Item 310
git_comments:

1. a dummy call just to test if the API works for merge_batches=True

git_commits:

1. **summary:** Merge pull request #1989 from pluskid/master
   message: Merge pull request #1989 from pluskid/master Fix module.predict

github_issues:

1. **title:** mxnet.ndarray.NDArray objects cannot be concatenated
   body: I had some problems after converting my network from `mxnet.model.FeedForward` to `mxnet.module.Module` when using the `predict()` method. With a module that has been trained, and a DataIterator `test_iter`, the following returns an error: ``` python mod.predict(test_iter) ``` The exception raised is: ``` --- ValueError Traceback (most recent call last) <ipython-input-134-73b4dc77d8cf> in <module>() ----> 1 model.predict(test_iter) python/mxnet/module/base_module.py in predict(self, eval_data, num_batch, merge_batches, reset, always_output_list) 259 'in mini-batches. Maybe bucketing is used?' 260 output_list2 = [np.concatenate([out[i] for out in output_list]) --> 261 for i in range(num_outputs)] ValueError: zero-dimensional arrays cannot be concatenated ``` [The code in question appears here] (https://github.com/dmlc/mxnet/blob/master/python/mxnet/module/base_module.py#L260) The problem seems to be due to trying to concatenate `mxnet.ndarray.NDArray` objects, which appear to have dimension 0. ``` python >>> pred = model.predict(test_iter, merge_batches=False) # This works >>> np.ndim(pred[0][0]) 0 # Checking that it's not due to end-of-batch-effects >>> len(pred) 79 >>> pred[0] [<mxnet.ndarray.NDArray at 0x1269f9240>] >>> np.concatenate([pred[0][0], pred[1][0]]) ... ValueError: zero-dimensional arrays cannot be concatenated ``` There is no problem when using `mxnet.model.FeedForward.predict()`, so I took a look to see what was different between the two `predict()` functions. In the `FeedForward` one, outputs are converted to numpy arrays before concatenation, which solves the problem pretty easily. [`mxnet.model.FeedForward.predict`] (https://github.com/dmlc/mxnet/blob/master/python/mxnet/model.py#L619) I will attach a minor change in lieu of rewriting the NDArray module to support concatenation.
2. **title:** mxnet.ndarray.NDArray objects cannot be concatenated
   body: I had some problems after converting my network from `mxnet.model.FeedForward` to `mxnet.module.Module` when using the `predict()` method. With a module that has been trained, and a DataIterator `test_iter`, the following returns an error: ``` python mod.predict(test_iter) ``` The exception raised is: ``` --- ValueError Traceback (most recent call last) <ipython-input-134-73b4dc77d8cf> in <module>() ----> 1 model.predict(test_iter) python/mxnet/module/base_module.py in predict(self, eval_data, num_batch, merge_batches, reset, always_output_list) 259 'in mini-batches. Maybe bucketing is used?' 260 output_list2 = [np.concatenate([out[i] for out in output_list]) --> 261 for i in range(num_outputs)] ValueError: zero-dimensional arrays cannot be concatenated ``` [The code in question appears here] (https://github.com/dmlc/mxnet/blob/master/python/mxnet/module/base_module.py#L260) The problem seems to be due to trying to concatenate `mxnet.ndarray.NDArray` objects, which appear to have dimension 0. ``` python >>> pred = model.predict(test_iter, merge_batches=False) # This works >>> np.ndim(pred[0][0]) 0 # Checking that it's not due to end-of-batch-effects >>> len(pred) 79 >>> pred[0] [<mxnet.ndarray.NDArray at 0x1269f9240>] >>> np.concatenate([pred[0][0], pred[1][0]]) ... ValueError: zero-dimensional arrays cannot be concatenated ``` There is no problem when using `mxnet.model.FeedForward.predict()`, so I took a look to see what was different between the two `predict()` functions. In the `FeedForward` one, outputs are converted to numpy arrays before concatenation, which solves the problem pretty easily. [`mxnet.model.FeedForward.predict`] (https://github.com/dmlc/mxnet/blob/master/python/mxnet/model.py#L619) I will attach a minor change in lieu of rewriting the NDArray module to support concatenation.
   label: code-design
3. **title:** mxnet.ndarray.NDArray objects cannot be concatenated
   body: I had some problems after converting my network from `mxnet.model.FeedForward` to `mxnet.module.Module` when using the `predict()` method. With a module that has been trained, and a DataIterator `test_iter`, the following returns an error: ``` python mod.predict(test_iter) ``` The exception

