を用いてニューラルネットワークの設計と振る舞いを捉える

アローに対応する代数的エフェクトでニューラルネットワークを設計し，エフェクトハンドラによって振る舞いを定義するスタイルのプログラミング言語が可能になる．

- ニューラルネットワークの数学的構造に基づいたプログラミング言語が実現できる
- 層の振る舞いはプログラマが定義できる．プリミティブ関数の微分は逆微分圏に由来する構造を使って自動的にできる．
- ニューラルネットワークの設計と振る舞いを分離できる．
- 量子化など勾配の情報を壊すような操作も，ユーザーが定義するエフェクトハンドラとして扱える．

関連研究

- 逆微分圏 [Cockett+, CSL 2020] とそれの基づいたプログラミング言語 [Abadi & Plotkin, POPL 2020][Cruttwell+, ACT 2020].
  - 逆方向自動微分を圏論的に捉えるための概念
- アローに対応する代数的エフェクトとハンドラ [眞田, JFP 2024].
  - 強プロモナドの代数と準同型を考察．アローハンドラを持つプログラミング言語を導出
- ニューラルネットワークの圏論的・代数的意味論とプログラミング言語 [眞田, 2025. Submitted]
  - 次数付き強プロモナドの代数に基づく
- パラ構成とレンズによるニューラルネットワークの圏論的構成 [Capucci+, ACT 2021][Cruttwell+, ESOP 2022]
  - ニューラルネットワークの構造と誤差逆伝播をレンズを用いて捉える
  - パラメータの入出力をパラ構成で捉える

目次

アイデア

次数付き強プロモナド

逆微分圏

逆微分圏上の次数付きプロモナド

3つのアイデア

1. ニューラルネットワークは層の列である．
   - 強プロモナドの代数として捉える．
2. ニューラルネットワークの各層は2種類の入出力を持つ．
   - データの入出力とパラメータの入出力．
   - 次数付き強プロモナドの代数として捉える．
3. ニューラルネットワークの設計と振る舞いは分離して記述するべきである．
   - 設計 = ニューラルネットワークの層どうしがどう結合しているか
   - 振る舞い = 推論時や学習時にどのような計算を行うか
   - HTML と CSS の関係のように，あるいは TEX の文章とスタイルの関係のように分離したい
   - これはエフェクトハンドラ（圏論的には準同型射）で実現できる．

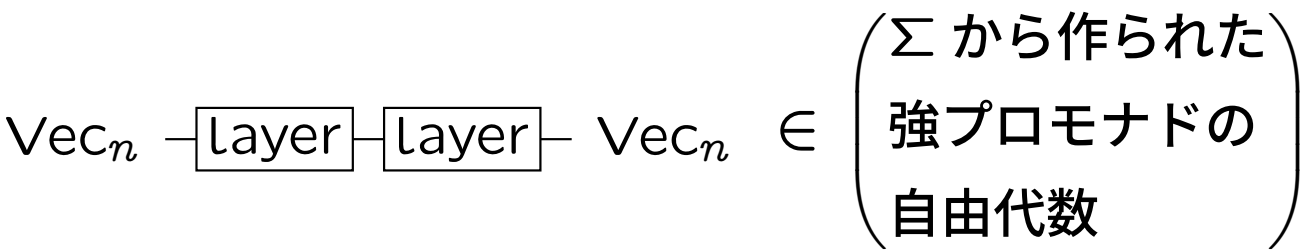
ニューラルネットワークは層の列である

層をアローに対応する代数的演算と考え，ニューラルネットワークを強プロモナドの自由代数として捉える

シグネチャ（代数的演算の集合）を $\Sigma = \{\mathsf{layer}\}$ とする．
プログラミング言語的には

$$v : \mathsf{Vec}_n \vdash \begin{pmatrix} \textbf{let}\, x_1 \leftarrow \mathsf{layer}(v)\ \textbf{in} \\ \textbf{let}\, x_2 \leftarrow \mathsf{layer}(x_1)\ \textbf{in} \\ \textbf{return}\, x_2 \end{pmatrix} : \mathsf{Vec}_n$$

ストリング図式を使って書くと



Q. なぜモナドではなくアロー（= プロモナド）？
A. モナドの自由代数は木．アローの自由代数は列．アローが層の列を表現するのに適している．また逆微分圏と組み合わせる時に，入力の情報が必要である．

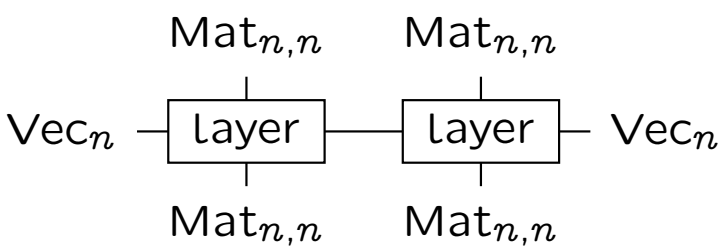
ニューラルネットワークの各層は２種類の入出力を持つ

データの入出力とパラメータの入出力．次数付き強プロモナドの自由代数として捉える．

各層はパラメータ（重み行列やバイアスベクトル）を受け取る．また，学習時にはそれを更新して新しいパラメータを返す．
プログラミング言語的には

$$v : \mathsf{Vec}_n \vdash^{\mathsf{Mat}_{n,n}\cdot\mathsf{Mat}_{n,n}}_{\mathsf{Mat}_{n,n}\cdot\mathsf{Mat}_{n,n}} \begin{pmatrix} \textbf{let}\, x_1 \leftarrow \mathsf{layer}(v)\ \textbf{in} \\ \textbf{let}\, x_2 \leftarrow \mathsf{layer}(x_1)\ \textbf{in} \\ \textbf{return}\, x_2 \end{pmatrix} : \mathsf{Vec}_n$$

ストリング図式を使って書くと



$$\in \begin{pmatrix} \Sigma\text{ から作られた} \\ \text{次数付き強プロモナドの} \\ \text{自由代数} \end{pmatrix}$$

ニューラルネットワークの設計と振る舞いは分離して記述するべきである

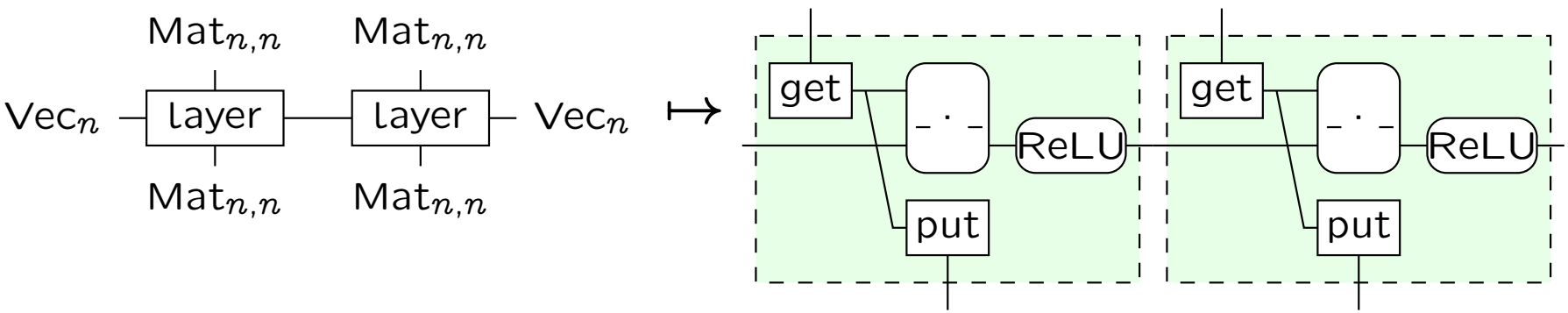
エフェクトハンドラ（圏論的には代数の間の準同型射）で実現できる.

プログラミング言語的には

$$
\begin{array}{l}
\textbf{handle} \\
\quad \textbf{let}\, x_1 \leftarrow \mathsf{layer}(v)\ \textbf{in} \\
\quad \textbf{let}\, x_2 \leftarrow \mathsf{layer}(x_1)\ \textbf{in} \\
\quad \textbf{return}\, x_2 \\
\textbf{with}\ H
\end{array}
\qquad
H = \left\{
\begin{array}{l}
\mathsf{layer}(x), k \mapsto \\
\quad \textbf{let}\, w \leftarrow \mathsf{get}()\ \textbf{in} \\
\quad \textbf{let}\, z \leftarrow w \cdot x\ \textbf{in} \\
\quad \textbf{let}\, \_ \leftarrow \mathsf{put}(w)\ \textbf{in} \\
\quad k \bullet \mathsf{ReLU}(z)
\end{array}
\right\}
$$

