tf.contrib.bayesflow.csiszar_divergence.csiszar_vimco

```
csiszar_vimco(
    f,
    p_log_prob,
    q,
    num_draws,
    num_batch_draws=1,
    seed=None,
    name=None
)
```

Defined in `tensorflow/contrib/bayesflow/python/ops/csiszar_divergence_impl.py`.

Use VIMCO to lower the variance of gradient[csiszar_function(Avg(logu))].

This function generalizes "Variational Inference for Monte Carlo Objectives" (VIMCO), i.e., https://arxiv.org/abs/1602.06725, to Csiszar f-Divergences.

> ★ **Note:** if `q.reparameterization_type = distribution.FULLY_REPARAMETERIZED`, consider using `monte_carlo_csiszar_f_divergence`.

The VIMCO loss is:

```
vimco = f(Avg{logu[i] : i=0,...,m-1})
where,
  logu[i] = log( p(x, h[i]) / q(h[i] | x) )
  h[i] iid~ q(H | x)
```

Interestingly, the VIMCO gradient is not the naive gradient of `vimco`. Rather, it is characterized by:

```
grad[vimco] - variance_reducing_term
where,
  variance_reducing_term = Sum{ grad[log q(h[i] | x)] *
                                  (vimco - f(log Avg{h[j;i] : j=0,...,m-1}))
                               : i=0, ..., m-1 }
  h[j;i] = { u[j]                              j!=i
           { GeometricAverage{ u[k] : k!=i}   j==i
```

(We omitted `stop_gradient` for brevity. See implementation for more details.)

The `Avg{h[j;i] : j}` term is a kind of "swap-out average" where the `i`-th element has been replaced by the leave-`i`-out Geometric-average.

This implementation prefers numerical precision over efficiency, i.e., `O(num_draws * num_batch_draws * prod(batch_shape) * prod(event_shape))`. (The constant may be fairly large, perhaps around 12.)

Args:

- `f`: Python `callable` representing a Csiszar-function in log-space.
- `p_log_prob`: Python `callable` representing the natural-log of the probability under distribution `p`. (In variational inference `p` is the joint distribution.)

- `q`: `tf.Distribution`-like instance; must implement: `sample(n, seed)`, and `log_prob(x)`. (In variational inference `q` is the approximate posterior distribution.)
- `num_draws`: Integer scalar number of draws used to approximate the f-Divergence expectation.
- `num_batch_draws`: Integer scalar number of draws used to approximate the f-Divergence expectation.
- `seed`: Python `int` seed for `q.sample`.
- `name`: Python `str` name prefixed to Ops created by this function.

Returns:

- `vimco`: The Csiszar f-Divergence generalized VIMCO objective.

Raises:

- `ValueError`: if `num_draws < 2`.

Last updated November 2, 2017.

Stay Connected

Blog

GitHub

Twitter

Support

Issue Tracker

Release Notes

Stack Overflow

English

Terms | **Privacy**