raised is: ``` --- ValueError Traceback (most recent call last) <ipython-input-134-73b4dc77d8cf> in <module>() ----> 1 model.predict(test_iter) python/mxnet/module/base_module.py in predict(self, eval_data, num_batch, merge_batches, reset, always_output_list) 259 'in mini-batches. Maybe bucketing is used?' 260 output_list2 = [np.concatenate([out[i] for out in output_list]) --> 261 for i in range(num_outputs)] ValueError: zero-dimensional arrays cannot be concatenated ``` [The code in question appears here] (https://github.com/dmlc/mxnet/blob/master/python/mxnet/module/base_module.py#L260) The problem seems to be due to trying to concatenate `mxnet.ndarray.NDArray` objects, which appear to have dimension 0. ``` python >>> pred = model.predict(test_iter, merge_batches=False) # This works >>> np.ndim(pred[0][0]) 0 # Checking that it's not due to end-of-batch-effects >>> len(pred) 79 >>> pred[0] [<mxnet.ndarray.NDArray at 0x1269f9240>] >>> np.concatenate([pred[0][0], pred[1][0]]) ... ValueError: zero-dimensional arrays cannot be concatenated ``` There is no problem when using `mxnet.model.FeedForward.predict()`, so I took a look to see what was different between the two `predict()` functions. In the `FeedForward` one, outputs are converted to numpy arrays before concatenation, which solves the problem pretty easily. [`mxnet.model.FeedForward.predict`] (https://github.com/dmlc/mxnet/blob/master/python/mxnet/model.py#L619) I will attach a minor change in lieu of rewriting the NDArray module to support concatenation.

4. **title:** mxnet.ndarray.NDArray objects cannot be concatenated
   body: I had some problems after converting my network from `mxnet.model.FeedForward` to `mxnet.module.Module` when using the `predict()` method. With a module that has been trained, and a DataIterator `test_iter`, the following returns an error: ``` python mod.predict(test_iter) ``` The exception raised is: ``` --- ValueError Traceback (most recent call last) <ipython-input-134-73b4dc77d8cf> in <module>() ----> 1 model.predict(test_iter) python/mxnet/module/base_module.py in predict(self, eval_data, num_batch, merge_batches, reset, always_output_list) 259 'in mini-batches. Maybe bucketing is used?' 260 output_list2 = [np.concatenate([out[i] for out in output_list]) --> 261 for i in range(num_outputs)] ValueError: zero-dimensional arrays cannot be concatenated ``` [The code in question appears here] (https://github.com/dmlc/mxnet/blob/master/python/mxnet/module/base_module.py#L260) The problem seems to be due to trying to concatenate `mxnet.ndarray.NDArray` objects, which appear to have dimension 0. ``` python >>> pred = model.predict(test_iter, merge_batches=False) # This works >>> np.ndim(pred[0][0]) 0 # Checking that it's not due to end-of-batch-effects >>> len(pred) 79 >>> pred[0] [<mxnet.ndarray.NDArray at 0x1269f9240>] >>> np.concatenate([pred[0][0], pred[1][0]]) ... ValueError: zero-dimensional arrays cannot be concatenated ``` There is no problem when using `mxnet.model.FeedForward.predict()`, so I took a look to see what was different between the two `predict()` functions. In the `FeedForward` one, outputs are converted to numpy arrays before concatenation, which solves the problem pretty easily. [`mxnet.model.FeedForward.predict`] (https://github.com/dmlc/mxnet/blob/master/python/mxnet/model.py#L619) I will attach a minor change in lieu of rewriting the NDArray module to support concatenation.
5. **title:** mxnet.ndarray.NDArray objects cannot be concatenated
   body: I had some problems after converting my network from `mxnet.model.FeedForward` to `mxnet.module.Module` when using the `predict()` method. With a module that has been trained, and a DataIterator `test_iter`, the following returns an error: ``` python mod.predict(test_iter) ``` The exception raised is: ``` --- ValueError Traceback (most recent call last) <ipython-input-134-73b4dc77d8cf> in <module>() ----> 1 model.predict(test_iter) python/mxnet/module/base_module.py in predict(self, eval_data, num_batch, merge_batches, reset, always_output_list) 259 'in mini-batches. Maybe bucketing is used?' 260 output_list2 = [np.concatenate([out[i] for out in output_list]) --> 261 for i in range(num_outputs)] ValueError: zero-dimensional arrays cannot be concatenated ``` [The code in question appears here] (https://github.com/dmlc/mxnet/blob/master/python/mxnet/module/base_module.py#L260) The problem seems to be due to trying to concatenate `mxnet.ndarray.NDArray` objects, which appear to have dimension 0. ``` python >>> pred = model.predict(test_iter, merge_batches=False) # This works >>> np.ndim(pred[0][0]) 0 # Checking that it's not due to end-of-batch-effects >>> len(pred) 79 >>> pred[0] [<mxnet.ndarray.NDArray at 0x1269f9240>] >>> np.concatenate([pred[0][0], pred[1][0]]) ... ValueError: zero-dimensional arrays cannot be concatenated ``` There is no problem when using `mxnet.model.FeedForward.predict()`, so I took a look to see what was different between the two `predict()` functions. In the `FeedForward` one, outputs are converted to numpy arrays before concatenation, which solves the problem pretty easily. [`mxnet.model.FeedForward.predict`] (https://github.com/dmlc/mxnet/blob/master/python/mxnet/model.py#L619) I will attach a minor change in lieu of rewriting the NDArray module to support concatenation.

