tf.contrib.bayesflow.entropy.renyi_ratio

```
renyi_ratio(
    log_p,
    q,
    alpha,
    z=None,
    n=None,
    seed=None,
    name='renyi_ratio'
)
```

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

See the guide: BayesFlow Entropy (contrib) > Ops

Monte Carlo estimate of the ratio appearing in Renyi divergence.

This can be used to compute the Renyi (alpha) divergence, or a log evidence approximation based on Renyi divergence.

Definition

With `z_i` iid samples from `q`, and `exp{log_p(z)} = p(z)`, this `Op` returns the (biased for finite `n`) estimate:

```
(1 - alpha)^{-1} Log[ n^{-1} sum_{i=1}^n ( p(z_i) / q(z_i) )^{1 - alpha},
\approx (1 - alpha)^{-1} Log[ E_q[ (p(Z) / q(Z))^{1 - alpha} ]  ]
```

This ratio appears in different contexts:

Renyi divergence

If `log_p(z) = Log[p(z)]` is the log prob of a distribution, and `alpha > 0`, `alpha != 1`, this `Op` approximates `-1` times Renyi divergence:

```
# Choose reasonably high n to limit bias, see below.
renyi_ratio(log_p, q, alpha, n=100)
                \approx -1 * D_alpha[q || p],  where
D_alpha[q || p] := (1 - alpha)^{-1} Log E_q[(p(Z) / q(Z))^{1 - alpha}]
```

The Renyi (or "alpha") divergence is non-negative and equal to zero iff `q = p`. Various limits of `alpha` lead to different special case results:

```
alpha       D_alpha[q || p]
-----       ---------------
--> 0       Log[ int_{q > 0} p(z) dz ]
= 0.5,      -2 Log[1 - Hel^2[q || p]],  (\propto squared Hellinger distance)
--> 1       KL[q || p]
= 2         Log[ 1 + chi^2[q || p] ],   (\propto squared Chi-2 divergence)
--> infty   Log[ max_z{q(z) / p(z)} ],  (min description length principle).
```

See "Renyi Divergence Variational Inference", by Li and Turner.

Log evidence approximation

If `log_p(z) = Log[p(z, x)]` is the log of the joint distribution `p`, this is an alternative to the ELBO common in variational inference.

```
L_alpha(q, p) = Log[p(x)] - D_alpha[q || p]
```

If `q` and `p` have the same support, and `0 < a <= b < 1`, one can show `ELBO <= D_b <= D_a <= Log[p(x)]`. Thus, this `Op` allows a smooth interpolation between the ELBO and the true evidence.

Stability notes

Note that when `1 - alpha` is not small, the ratio `(p(z) / q(z))^{1 - alpha}` is subject to underflow/overflow issues. For that reason, it is evaluated in log-space after centering. Nonetheless, infinite/NaN results may occur. For that reason, one may wish to shrink `alpha` gradually. See the `Op` `renyi_alpha`. Using `float64` will also help.

Bias for finite sample size

Due to nonlinearity of the logarithm, for random variables `{X_1,...,X_n}`, `E[ Log[sum_{i=1}^n X_i] ] != Log[ E[sum_{i=1}^n X_i] ]`. As a result, this estimate is biased for finite `n`. For `alpha < 1`, it is non-decreasing with `n` (in expectation). For example, if `n = 1`, this estimator yields the same result as `elbo_ratio`, and as `n` increases the expected value of the estimator increases.

Call signature

User supplies either `Tensor` of samples `z`, or number of samples to draw `n`

Args:

- `log_p`: Callable mapping samples from `q` to `Tensors` with shape broadcastable to `q.batch_shape`. For example, `log_p` works "just like" `q.log_prob`.
- `q`: `tf.contrib.distributions.Distribution`. `float64` `dtype` recommended. `log_p` and `q` should be supported on the same set.
- `alpha`: `Tensor` with shape `q.batch_shape` and values not equal to 1.
- `z`: `Tensor` of samples from `q`, produced by `q.sample` for some `n`.
- `n`: Integer `Tensor`. The number of samples to use if `z` is not provided. Note that this can be highly biased for small `n`, see docstring.
- `seed`: Python integer to seed the random number generator.
- `name`: A name to give this `Op`.

Returns:

- `renyi_result`: The scaled log of sample mean. `Tensor` with `shape` equal to batch shape of `q`, and `dtype` = `q.dtype`.

Last updated November 2, 2017.