

Stephen's Work Log

Stephen Malina

February 6, 2020

Contents

1		2
1.1	Kipoi Dropout Research	2
2		3
2.1	DeepSea Uncertainty Estimates	3
3		5
3.1	Kipoi Uncertainty Estimates Cont.	5
3.2	GPU Memory Issues	5
4		7
4.1	Kipoi Uncertainty Estimates Cont.	7
5		9
5.1	Misc.	9

Minutes 1

Those present Stephen Malina

Date January 29, 2020

List of topics

1.1 Kipoi Dropout Research	2
--------------------------------------	---

1.1 Kipoi Dropout Research

My current understanding is that, assuming DeepSea is a PyTorch model in Kipoi (will verify shortly), I should be able to run a simple function like the following to activate Dropout in the Kipoi model (source 1, source 2).

```
def apply_dropout(m):  
    if type(m) == nn.Dropout:  
        m.train()
```

```
model.eval()
```

```
model.apply(apply_dropout)
```

This means that the next step is to figure out the best way to generate repeated predictions for the same sequence using Kipoi while ensuring that we use the same dropout mask on predictions for wild type and mutated sequences. Unfortunately, this most likely means we won't be able to use kipoi-interpret's out-of-box in-silico mutagenesis code.

Minutes 2

Those present Stephen Malina

Date January 30, 2020

List of topics

2.1	DeepSea Uncertainty Estimates	3
2.1.1	Verifying that we don't need precision for classification	3
2.1.2	Verifying that torch uses 1 Dropout mask for an entire mini-batch	4
2.1.3	Working with Kipoi to get uncertainty estimates . .	4

2.1 DeepSea Uncertainty Estimates

Huzzah! I'm able to force DeepSEA to apply Dropout at evaluation time using the strategy I mentioned yesterday. That said, one thing that's not clear to me is whether it's OK that not every layer is using Dropout... For now, I'm going to assume it is and flag it as something to follow up on.

Now, the next step is to figure out the best way to ensure that Kipoi uses the same Dropout mask for original and diff predictions.

2.1.1 Verifying that we don't need precision for classification

I found the appendix to Yarin Gal's "Dropout as a Bayesian Approximation" paper and I'm pretty confident that for classification that the predictive variance is just the empirical variance of the predicted probability.

One question that I'm putting aside for now is the question whether 0-normalizing is necessary.

2.1.2 Verifying that torch uses 1 Dropout mask for an entire mini-batch

So it's looking like torch does **not** use the same Dropout mask within the same mini-batch. This means either we:

1. Hot swap the torch dropout layers with custom ones which use 1 mask per mini-batch.
2. Use the standard dropout layers and accept that we'll lose the $\text{Cov}(X, Y)$ term from the

$$\text{Var}(X - Y) = \text{Var}(X) + \text{Var}(Y) - 2 \cdot \text{Cov}(X, Y)$$

formula for the variance of our predicted difference.

One caveat with option 1 is that, even if torch did use the same mask for the mini-batch, I'm not sure that we'll be able to fit all 4000 mutated sequences in the same batch. I *think* we will but will have to verify.

I think ultimately we should go with the first option, but I'm tempted to start with the second just to get all the scaffolding in place and then we can switch.

2.1.3 Working with Kipoi to get uncertainty estimates

The key question determining whether we can use the built-in mutagenesis code is whether Kipoi makes all the predictions as part of one batch. If it doesn't, that precludes us from ever going with option 2 and calls into question whether it makes sense to use the built-in mutagenesis code in the first place.

Minutes 3

Those present Stephen Malina

Date February 02, 2020

List of topics

3.1 Kipoi Uncertainty Estimates Cont.	5
3.1.1 Custom Dropout	5
3.1.2 In-silico mutagenesis with uncertainty	5
3.2 GPU Memory Issues	5

3.1 Kipoi Uncertainty Estimates Cont.

3.1.1 Custom Dropout

After fiddling around a bit with the code and reading some helpful PyTorch forum posts, I got custom Dropout working. I'm using a **barely** modified version of Salesforce's `LockedDropout` class code to do Dropout with a constant mask (within a batch). I'm using an also **barely** modified version of this `convert_layers` code to replace the original Dropout layers with my own.

3.1.2 In-silico mutagenesis with uncertainty

I now have code that can generate batches of (currently un-)mutated sequences and feed them to DeepSEA. Unfortunately, this is really slow...

3.2 GPU Memory Issues

I'm encountering weird behavior where Kipoi / Torch allocate a huge amount of memory (~10 GB) just to make predictions for a single 1000-record batch of data. Now, since each sequence is 1000 base pairs, I get that this is a non-trivial amount of data, but there's no way it requires 10 GB just to load and make

predictions on. There are two other pieces of evidence that convinced me that something else must be going on:

1. The 10 GB stick around after I make the predictions until I free the model
2. Calling `predict_on_example` doesn't cause the same issues even though

Managed to resolve these by just using small batch sizes.

Minutes 4

Those present Stephen Malina

Date February 04, 2020

List of topics

4.1	Kipoi Uncertainty Estimates Cont.	7
-----	---	---

4.1 Kipoi Uncertainty Estimates Cont.

Managed to get mean, variance, and covariance estimates using all the stuff I've been working on this past week. In the process, I made two graphs which seem worth capturing here.

The first depicts predictive variance versus predictive mean with colors representing different sequences (each dot is for a maybe mutated variant of the sequence). As expected, predictive variance increases as predictions become more uncertain (become closer to 0.5).

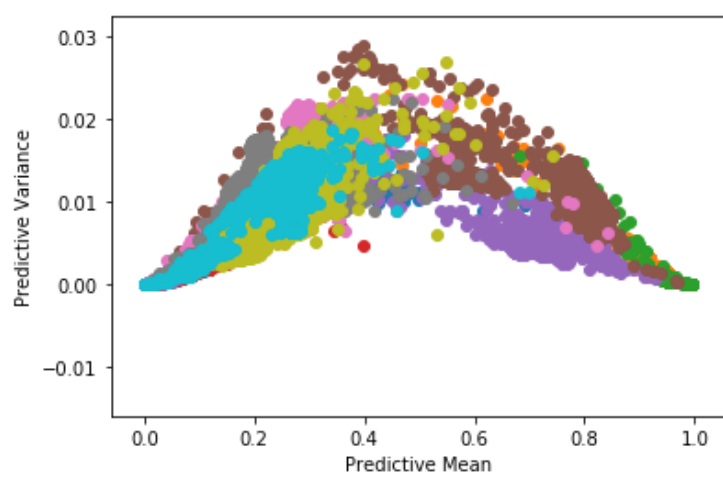


Figure 4.1: Plot of predictive mean vs. predictive variance. Colors code for different sequences.

Minutes 5

Those present Stephen Malina

Date February 06, 2020

List of topics

5.1	Misc.	9
-----	---------------	---

5.1 Misc.

I spent most of my work time today reading about MR-Egger while waiting for my run to finish. After that, I realized that I'd implicitly been leaving a bunch of un-mutated sequence predictions at the end of the final batch for each sequence in the data. Fixed that by making the batch size an even divisor of 3000. Altogether, not the most productive day...