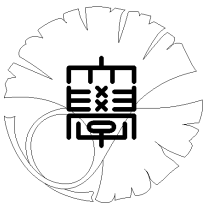
数理科学実践研究レター　2023–9　May 09, 2023

機械学習の圏論的解釈と **Haskell** インターフェースの提案

by

後藤 祐輝



UNIVERSITY OF TOKYO

GRADUATE SCHOOL OF MATHEMATICAL SCIENCES

KOMABA, TOKYO, JAPAN

機械学習の圏論的解釈と Haskell インターフェースの提案

後藤 祐輝[1] (東京大学大学院 数理科学研究科)
Yuki Goto (Graduate School of Mathematical Sciences, The University of Tokyo)

概 要

本稿では，圏論的定式化を通して，機械学習における学習手法とレンズとの関係性を数学的に論じる．さらに，その結果を基盤とした機械学習の Haskell インターフェースを提案する．

1 はじめに

近年，機械学習は様々な場面に応用され生活に不可欠な技術になっており，それに伴って機械学習の数理的分析も広く行われている．しかし，機械学習の圏論的な解釈を行った研究はまだ少ない．機械学習に圏論的解釈を与える利点としては，以下の 2 点が挙げられる．

- 圏論とはあらゆる数学的概念を「対象」とそれを繋ぐ「射」という形式で抽象化する理論であり，その対象や射の具体的な中身についての情報は必要としない．この性質から，具体的な形を見る限りでは全く異なる概念が，圏論的な抽象化を介することで，実は同じ構造をしていると判明することがある．したがって，圏論をインターフェースとして，一方の数学的概念の理論をもう一方の数学的概念の分析に利用できる可能性が生まれる．
- 圏論は型理論と深い関連があることが古くから知られている．そのため，数学的概念に圏論的解釈を与えることで，Haskell を始めとするプログラミング言語上でその数学的概念を扱うときの方針が与えられる．

本稿では，上に述べた圏論的解釈の 2 つの利点に対応して，次のような結果を報告する．

- 計算機科学の分野では，Foster et al. [4] により view-update problem の解決のために提唱されたレンズと呼ばれる概念がある．本研究では，Fong–Spivak–Tuyéras [2] によって機械学習の圏論的解釈として提案された圏が，このレンズが成す圏の射のパラメータ化として得られることを示した．機械学習とレンズの関係性については Fong–Johnson [3] でも言及されているが，本研究では，レンズが成す圏の射のパラメータ化との圏同型を示すという具体的な形で，機械学習とレンズの関係性を示した．以上の結果により，機械学習とレンズの間には明確な関係性があることが分かり，レンズの理論を機械学習の分析に応用できる可能性が示唆されたことになる．これについては，本稿の第 2 節で述べる．
- 前述の結果を基盤として，機械学習のインターフェースの Haskell 上での実装例を作成した．なお，プログラミング言語に Haskell を選んだ理由は，Haskell が静的型付け言語であるため Python などと違い静的型チェックの恩恵を受けられる点，Haskell が抽象化に強く前述の圏論的解釈をプログラムとして実現するための十分な言語機能を持っている点，そして Haskell にはよく整備されたレンズライブラリがすでに存在している点の 3 点である．これについては，本稿の第 3 節で述べる．

ここで，view-update problem とレンズの概要を述べておく．
あるデータ A の一部分 B を取り出したとき，B に加えた変更と同様の変更を A にも施したいという問題を考える．データの階層が A と B の 2 段階であれば，B に対する変更を A にも伝播するのはそれほど難しくない．しかし，実務においてはデータ構造は何重にもネストしていることが多い．データ A の中に構造 B_1 があり，B_1 の中にさらに構造 B_2 があるという階層が n 層あるとき，n が大きくなるにつれて B_n に加えた変更を A にも伝播する方法は複雑になる．さらに，n によらず B_n の変更を A に伝播する統一的な方法を構築するのは難しい．これが view-update problem である．