github_issues_comments:

1. OK, I am unsure of how to get this to work since I already have a pull request pending, but there's some sort of issue with the linter so it hasn't been merged. So now they appear to be two commits on the same pull request, #1977 The change I suggested was for `mxnet/python/mxnet/module/base_module.py`, altering line 247: ``` - outputs = [out[0:out.shape[0]-pad] for out in self.get_outputs()] + outputs = [out[0:out.shape[0]-pad].asnumpy() for out in self.get_outputs()] ``` I can attest that it fixed my issues using `predict` and seemed like the easiest change that would ensure similar behavior.
2. **body:** @rldotai Thanks for spotting this bug! I proposed a different fix in #1989, which still keeps the `NDArray` as output types, but call `nd.concatenate` instead of `np.concatenate`. Returning `NDArray` allows us to better utilize the asynchronized execution engine in the backend and also a relatively consistent API. Please call `asnumpy()` explicitly on the returned results if you need to work with numpy. Thanks again!
   label: code-design
3. No worries, just glad to help.

github_pulls:

1. **title:** Fix module.predict
   body: See #1985 Also changed `fit` to reset eval metric on training data for each batch, to be compatible with the recent changes in `FeedForward`.
2. **title:** Fix module.predict
   body: See #1985 Also changed `fit` to reset eval metric on training data for each batch, to be compatible with the recent changes in `FeedForward`.
3. **title:** Fix module.predict
   body: See #1985 Also changed `fit` to reset eval metric on training data for each batch, to be compatible with the recent changes in `FeedForward`.
4. **title:** Fix module.predict
   body: See #1985 Also changed `fit` to reset eval metric on training data for each batch, to be compatible with the recent changes in `FeedForward`.
5. **title:** Fix module.predict
   body: See #1985 Also changed `fit` to reset eval metric on training data for each batch, to be compatible with the recent changes in `FeedForward`.
6. **title:** Fix module.predict
   body: See #1985 Also changed `fit` to reset eval metric on training data for each batch, to be compatible with the recent changes in `FeedForward`.
   label: code-design
7. **title:** Fix module.predict
   body: See #1985 Also changed `fit` to reset eval metric on training data for each batch, to be compatible with the recent changes in `FeedForward`.
8. **title:** Fix module.predict
   body: See #1985 Also changed `fit` to reset eval metric on training data for each batch, to be compatible with the recent changes in `FeedForward`.
   label: code-design
9. **title:** Fix module.predict
   body: See #1985 Also changed `fit` to reset eval metric on training data for each batch, to be compatible with the recent changes in `FeedForward`.
   label: code-design
10. **title:** Fix module.predict
    body: See #1985 Also changed `fit` to reset eval metric on training data for each batch, to be compatible with the recent changes in `FeedForward`.

github_pulls_comments:

1. let me know if you think it is good to merge things in. the r test seems to be broken for unknown reasons @thirdwing can you look into if some of the testcase break because recent updates in dependent libraries?
2. Looks like you removed the metric reset. I meant it should be reset whenever you print metric. Is that what it's doing now? Otherwise looks good to me

3. @piiswrong Yes, the validation on val set will reset the eval_metric. Otherwise, the eval_metric will also got reset at the beginning of each epoch. @tqchen can you merge it? Thanks!

github_pulls_reviews:

1. I think reset on each reporting is better
2. **body:** I agree with this change but this is not backward compatible. Maybe add a option to output ndarray and set false as default?
   label: code-design
3. OK agreed.
4. **body:** I think in this particular case it is better to keep a relatively unified interface than to keep it (exactly) backward compatible. Otherwise, it is also a question that whether you want to return np.array or ndarray for `iter_predict`. I think since the module API is already not exactly compatible with `FeedForward`, my personal opinion is to use this chance to have a cleaner interface here. I guess it might also perform better without those blocking `asnumpy()` call after each `forward`.
   label: code-design
5. **body:** Ok I didn't notice this is for module. Then I guess it's ok since there isn't much legacy code depending on this
   label: code-design

jira_issues:

jira_issues_comments: