Lenses and Learners (Fong and Johnson, [2019])

Overview and Discussion.
This paper presents a few connections between "lenses", a mathematical formulation of bidirectional transformations, and "learners", a generalization of compositional learning algorithms (such as neural networks) introduced in [2].

Tthe basic form of a lens is an asymmetric lens. For sets A, B an asymmetric lens $(p, g) : A \to B$ is a pair of functions put $p : B \times A \to A$ and get $g : A \to B$. These functions roughly mimic views or containment of data, and a lens is *well behaved* if *put* and *get* behave how we would expect for a database: $put(get(a), a) = a$ (**GetPut**) and $get(put(a, b)) = b$ (**PutGet**).

Asymmetric lenses are the arrows in a symmetric monoidal category where objects are sets and monoidal product is cartesian product. Well-behaved asymmetric lenses form a subcategory of this category. We can think of the composition of the lens $(p_1, g_1) : A \to B$ with $(p_2, g_2) : B \to C$ to represent A managing the view on (containing) B, which manages the view on (contains) C.

We can extend this construct to symmetric lenses, which are approximately spans of asymmetric lenses. That is, a symmetric lens $A \leftarrow S \to B$ is approximately the tuple of functions $(p_1, g_1, p_2, g_2) : A \to B$ where $(p_1, g_1) : S \to A$ and $(p_2, g_2) : S \to B$. Like asymmetric lenses, symmetric lenses[1] also form a symmetric monoidal category. The composition of symmetric lenses is equivalent to standard composition of spans (up to equivalence class). That is, the composition of symmetric lenses $A_1 \leftarrow S_1 \to A_2$ and $A_2 \leftarrow S_2 \to A_3$ is $A_1 \leftarrow T \to A_3$ where T is the apex of the pullback $S_1 \leftarrow T \to S_2$ of the cospan $S_1 \to A_2 \leftarrow S_2$. For example, T could be the commuting subset of the cartesian product of S_1 and S_2. We can use symmetric lenses to express more general kinds of composition. For example, we can use the apex to represent a notion of "shared context" that gets updated by each composition. This kind of composition preserves the property: "left left of the symmetric lens is well-behaved," which we can use to build the functor from learners to lenses.

The *learner*, introduced recently in [2], is a general formulation of a compositional learning algorithm. Learners are a set P, an implementation function $I : P \times A \to B$, an update function $U : B \times P \times A \to P$, and a request function $r : B \times P \times A \to A$. In the case of neural networks, P corresponds to a parameter space, U is gradient descent, and r is the input layer gradient. Like lenses, learners form a symmetric monoidal category with cartesian product as tensor product. In this category the composition of learners is done along the implementation function. This is akin to the "stacking" technique in machine learning, where the input to a downstream classifier is the output of an upstream classifier. This technique is a core part of neural networks. In [2], the authors proved that when P, A, B are $\mathbb{R}^{n_1}, \mathbb{R}^{n_2}, \mathbb{R}^{n_3}$, I is a differentiable function and U and r are particular functions of I's gradient, I is functorial. This raises the question: what are the most general conditions under which the learner's structure is "machine learning model-like" and I is functorial?

When P is a singleton, learners reduce to lenses with *get* I and *put* r. This is reminiscent of nonparametric machine learning models, which behave like databases with complicated *get* operations. We could break this down even further into nonparametric models that satisfy **PutGet** (e.g. K-Nearest Neighbors) and those that do not (e.g. Gaussian Processes).

We can define a learner as an asymmetric lens $(\langle U, r\rangle, I)$ where *get* is I and *put* is the product $\langle U, r\rangle$, but this does not lead to an easy notion of composition, since the parameters of a downstream learner will not be part of the output of an upstream learner. If we instead cast a learner as a symmetric lens with right leg $(\langle U, r\rangle, I)$ and left leg equal to the constant complement lens (a lens with convenient composition properties), then we can represent learner composition as symmetric lens composition. Essentially, the composition of the learners $A \leftarrow P_1 \times A \to B$ and $B \leftarrow P_2 \times B \to C$ will be $A \leftarrow P_1 \times P_2 \times A \to C$. That is, the apex of the span stores the parameters of the composite model as the cartesian product of the component models' parameters. This is a typical kind of symmetric lens composition: the "context" of the components, or parameters in this case, "merge" with a pullback (cartesian product in this case) when the lenses compose.

Upon the insight that we can cast all learners as symmetric lenses a natural question is: which kinds of learners, when turned into lenses, satisfy **PutGet** and **GetPut**? First, it's relatively easy to see that any learner for which the update function does not change the parameters for any correctly classified samples (zero loss samples) will satisfy **GetPut**. This includes most models without stochastic update functions, such as typical neural network models. Next, learners that satisfy **PutGet** will exhibit the property that after a training sample has been seen, the model will always classify it correctly. This is the class of models with infinite VC (vapnik-chervonenkis) dimension [3].

REFERENCES

[1] Fong, B. and Johnson, M. (2019). Lenses and learners. *CoRR*, abs/1903.03671.
[2] Fong, B., Spivak, D. I., and Tuyéras, R. (2017). Backprop as functor: A compositional perspective on supervised learning. *arXiv preprint arXiv:1711.10455*.
[3] Vapnik, V. N. and Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280.

[1] Technically equivalence classes of symmetric lenses