ストリング図式を使って書くと

（自由代数） $\xrightarrow{\text{普遍性で生じる準同型射}}$ （自由でない代数）



アイデア

次数付き強プロモナド

逆微分圏

逆微分圏上の次数付きプロモナド

モナドと代数

モナドは代数の「型」を指定する概念

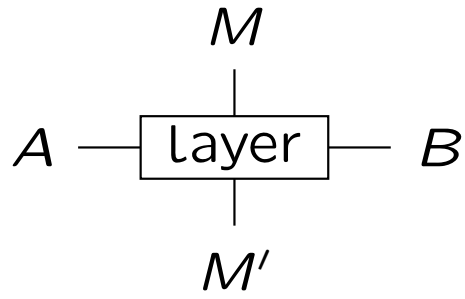
ニューラルネットワークを代数として捉えたい.

- モナド $\mathcal{T}\colon \mathbb{C} \to \mathbb{C}$ に対して $\mathcal{T}$ 代数とは $\alpha\colon \mathcal{T}C \to C$ のこと
- 例：モナド $\mathcal{T}C = \{\star\} + C \times C$ に対して $\mathbb{N}$ は $\mathcal{T}$ 代数である

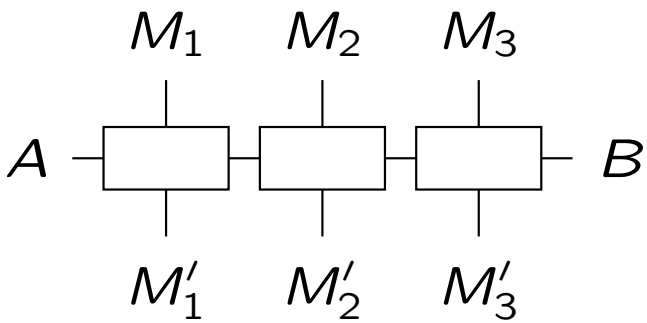
$$
\begin{array}{ccccc}
\{\star\} & + & \mathbb{N} \times \mathbb{N} & \to & \mathbb{N} \\
\star & & & \mapsto & 0 \\
 & & (L_1, L_2) & \mapsto & L_1 + L_2
\end{array}
$$

Q. $\mathcal{X}$ 代数がニューラルネットワーク全体の集合となるようなモナド的概念 $\mathcal{X}$ は何か？
A. $\mathcal{X} =$ 次数付き強プロモナド $\mathcal{A}$.

次数付きプロモナド



$\in \Sigma$



$\in \mathcal{A}_{\Sigma,(M_1 \boxtimes M_2 \boxtimes M_3, M'_1 \boxtimes M'_2 \boxtimes M'_3)}(A, B)$

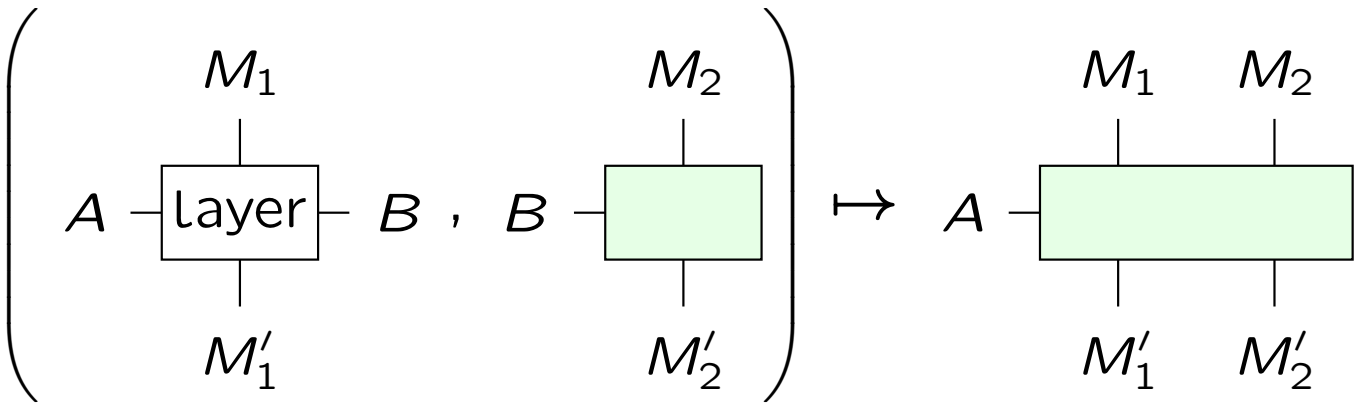
$\mathbb{C}$ を圏，$\mathcal{M} = (\mathcal{M}, \boxtimes, J)$ をモノイダル圏とする．

