Parametric Profunctor Preoptics

Profunctor representations of parametric lenses and pre-lenses and their application to neural networks

Bartosz Milewski

Programming Cafe

2024

Para Optics

- Forward pass: parameters `p`, input `s`
  - yields output `a`
- Backward pass: parameters `p`, input `s`, desired output change `da`
  - yields: parameter delta `dp` and input delta `ds`

```
fwd :: (p, s) -> a
bwd :: (p, s, da) -> (dp, ds)
```

- Composition and identity: (bi-)category of optics.
  - objects are pairs `(s, ds)`, `(a, da)`
  - morphism from `(s, ds)` to `(a, da)` is a pair `(fwd, bwd)`
  - what about parameters?

Para Construction

- Hom-sets are parameterized by objects from a monoidal category $\mathcal{P}$

$$a \xrightarrow[f_p]{} b$$

$$a \xrightarrow[f_p]{} b \xrightarrow[g_q]{} c \qquad a \xrightarrow{h_{q \otimes p}} c$$

- Given a small monoidal category $(V, \otimes, I)$, a *locally graded* category is a category C enriched over $([V, \mathbf{Set}], \star, V(I, -))$ with Day convolution as the tensor product:

$$(F \star G)\ u = \int^{vw} V(v \otimes w, u) \times Fv \times Gw$$

- Hom-object $C(a, b)$ is a functor $F\colon V \to \mathbf{Set}$. You may think of it as a collection of sets indexed by objects of V.
- Composition is given by a natural transformation in $[V, \mathbf{Set}]$:

$$\circ\colon C(b, c) \star C(a, b) \Rightarrow C(a, c)$$

Actegory

An actegory $\mathcal{C}$ is a category in which we define a (functorial) action of a monoidal category $\mathcal{P}$:

$$\bullet : \mathcal{P} \times \mathcal{C} \to C$$

satisfying some obvious coherency conditions (unit and composition):

$$I \bullet c \cong c$$

$$p \bullet (q \bullet c) \cong (p \otimes q) \bullet c$$

Existential Parametric Optic

$$O\langle a, da\rangle\langle p, dp\rangle\langle s, ds\rangle = \int^{m} \mathcal{C}(p \bullet s, m \bullet a) \times \mathcal{C}(m \bullet da, dp \bullet ds)$$

- the residues m are objects of some monoidal category $\mathcal{M}$,
- the parameters $\langle p, dp\rangle$ come from another monoidal category $\mathcal{P}$.

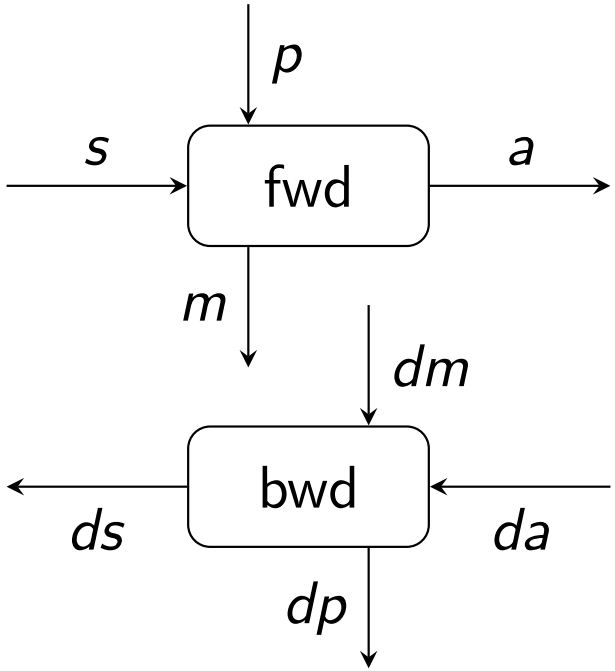
```
data ExLens a da p dp s ds =
  forall m . ExLens ((p, s)  -> (m, a))
                    ((m, da) -> (dp, ds))
```