[1] ygoto@ms.u-tokyo.ac.jp

Foster et al. が提唱したレンズは，内部のデータを取り出すゲッターとそのデータに変更を施すセッターの対である．ここには合成と呼ばれる操作が定義されており，A の中の B_1 を操作するレンズ λ_1 から B_{n-1} の中の B_n を操作するレンズ λ_n までの n 個のレンズを合成することで，A の中の B_n を一気に操作するレンズが得られるようになっている．この便利さから，Haskell などの関数型言語にはレンズを扱うライブラリが用意されており，データの操作に利用されている．
また，レンズは，コモナドや Tambara 加群といった概念を通して，圏論的にも意味のある対象として実現されることが知られており，研究が進められている．本稿ではレンズの圏論的解釈の詳しい領域までは立ち入らないので，この説明は B. Clarke et al. [1] などに譲る．

2　レンズのパラメータ化としての学習機

まず，Fong–Spivak–Tuyéras [2] が機械学習の圏論的解釈として提案した圏の定義を述べる．以下，$\mathbb{S}$ で集合の圏を表す[2]．

定義 1. 圏 $\mathbb{S}$ の対象 A, B に対し，$\mathbb{S}$ 上の A から B への**学習機 (learner)** とは，

- $\mathbb{S}$ の対象 P,
- $\mathbb{S}$ の射 $i\colon P \times A \to B$,
- $\mathbb{S}$ の射 $u\colon P \times A \times B \to P$,
- $\mathbb{S}$ の射 $r\colon P \times A \times B \to A$

から成る 4 つ組 (P, i, u, r) のことである．始域と終域を明示して $(P, i, u, r)\colon A \to B$ と表すこともある．学習機 (P, i, u, r) を成す 4 つの要素は，順に**パラメータ空間 (parametre space)**, **実装射 (implementation morphism)**, **更新射 (update morphism)**, **要求射 (request morphism)** と呼ぶ．

実際の機械学習では，ニューラルネットワークを構築し，教師データを用いてネットワークのパラメータを更新していくという処理を行うが，上記の学習機はそのような機械学習の手法の抽象化になっている．その詳細は Fong–Spivak–Tuyéras [2] に譲り，ここでは省略する．
さて，学習機には以下のように合成をうまく定めることができ，それによって圏を成す．

定義 2. 圏 $\mathbb{S}$ 上の**学習機の圏 (category of learners)** とは，以下で定義される圏 **Learn** である．

- **Learn** の対象は，$\mathbb{S}$ の対象全体とする．
- **Learn** の対象 A, B の間の射は，$\mathbb{S}$ 上の学習機 $(P, i, u, r) : A \to B$ の同型類全体とする．
- **Learn** の射 $(P, i, u, r)\colon A \to B$, $(Q, j, v, s)\colon B \to C$ の合成は，

$$
\begin{aligned}
k\colon (P \times Q) \times A &\longrightarrow C \\
p, q, a &\longmapsto j(q, i(p, a)) \\
w\colon (P \times Q) \times A \times C &\longrightarrow P \times Q \\
p, q, a, c &\longmapsto u(p, a, s(q, i(p, a), c)), v(q, i(p, a), c) \\
t\colon (P \times Q) \times A \times C &\longrightarrow A \\
p, q, a, c &\longmapsto r(p, a, s(q, i(p, a), c))
\end{aligned}
$$

  と定義することで得られる射 $(P \times Q, k, w, t)\colon A \to C$ とする．

次に，レンズの定義を簡単に述べておく．レンズとは大きなオブジェクトの一部分のデータを取得したり更新したりするためのデータ構造で，Haskell に代表される関数型プログラミング言語で利用されている．数学的には以下のように定式化される．詳細は B. Clarke et al. [1] や Foster et al. [4] などを参照されたい．

[2] より一般に $\mathbb{S}$ は有限直積をもつ任意の圏とすることができるが，以降の定義中に出現する射を定めるのを簡単にするため，ここでは $\mathbb{S}$ は集合の圏とした．

定義 3. 圏 $\mathbb{S}$ 上の**レンズの圏 (category of lens)** とは，以下で定義される圏 **Lens** である．

- **Lens** の対象は，$\mathbb{S}$ の対象全体とする．
- **Lens** の対象 A, B の間の射は，$\mathbb{S}$ の 2 つの射 $g\colon A \to B$, $p\colon A \times B \to A$ の組 (g, p) とする．
- **Lens** の射 $(g, p)\colon A \to B$, $(h, q)\colon B \to C$ の合成は，

$$\begin{aligned} k\colon A &\longrightarrow C & \qquad r\colon A \times C &\longrightarrow A \\ a &\longmapsto h(g(a)) & a, c &\longmapsto p(a, q(g(a), c)) \end{aligned}$$

  と定義することで得られる射 $(k, r)\colon A \to C$ とする．

ここで，射のパラメータ化と呼ばれる操作を導入する．これは，対称モノイダル圏から射の構造を変えた新しい圏を得る操作である．

定義 4. 対称モノイダル圏 $\mathcal{M}$ に**射のパラメータ化 (parametrisation of morphisms)** を施して得られる圏とは，以下で定義される圏 $\mathbf{Prm}(\mathcal{M})$ である．

- $\mathbf{Prm}(\mathcal{M})$ の対象は，$\mathcal{M}$ の対象全体とする．
- $\mathbf{Prm}(\mathcal{M})$ の対象 A, B の間の射は，$\mathcal{M}$ の対象 P と $\mathcal{M}$ の射 $f\colon P \otimes A \to B$ の組 (P, f) の同型類全体とする．
- $\mathbf{Prm}(\mathcal{M})$ の射 $(P, f)\colon A \to B$, $(Q, g)\colon B \to C$ の合成は，

$$h \coloneqq \left[(P \otimes Q) \otimes A \longrightarrow Q \otimes (P \otimes A) \xrightarrow{\mathrm{id} \otimes \mathrm{f}} Q \otimes B \xrightarrow{g} C \right]$$

  と定義することで得られる射 $(P \otimes Q, h)\colon A \to C$ とする．なお，上記のラベルのない射は標準的な同型射である．

すると，次の 2 つの定理で示すように，レンズの圏は対称モノイダル圏と見なすことができ，それに射のパラメータ化を施したものは学習機の圏と同型になる．

定理 5. 圏 **Lens** には対称モノイダル圏の構造が定まる．

証明の概略. **Lens** の対象 A, C に対して，$A \otimes C \coloneqq A \times C$ とする．すなわち，対象の上ではモノイダル積を直積で定める．また，**Lens** の射 $(g, p)\colon A \to B$, $(h, q)\colon C \to D$ に対し，

$$\begin{aligned} k\colon A \times C &\longrightarrow B \times D & \qquad r\colon (A \times C) \times (B \times D) &\longrightarrow A \times C \\ a, c &\longmapsto g(a), h(c) & a, c, b, d &\longmapsto p(a, b), q(c, d) \end{aligned}$$

と定義し，射のモノイダル積を $(g, p) \otimes (h, q) \coloneqq (k, r)\colon A \times C \to B \times D$ で定める．このようにすると，圏 **Lens** には対称モノイダル圏の構造が定まる． □

定理 6. 圏 **Lens** を定理 5 の通りに対称モノイダル圏と見なすと，圏同型 $\mathbf{Prm}(\mathbf{Lens}) \cong \mathbf{Learn}$ が成立する．

証明の概略. 定義に従うと，$\mathbf{Prm}(\mathbf{Lens})$ の射は $(P, (g, p))\colon A \to B$ という形をしており，(g, p) は **Lens** の射 $(g, p)\colon P \otimes A \to B$ である．したがって，g, p はそれぞれ $\mathbb{S}$ の射 $g\colon P \times A \to B$, $p\colon P \times A \times B \to P \times A$ である．p と第 1 射影との合成を $u\colon P \times A \times B \to P$ とおき，p と第 2 射影との合成を $r\colon P \times A \times B \to A$ とおくと，すると，**Learn** の射 $(P, g, u, r)\colon A \to B$ が得られる．この対応が，定理の主張の圏同型を与える． □

以上により，学習機はレンズのパラメータ化であるという形で，機械学習とレンズの関係性を述べることができた．続く節では，この関係性を用いた機械学習の Haskell インターフェースを提案する．