$\mathbb{C}$ 上の $\mathcal{M}$ 次数付きプロモナド $\mathcal{A}$ とは，緩モノイダル関手 $\mathcal{A} \colon \mathcal{M} \to \mathbf{Prof}(\mathbb{C}, \mathbb{C})$ である．

モノイダル圏 $\mathrm{IO}(\mathcal{M}) = \mathcal{M}^{\mathrm{op}} \times \mathcal{M}$ とする．$\mathrm{IO}(\mathcal{M})$ 次数付きプロモナド $\mathcal{A}$ を考える．対象 $A, B \in \mathbb{C}$ と $(M, M') \in \mathrm{IO}(\mathcal{M})$ に対して，集合 $\mathcal{A}_{\Sigma,(M,M')}(A, B)$ がある．

次数付きプロモナドの代数

$$\mathcal{A}_{\Sigma,(M_1,M_1')}(A,B) \quad \times \quad G_{(M_2,M_2')}(B) \quad \xrightarrow{\alpha} \quad G_{(M_1 \boxtimes M_2, M_1' \boxtimes M_2')}(A)$$



$\mathcal{A}$ を $\mathbb{C}$ 上の $\mathcal{M}$ 次数付きプロモナドとする．

$\mathcal{A}$ 代数とは，前層の族 $\{G_M : \mathbb{C}^{\mathrm{op}} \to \mathbf{Set}\}_{M \in \mathcal{M}}$ と射の族

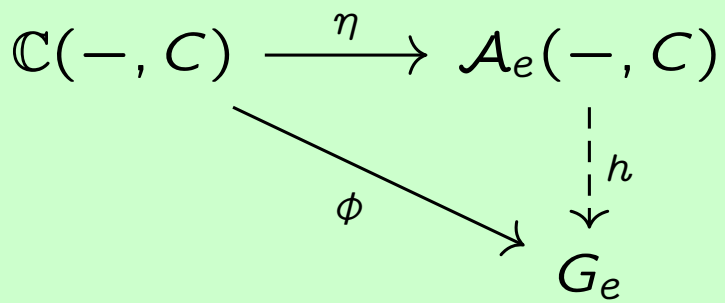
$$\{\alpha_{A,B,M_1,M_2} : \mathcal{A}_{M_1}(A,B) \times G_{M_2}(B) \to G_{M_1 \boxtimes M_2}(A)\}_{M,M'}$$

であって，適切な公理を満たすもの．

自由代数の普遍性

Proposition（自由代数の普遍性）

$C \in \mathbb{C}$．(G, α) を $\mathcal{A}$ 代数とする．このとき以下の図式を可換にする準同型射 h が一意的に存在する．



- 自由代数 $\mathcal{A}_M(-, C)$ でニューラルネットワークの設計を記述する
- 自由でない代数 G_M でニューラルネットワークの振る舞い（例えば誤差逆伝播による学習）を記述する
- 普遍性によって生じる準同型射 h が設計と振る舞いを結びつける

アイデア

次数付き強プロモナド

逆微分圏

逆微分圏上の次数付きプロモナド

逆微分圏

逆微分圏 [Cockett+, CSL 2020] は**逆方向自動微分**を捉えるための圏．
射を「微分する」という操作を持つ．

微分は和を保つ：$\frac{\mathrm{d}f}{\mathrm{d}x} + \frac{\mathrm{d}g}{\mathrm{d}x} = \frac{\mathrm{d}(f+g)}{\mathrm{d}x}$．これを定式化するために，逆微分圏 $\mathbb{X}$ の射は加法的な構造を持つ．

$$\frac{f\colon A \to B \quad g\colon A \to B}{(f+g)\colon A \to B}$$

逆微分は次の**逆微分作用素** $\mathsf{R}[-]$ で定式化される．

$$\frac{f\colon A \to B}{\mathsf{R}[f]\colon A \times B \to A} \qquad \frac{f(x) = (f_j(x_1, \ldots, x_n))_{j=1}^m}{\mathsf{R}[f](x,y) = \left(\sum_{j=1}^m y_j \cdot \frac{\partial f_j}{\partial x_i}\right)_{i=1}^n}$$

逆微分作用素は和の保存や連鎖律などに対応する公理を満たす．

$$\mathsf{R}[f] + \mathsf{R}[g] = \mathsf{R}[f+g], \quad \ldots$$

アイデア

次数付き強プロモナド

逆微分圏

逆微分圏上の次数付きプロモナド

準同型（エフェクトハンドラ）としての逆微分作用素

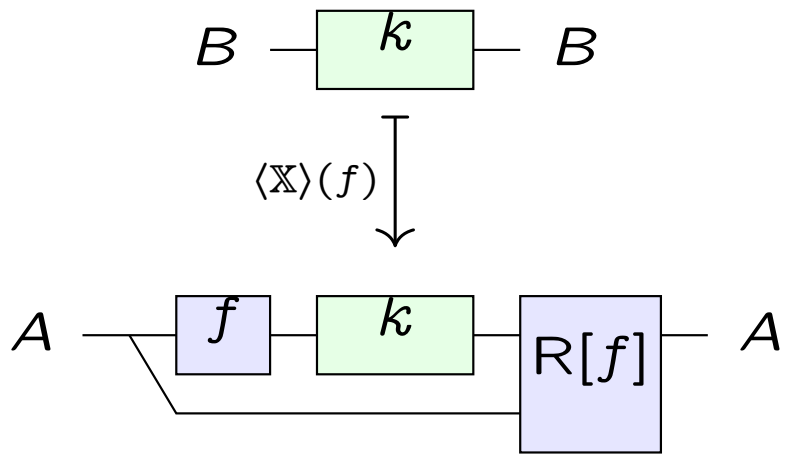
逆微分作用素 $\mathsf{R}[-]$ はプロモナドの代数の間の準同型射を作る

逆微分圏 $\mathbb{X}$ に対して，前層 $\langle\mathbb{X}\rangle$ を以下で定める．

$$\langle\mathbb{X}\rangle : \mathbb{X}^{\mathrm{op}} \to \mathbf{Set}$$
$$A \mapsto \langle\mathbb{X}\rangle(A) := \mathbb{X}(A, A)$$

射 $f : A \to B$ に対して，$\langle\mathbb{X}\rangle(f) : \langle\mathbb{X}\rangle(B) \to \langle\mathbb{X}\rangle(A)$ は

$$\begin{array}{ccc} \langle\mathbb{X}\rangle(B) = \mathbb{X}(B, B) & \ni & \\ \Big\downarrow{\scriptstyle \langle\mathbb{X}\rangle(f)} & & \\ \langle\mathbb{X}\rangle(A) = \mathbb{X}(A, A) & \ni & \end{array}$$



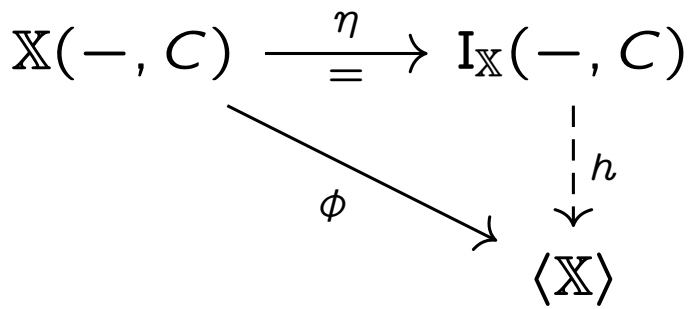
前層 $\langle\mathbb{X}\rangle$ は恒等プロモナド $\mathrm{I}_{\mathbb{X}}$（$\mathrm{I}_{\mathbb{X}}(A, B) = \mathbb{X}(A, B)$）の代数である．

$$\alpha : \mathbb{X}(A, B) \times \langle\mathbb{X}\rangle(B) \to \langle\mathbb{X}\rangle(A)$$

普遍性

微分はハンドラ（準同型射）をつくる

$\langle \mathbb{X} \rangle$ は恒等プロモナド $\mathrm{I}_{\mathbb{X}}$ の代数である．
ゆえに，以下の図式を可換にする自由 $\mathrm{I}_{\mathbb{X}}$ 代数から $\langle \mathbb{X} \rangle$ への準同型射 h が一意的に存在する．



米田の補題より ϕ を決めることは $\mathbb{X}(C, C)$ の元を指定することと同じ．直観的には，$\mathbb{X}(C, C)$ の元は損失関数を微分して得られる勾配ベクトル場を指定することに対応する．

$\mathbb{X} = \mathbf{Smooth}$ のとき，ϕ を指定することは，滑らかな関数 $\mathbb{R}^C \to \mathbb{R}^C$ を指定することに対応する．

逆微分圏上の強プロモナド

$\mathcal{A}_\Sigma : \mathbb{X} \nrightarrow \mathbb{X}$ を逆微分圏 $\mathbb{X}$ 上の強プロモナドとする．先ほどと同様に前層 $\langle \mathcal{A}_\Sigma \rangle$ がある．

$$\langle \mathcal{A}_\Sigma \rangle : \mathbb{X}^{\mathrm{op}} \to \mathbf{Set}$$

これを $\mathcal{A}_\Sigma$ 代数とみなすには以下の射があればよい．

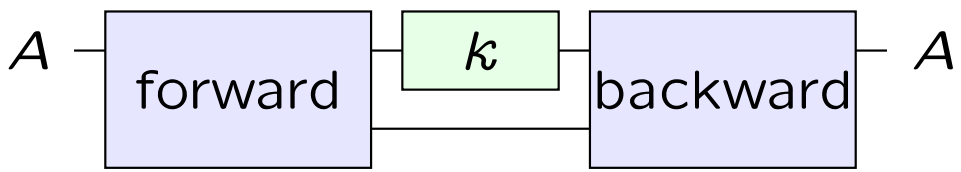
$$\alpha_{A,B} : \mathcal{A}_\Sigma(A, B) \times \langle \mathcal{A}_\Sigma \rangle(B) \to \langle \mathcal{A}_\Sigma \rangle(A)$$

そのためには，各演算（層） $A \xrightarrow{\boxed{\mathsf{layer}}} B \in \Sigma$ に対して以下の射があればよい．

$$q^{\mathsf{layer}} : \langle \mathcal{A}_\Sigma \rangle(B) \to \langle \mathcal{A}_\Sigma \rangle(A)$$

$$B \xrightarrow{\boxed{k}} B \mapsto A \xrightarrow{\boxed{q^{\mathsf{layer}}(k)}} A$$

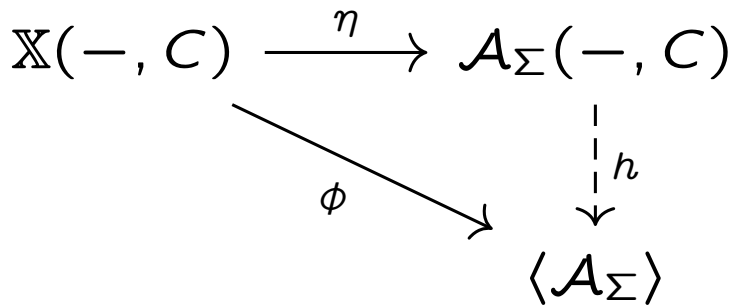
誤差逆伝播のための $A \xrightarrow{\boxed{q^{\mathsf{layer}}(k)}} A$ は以下の形になる：