Compare with:

```
fwd :: (p, s) -> a
bwd :: (p, s, da) -> (dp, ds)
```

Pre-Lens

$$O\langle a, da\rangle\langle m, dm\rangle\langle p, dp\rangle\langle s, ds\rangle = \mathcal{C}(p \bullet s, m \bullet a) \times \mathcal{C}(dm \bullet da, dp \bullet ds)$$




```
data PreLens a da m dm p dp s ds =
  PreLens ((p, s)   -> (m, a))
          ((dm, da) -> (dp, ds))
```

Notation

- $\mathbf{C}$ for category $\mathcal{C}^{op} \times \mathcal{C}$
- $\mathbf{f} : \mathbf{a} \to \mathbf{b}$ a member of the hom-set $\mathbf{C}(\mathbf{a}, \mathbf{b})$ represented by a pair $\langle f : b \to a, g : a' \to b' \rangle : \langle a, a' \rangle \to \langle b, b' \rangle$.
- $\mathbf{m} \bullet \mathbf{a}$ monoidal action of $\mathcal{M}^{op} \times \mathcal{M}$ on $\mathcal{C}^{op} \times \mathcal{C}$:

$$\langle m, dm \rangle \bullet \langle a, da \rangle = \langle m \bullet a, dm \bullet da \rangle$$

- Preoptic:

$$O\langle a, da \rangle \langle m, dm \rangle \langle p, dp \rangle \langle b, db \rangle = \mathcal{C}(p \bullet b, m \bullet a) \times \mathcal{C}(dm \bullet da, dp \bullet db)$$

$$O\ \mathbf{a}\,\mathbf{m}\,\mathbf{p}\,\mathbf{b} = \mathbf{C}(\mathbf{m} \bullet \mathbf{a}, \mathbf{p} \bullet \mathbf{b})$$

- Individual morphism from $\mathbf{a}$ to $\mathbf{b}$, a triple:

$$(\mathbf{m}, \mathbf{p}, \mathbf{f} : \mathbf{m} \bullet \mathbf{a} \to \mathbf{p} \bullet \mathbf{b})$$

Composition and Identity

- Compose:

$$\mathbf{C}(\mathbf{m} \bullet \mathbf{b}, \mathbf{p} \bullet \mathbf{c}) \times \mathbf{C}(\mathbf{n} \bullet \mathbf{a}, \mathbf{q} \bullet \mathbf{b}) \to \mathbf{C}((\mathbf{m} \otimes \mathbf{n}) \bullet \mathbf{a}, (\mathbf{q} \otimes \mathbf{p}) \bullet \mathbf{c})$$

- Lift the first morphism by $(\mathbf{q} \bullet -)$ and the second by $(\mathbf{m} \bullet -)$:

$$\xrightarrow{(\mathbf{q}\bullet)\times(\mathbf{m}\bullet)} \mathbf{C}(\mathbf{q} \bullet \mathbf{m} \bullet \mathbf{b}, \mathbf{q} \bullet \mathbf{p} \bullet \mathbf{c}) \times \mathbf{C}(\mathbf{m} \bullet \mathbf{n} \bullet \mathbf{a}, \mathbf{m} \bullet \mathbf{q} \bullet \mathbf{b})$$

- Compose these hom-sets in $\mathbf{C}$, as long as the two monoidal actions commute:

$$\mathbf{q} \bullet \mathbf{m} \bullet \mathbf{b} \to \mathbf{m} \bullet \mathbf{q} \bullet \mathbf{b}$$

  for all $\mathbf{q}$, $\mathbf{m}$, and $\mathbf{b}$.
- The identity morphism is a triple: $(\mathbf{1}, \mathbf{1}, \mathbf{id})$
- Modulo associators, unitors, and symmetrizers. Hence bi-category

Composition and Identity in Haskell

```
preCompose ::
    PreLens a' da' m dm p dp s ds ->
    PreLens a da n dn q dq a' da' ->
    PreLens a da (m, n) (dm, dn) (q, p) (dq, dp) s ds
preCompose (PreLens f1 g1) (PreLens f2 g2) = PreLens f3 g3
  where
    f3 = unAssoc . second f2 . assoc . first sym .
         unAssoc . second f1 . assoc
    g3 = unAssoc . second g1 . assoc . first sym .
         unAssoc . second g2 . assoc
```