3 Haskell インターフェースの提案

この節では，前節の内容を踏まえて，機械学習のインターフェースの Haskell での実装例について述べる．

圏 $\mathbf{Prm}(\mathbf{Lens})$ の射 $A \to B$ は，$\mathbb{S}$ のある対象 P と $\mathbf{Lens}$ の射 $(g, p)\colon P \times A \to B$ の組 $(P, (g, p))$ であった．すなわち，

$$\mathrm{Hom}_{\mathbf{Prm}(\mathbf{Lens})}(A, B) = \coprod_{P \in \mathbb{S}} \mathrm{Hom}_{\mathbf{Lens}}(P \times A, B)$$

と書ける．したがって，圏 $\mathbf{Prm}(\mathbf{Lens})$ の射の型は次のように Haskell で定義できる．

```
import Control.Lens.Lens

data ParamedLens a b = forall p. ParamedLens (Lens' (p, a) b)
```

前節の結果によって学習機とレンズの間に明確な対応が分かっているため，上記の定義では Haskell に存在する既存のレンズライブラリを使用できている．これによって，レンズライブラリに定義されている膨大なコンビネータを流用することができ，学習機の定義をそのまま用いるのに比べて利便性が高まっていると言える．

さて，$\mathbf{Prm}(\mathbf{Lens})$ は圏であるわけだが，実際に以下のようにすると `ParamedLens` を `Category` のインスタンスにすることができる．

```
import qualified Control.Arrow as Arr
import qualified Control.Category as Cat
import Control.Lens.Lens
import Control.Lens.Iso
import Data.Tuple

assoc :: ((a, b), c) -> (a, (b, c))
assoc ((a, b), c) = (a, (b, c))

assoc' :: (a, (b, c)) -> ((a, b), c)
assoc' (a, (b, c)) = ((a, b), c)

lunit :: ((), a) -> a
lunit (_, a) = a

lunit' :: a -> ((), a)
lunit' a = ((), a)

lensSecond :: Lens' a b -> Lens' (c, a) (c, b)
lensSecond lens h (c, a) = rstr . Arr.second (lens $ fmap snd . h . (c, )) $ (c, a)

instance Cat.Category ParamedLens where
  id = ParamedLens (iso lunit lunit')
  ParamedLens g . ParamedLens f = ParamedLens (iso assoc assoc' . lensSecond f . g)
```

これにより，`Category` に定義されている `<<<` や `>>>` などの演算子を用いて，学習機の合成を行うことができる．

以上のように，機械学習とレンズの関係性を利用したインターフェースを Haskell 上で提供することで，Haskell の静的型チェックの恩恵により堅牢なプログラムを組むことができるようになる他，すでに存在するレンズや圏に関する便利なコンビネータを流用することができる．

4　おわりに

本稿では，Fong–Spivak–Tuyéras による機械学習の圏論的定式化がレンズのパラメータ化と見なせることを示し，それをもとにした機械学習の Haskell インタフェースを提案した．しかし，レンズのパラメータ化としての学習機が Haskell 上で実現できることを示したのみで，ニューラルネットワークから学習機を実際に生成する部分の具体的な実装はまだできていない．この箇所を Haskell で実装し，本稿で提案したインターフェースを利用する形で機械学習を実際に実行できるようにするのが今後の課題である．

最後に，本課題を提供してくださった株式会社ニコンの小池哲也様, 中村ちから様, 信田萌伽様に感謝いたします．また，本研究は数物フロンティア・リーディング大学院の助成を受けています．

参考文献

[1] B. Clarke, D. Elkins, J. Gibbons, F. Loregian, B. Milewski, E. Pillmore, M. Román. (2020). Profunctor optics, a categorical update. arXiv:2001.07488.

[2] B. Fong, D. Spivak, R. Tuyéras. (2019). Backprop as functor: a compositional perspective on supervised learning. 34th Annual ACM/IEEE Symposium on Logic in Computer Science, 1–13.

[3] B. Fong, M. Johnson. (2019). Lenses and learners. Proceedings of the Eighth International Workshop on Bidirectional Transformations (Bx 2019), 16–29.

[4] J. N. Foster, M. B. Greenwald, J. T. Moore, B. C. Pierce, A. Schmitt (2005). Combinators for bi-directional tree transformations: a linguistic approach to the view update problem. ACM SIGPLAN Notices, 40(1), 233–246.