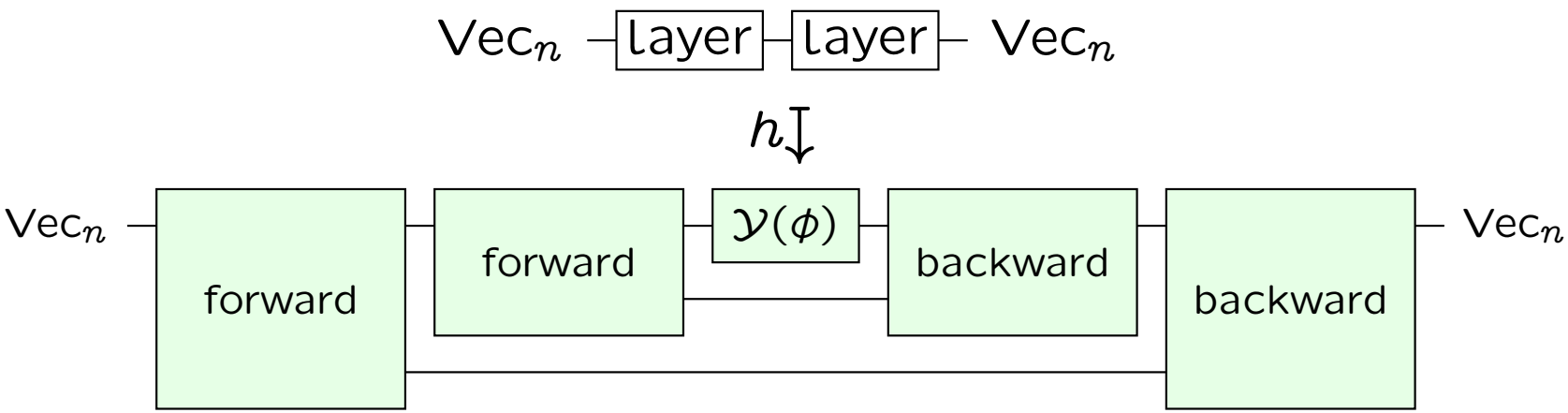
q^{layer} が，各層 layer の振る舞いを決定している

普遍性

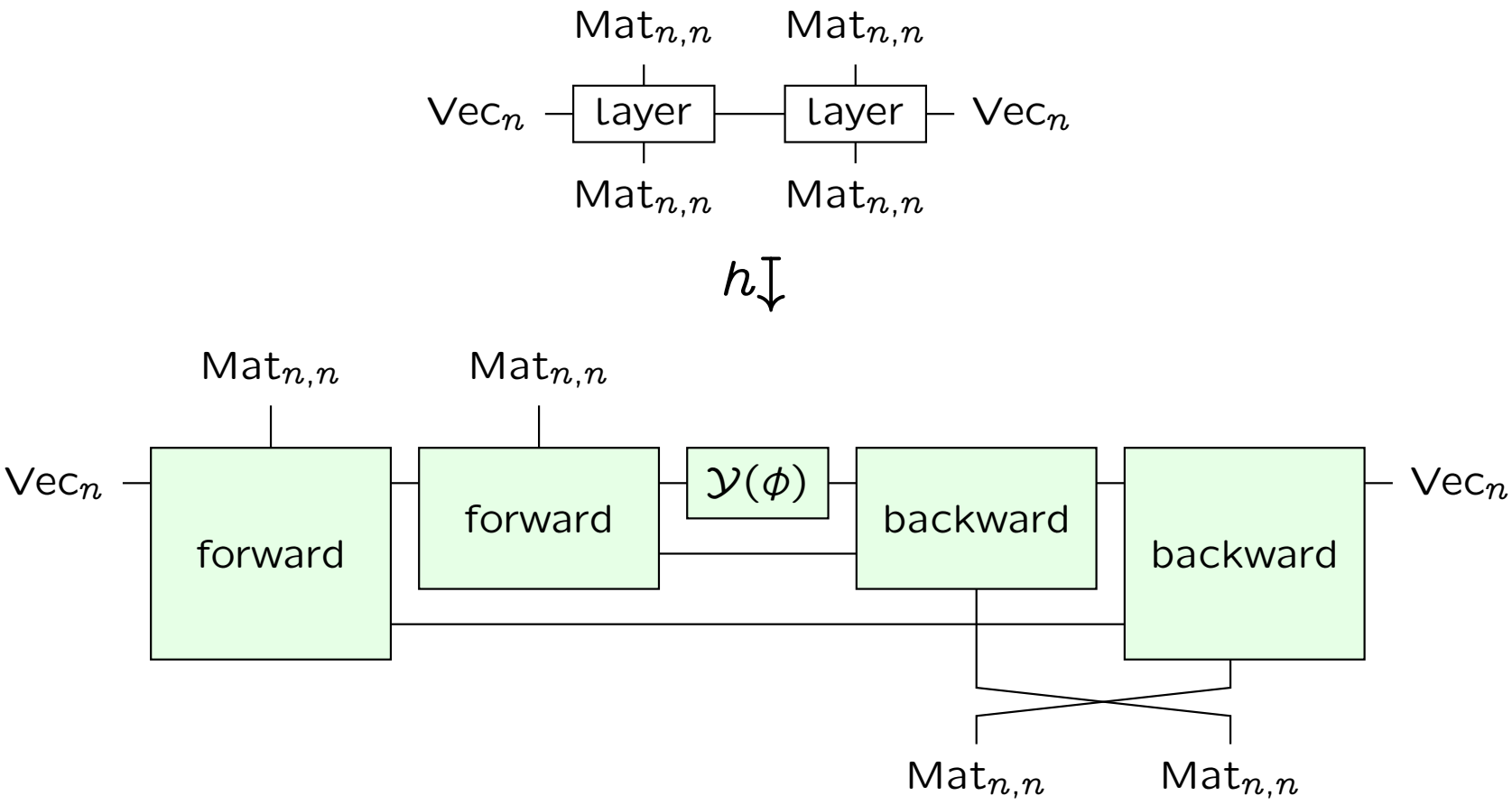
以下の図式を可換にする自由 $\mathcal{A}_\Sigma$ 代数 $\mathcal{A}_\Sigma(-, C)$ から自由でない $\mathcal{A}_\Sigma$ 代数 $\langle \mathcal{A}_\Sigma \rangle$ への準同型射 h が一意的に存在する.



この h が誤差逆伝播の振る舞いを実装するハンドラの意味を与える



逆微分圏上の次数付き強プロモナド



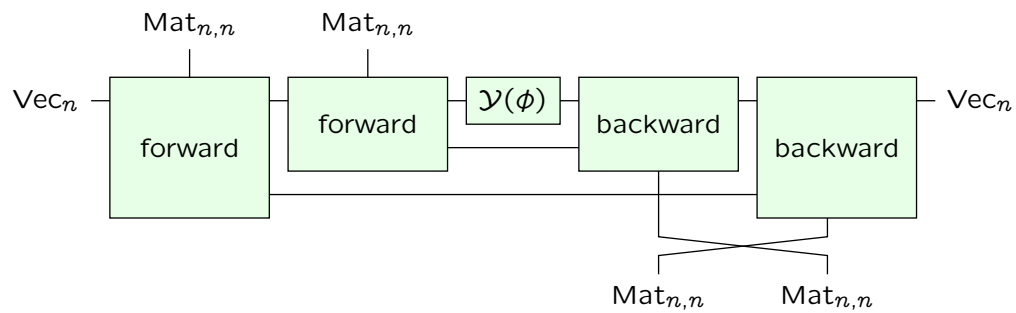
Theorem（詳細は講演論文の定理 19 を参照）

forward と *backward* を適切に定めることで，h はニューラルネットワークの設計を表す要素を，損失関数の勾配に基づいて重みを更新する振る舞いを表す要素に送る準同型射になる．

まとめ

- 微分はハンドラ（代数の間の普遍性によって生じる準同型射）をつくる
- 逆微分圏上の次数付き強プロモナドの代数がニューラルネットワークの圏論的・代数的意味論を与える
- またその構造はアローに対応する代数的エフェクトとアローハンドラを持つプログラミング言語の意味論も与える
- よって，ニューラルネットワークの設計を代数的エフェクト，振る舞いをハンドラで記述するようなニューラルネットワークプログラミング言語が得られるであろう．

$$
\left[\!\!\left[
\begin{array}{l}
\textbf{rev handle} \\
\quad \textbf{let}\, x_1 \leftarrow \mathsf{layer}(v)\ \textbf{in} \\
\quad \textbf{let}\, x_2 \leftarrow \mathsf{layer}(x_1)\ \textbf{in} \\
\quad \textbf{return}\, x_2 \\
\textbf{with}\ H
\end{array}
\right]\!\!\right] =
$$



エフェクトハンドラの型付け規則

我々の「逆ハンドラ」

$$\frac{\diamond \fatsemi x : C \vdash P : C \quad (k : \delta \rightsquigarrow \delta \fatsemi z : \gamma \vdash Q_{\mathsf{op}} : \gamma)_{\mathsf{op}:\gamma \to \delta}}{\vdash H : C \text{ rev handler}}$$

$$\frac{\Gamma \fatsemi x : A \vdash R : C \quad \vdash H : C \text{ rev handler}}{\Gamma \fatsemi x : A \vdash \mathbf{rev\ handle}\, R\, \mathbf{with}\, H : A}$$

通常のアローハンドラ

$$\frac{\diamond \fatsemi x : C \vdash P : D \quad (k : \delta \rightsquigarrow D \fatsemi z : \gamma \vdash Q_{\mathsf{op}} : D)_{\mathsf{op}:\gamma \to \delta}}{\vdash H : C \Rightarrow D}$$

$$\frac{\Gamma \fatsemi \Delta \vdash R : C \quad \vdash H : C \Rightarrow D}{\Gamma \fatsemi \Delta \vdash \mathbf{handle}\, R\, \mathbf{with}\, H : D}$$