Notice the decoupling of the forward and backward passes.

```
idPreLens :: PreLens a da () () () () a da
idPreLens = PreLens id id
```

Triple Tambara Modules

- A Tambara module is a profunctor $T : \mathcal{C}^{op} \times \mathcal{C} \to \mathbf{Set}$ equipped with a family of natural transformations:

$$\alpha : T\,\langle a, b\rangle \to T\,\langle n \bullet a, n \bullet b\rangle$$

- A triple Tambara module is a functor:

$$T : \mathbf{M}^{op} \times \mathbf{P} \times \mathbf{C} \to \mathbf{Set}$$

  equipped with two families of natural transformations:

$$\alpha : T\,\mathbf{m}\,\mathbf{p}\,\mathbf{a} \to T\,(\mathbf{n} \otimes \mathbf{m})\,\mathbf{p}\,(\mathbf{n} \bullet a)$$

$$\beta : T\,\mathbf{m}\,\mathbf{p}\,(\mathbf{q} \bullet \mathbf{a}) \to T\,\mathbf{m}\,(\mathbf{p} \otimes \mathbf{q})\,\mathbf{a}$$

- Both residues and parameter are accumulated using tensor products
- coherence condition
- natural transformations between triple Tambara modules
- category **TriTamb**

Triple Tambara in Haskell

$$\alpha\colon T\,\mathbf{m}\,\mathbf{p}\,\mathbf{a} \to T\,(\mathbf{n} \otimes \mathbf{m})\,\mathbf{p}\,(\mathbf{n} \bullet a)$$

$$\beta\colon T\,\mathbf{m}\,\mathbf{p}\,(\mathbf{q} \bullet \mathbf{a}) \to T\,\mathbf{m}\,(\mathbf{p} \otimes \mathbf{q})\,\mathbf{a}$$

```
class TriProFunctor t => Trimbara t where
  alpha :: t m dm p dp s ds ->
           t (m1, m) (dm1, dm) p dp (m1, s) (dm1, ds)
  beta  :: t m dm p dp (p1, s) (dp1, ds) ->
           t m dm (p, p1) (dp, dp1) s ds
```

Pre-optic as an Instance of Triple Tambara

- For any fixed $\mathbf{a}$, $T\,\mathbf{m}\,\mathbf{p}\,\mathbf{b} = (\mathbf{m}, \mathbf{p}, \mathbf{f} : \mathbf{m} \bullet \mathbf{a} \to \mathbf{p} \bullet \mathbf{b})$
- $\alpha : T\,\mathbf{m}\,\mathbf{p}\,\mathbf{b} \to T\,(\mathbf{n} \otimes \mathbf{m})\,\mathbf{p}\,(\mathbf{n} \bullet b)$
  - from the triple: $(\mathbf{m}, \mathbf{p}, \mathbf{f} : \mathbf{m} \bullet \mathbf{a} \to \mathbf{p} \bullet \mathbf{b})$
  - to the triple: $(\mathbf{n} \otimes \mathbf{m}, \mathbf{p}, \mathbf{f}' : (\mathbf{n} \otimes \mathbf{m}) \bullet \mathbf{a} \to \mathbf{p} \bullet (\mathbf{n} \bullet \mathbf{b}))$
  - use lifting of $\mathbf{f}$ by $(\mathbf{n} \bullet -)$
- $\beta : T\,\mathbf{m}\,\mathbf{p}\,(\mathbf{q} \bullet \mathbf{b}) \to T\,\mathbf{m}\,(\mathbf{p} \otimes \mathbf{q})\,\mathbf{b}$
  - from $(\mathbf{m}, \mathbf{p}, \mathbf{f} : \mathbf{m} \bullet \mathbf{a} \to \mathbf{p} \bullet (\mathbf{q} \bullet \mathbf{b}))$
  - to: $(\mathbf{m}, \mathbf{p} \otimes \mathbf{q}, \mathbf{f}' : \mathbf{m} \bullet \mathbf{a} \to (\mathbf{p} \otimes \mathbf{q}) \bullet \mathbf{b})$

