tf.contrib.nn.rank_sampled_softmax_loss

```
rank_sampled_softmax_loss(
    weights,
    biases,
    labels,
    inputs,
    num_sampled,
    num_resampled,
    num_classes,
    num_true,
    sampled_values,
    resampling_temperature,
    remove_accidental_hits,
    partition_strategy,
    name=None
)
```

Defined in `tensorflow/contrib/nn/python/ops/sampling_ops.py`.

Computes softmax loss using rank-based adaptive resampling.

This has been shown to improve rank loss after training compared to `tf.nn.sampled_softmax_loss`. For a description of the algorithm and some experimental results, please see: TAPAS: Two-pass Approximate Adaptive Sampling for Softmax.

Sampling follows two phases: *In the first phase,* `num_sampled` *classes are selected using* `tf.nn.learned_unigram_candidate_sampler` *or supplied* `sampled_values`. *The logits are calculated on those sampled classes. This phases is similar to* `tf.nn.sampled_softmax_loss`. In the second phase, the `num_resampled` classes with highest predicted probability are kept. Probabilities are `LogSumExp(logits / resampling_temperature)`, where the sum is over `inputs`.

The `resampling_temperature` parameter controls the "adaptiveness" of the resampling. At lower temperatures, resampling is more adaptive because it picks more candidates close to the predicted classes. A common strategy is to decrease the temperature as training proceeds.

See `tf.nn.sampled_softmax_loss` for more documentation on sampling and for typical default values for some of the parameters.

This operation is for training only. It is generally an underestimate of the full softmax loss.

A common use case is to use this method for training, and calculate the full softmax loss for evaluation or inference. In this case, you must set `partition_strategy="div"` for the two losses to be consistent, as in the following example:

```
if mode == "train":
  loss = rank_sampled_softmax_loss(
      weights=weights,
      biases=biases,
      labels=labels,
      inputs=inputs,
      ...,
      partition_strategy="div")
elif mode == "eval":
  logits = tf.matmul(inputs, tf.transpose(weights))
  logits = tf.nn.bias_add(logits, biases)
  labels_one_hot = tf.one_hot(labels, n_classes)
  loss = tf.nn.softmax_cross_entropy_with_logits(
      labels=labels_one_hot,
      logits=logits)
```

Args:

- `weights`: A `Tensor` or `PartitionedVariable` of shape `[num_classes, dim]`, or a list of `Tensor` objects whose concatenation along dimension 0 has shape [num_classes, dim]. The (possibly-sharded) class embeddings.
- `biases`: A `Tensor` or `PartitionedVariable` of shape `[num_classes]`. The (possibly-sharded) class biases.
- `labels`: A `Tensor` of type `int64` and shape `[batch_size, num_true]`. The target classes. Note that this format differs from the `labels` argument of `nn.softmax_cross_entropy_with_logits`.
- `inputs`: A `Tensor` of shape `[batch_size, dim]`. The forward activations of the input network.
- `num_sampled`: An `int`. The number of classes to randomly sample per batch.
- `num_resampled`: An `int`. The number of classes to select from the `num_sampled` classes using the adaptive resampling algorithm. Must be less than `num_sampled`.
- `num_classes`: An `int`. The number of possible classes.
- `num_true`: An `int`. The number of target classes per training example.
- `sampled_values`: A tuple of (`sampled_candidates`, `true_expected_count`, `sampled_expected_count`) returned by a `*_candidate_sampler` function. If None, default to `nn.learned_unigram_candidate_sampler`.
- `resampling_temperature`: A scalar `Tensor` with the temperature parameter for the adaptive resampling algorithm.
- `remove_accidental_hits`: A `bool`. Whether to remove "accidental hits" where a sampled class equals one of the target classes.
- `partition_strategy`: A string specifying the partitioning strategy, relevant if `len(weights) > 1`. Currently `"div"` and `"mod"` are supported. See `tf.nn.embedding_lookup` for more details.
- `name`: A name for the operation (optional).

Returns:

A `batch_size` 1-D tensor of per-example sampled softmax losses.

Raises:

- `ValueError`: If `num_sampled <= num_resampled`.

Stay Connected

Blog

GitHub

Twitter

Support

Issue Tracker

Release Notes

Stack Overflow

English

Terms | **Privacy**