Triple Tambara Representation of Optics

- A vanilla lens:

$$L\langle a, da\rangle\langle s, ds\rangle = \int^{m} \mathcal{C}(s, m \bullet a) \times \mathcal{C}(m \bullet da, ds)$$

- is equivalent to an end:

$$\int_{T\,:\,\mathbf{Tamb}} \mathbf{Set}\big(T\langle a, da\rangle, T\langle s, ds\rangle\big)$$

- A pre-optic:

$$(\mathbf{m}, \mathbf{p}, \mathbf{f} : \mathbf{m} \bullet \mathbf{a} \to \mathbf{p} \bullet \mathbf{b})$$

- is equivalent to a triple end:

$$\int_{\mathbf{q}\,:\,\mathbf{P}} \int_{\mathbf{n}\,:\,\mathbf{M}} \int_{T\,:\,\mathbf{TriTamb}} \mathbf{Set}\big(T\,\mathbf{n}\,\mathbf{q}\,\mathbf{a}, T\,(\mathbf{m} \otimes \mathbf{n})\,(\mathbf{q} \otimes \mathbf{p})\,\mathbf{b}\big)$$

TriLens in Haskell

$$\int_{\mathbf{q}\colon \mathbf{P}} \int_{\mathbf{n}\colon \mathbf{M}} \int_{T\colon \mathbf{TriTamb}} \mathbf{Set}\left(T\,\mathbf{n}\,\mathbf{q}\,\mathbf{a},\, T\,(\mathbf{m}\otimes\mathbf{n})\,(\mathbf{q}\otimes\mathbf{p})\,\mathbf{b}\right)$$

```
type TriLens a da m dm p dp s ds =
    forall t. Trimbara t => forall p1 dp1 m1 dm1.
      t m1 dm1 p1 dp1 a da ->
      t (m, m1) (dm, dm1) (p1, p) (dp1, dp) s ds
```

Pre-Optic to Triple Tambara

- Given an optic:

$$(\mathbf{m}, \mathbf{p}, \mathbf{f} : \mathbf{m} \bullet \mathbf{a} \to \mathbf{p} \bullet \mathbf{b})$$

- produce:

$$T\,\mathbf{n}\,\mathbf{q}\,\mathbf{a} \to T\,(\mathbf{m} \otimes \mathbf{n})\,(\mathbf{q} \otimes \mathbf{p})\,\mathbf{b}$$

- using composition:

$$T\,\mathbf{n}\,\mathbf{q}\,\mathbf{a} \xrightarrow{\alpha} T\,(\mathbf{m} \otimes \mathbf{n})\,\mathbf{q}\,(\mathbf{m} \bullet \mathbf{a})$$

$$\xrightarrow{T\,\mathbf{f}} T\,(\mathbf{m} \otimes \mathbf{n})\,\mathbf{q}\,(\mathbf{p} \bullet \mathbf{b})$$

$$\xrightarrow{\beta} T\,(\mathbf{m} \otimes \mathbf{n})\,(\mathbf{q} \otimes \mathbf{p})\,\mathbf{b}$$

Triple Tambara to Pre-Optic

- The other direction: Apply

$$T\,\mathbf{n}\,\mathbf{q}\,\mathbf{a} \to T\,(\mathbf{m} \otimes \mathbf{n})\,(\mathbf{q} \otimes \mathbf{p})\,\mathbf{b}$$

  to identity optic $(\mathbf{1}, \mathbf{1}, \mathbf{id}_{\mathbf{m} \bullet \mathbf{a}})$
- The result

$$T\,(\mathbf{m} \otimes \mathbf{1})\,(\mathbf{1} \otimes \mathbf{p})\,\mathbf{b} = (\mathbf{m}, \mathbf{p}, \mathbf{f} \colon \mathbf{m} \bullet \mathbf{a} \to \mathbf{p} \bullet \mathbf{b})$$

TriLens conversion to Pre-Optics

```
toTamb :: PreLens a da m dm p dp s ds ->
    TriLens a da m dm p dp s ds
toTamb (PreLens fw bw) = beta . dimapS fw bw . alpha
```

```
fromTamb :: TriLens a da m dm p dp s ds ->
    PreLens a da m dm p dp s ds
fromTamb f = dimapM runit unRunit $
             dimapP unLunit lunit $
             f idPreLens
```

TriLens Composition

Sequential composition:

```
triCompose ::
    TriLens b db m dm p dp s ds ->
    TriLens a da n dn q dq b db ->
    TriLens a da (m, n) (dm, dn) (q, p) (dq, dp) s ds
triCompose f g = dimapP unAssoc assoc .
                 dimapM unAssoc assoc .
                 f . g
```

Parallel composition similarly defined.

```
prodLensT :: TriLens a da m dm p  dp  s  ds ->
             TriLens a' da' m' dm' p' dp' s' ds' ->
             TriLens (a, a') (da, da') (m, m') (dm, dm')
                     (p, p') (dp, dp') (s, s') (ds, ds')
```

Linear Lens

```
type D = Double
type V = [D]
```

```
linearL :: Int -> PreLens D D (V, V) (V, V) V V V V
linearL n = PreLens fw bw
  where
    fw :: (V, V) -> ((V, V), D)
    fw (p, s) = ((s, p), sumN n (zipWith (*) p s))
    bw :: ((V, V), D) -> (V, V)
    bw ((s, p), da) = (fmap (da *) s
                      ,fmap (da *) p)
```

$$a = \sum_{i=1}^{n} p_i \times s_i \qquad \frac{\partial a}{\partial p_i} = s_i \qquad \frac{\partial a}{\partial s_i} = p_i$$

Neuron

```
linearT :: Int -> TriLens D D (V, V) (V, V) V V V V
linearT n = toTamb (linearL n)
```

We get a profunctor representation of a neuron by composing its three components:

```
neuronT :: Int ->
  TriLens D D ((V, V), D) ((V, V), D) Para Para V V
neuronT mIn =
  dimapP (second (unLunit . unPara))
         (second (mkPara . lunit)) .
  triCompose (dimapM (first runit) (first unRunit) .
  triCompose (linearT mIn) biasT) activT
```

```
data Para = Para { bias :: D, weight :: V }
```

Training a Multi-Layer Perceptron

```
makeMlp :: Int -> [Int] ->
  TriLens V V -- output
          [[((V, V), D)]] [[((V, V), D)]] -- residues
          [[Para]] [[Para]] -- parameters
          V V -- input
```

```
batchN :: (VSpace dp) => Int ->
    TriLens  a da m dm p dp s ds ->
    TriLens [a] [da] [m] [dm] p dp [s] [ds]
```

Loss Function

```
lossL :: PreLens D D ([V], [V]) ([V], [V]) [V] [V] [V] [V]
lossL = PreLens fw bw
  where
  fw (gTruth, s) =
    ((gTruth, s), sqDist (concat s) (concat gTruth))
  bw ((gTruth,s),da) = (fmap (fmap negate) delta', delta')
    where
    delta' = fmap (fmap (da *)) (zipWith minus s gTruth)
```

Bibliography

- Bartosz Milewski, https://github.com/BartoszMilewski/Publications/blob/master/NeuralLens.pdf
- Bartosz Milewski, The Dao of Functional Programming (available on github)
- Brendan Fong, Michael Johnson, Lenses and Learners.
- Brendan Fong, David Spivak, Rémy Tuyéras, Backprop as Functor: A compositional perspective on supervised learning, 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) 2019, pp. 1-13, 2019.
- G.S.H. Cruttwell, Bruno Gavranović, Neil Ghani, Paul Wilson, Fabio Zanasi, Categorical Foundations of Gradient-Based Learning.
- Bruno Gavranović, Compositional Deep Learning.
- Bruno Gavranović, Fundamental Components of Deep Learning, PhD Thesis. 2024.