tf.SparseConditionalAccumulator

Contents

Class **`SparseConditionalAccumulator`**

Inherits From: `ConditionalAccumulatorBase`

Defined in `tensorflow/python/ops/data_flow_ops.py`.

See the guide: Inputs and Readers > Conditional Accumulators

A conditional accumulator for aggregating sparse gradients.

Sparse gradients are represented by IndexedSlices.

Up-to-date gradients (i.e., time step at which gradient was computed is equal to the accumulator's time step) are added to the accumulator.

Extraction of the average gradient is blocked until the required number of gradients has been accumulated.

Args:

- `dtype`: Datatype of the accumulated gradients.
- `shape`: Shape of the accumulated gradients.
- `shared_name`: Optional. If non-empty, this accumulator will be shared under the given name across multiple sessions.
- `name`: Optional name for the accumulator.

Properties

`accumulator_ref`

The underlying accumulator reference.

`dtype`

The datatype of the gradients accumulated by this accumulator.

`name`

The name of the underlying accumulator.

Methods

`__init__`

```
__init__(
    dtype,
    shape=None,
    shared_name=None,
    name='sparse_conditional_accumulator'
)
```

`apply_grad`

```
apply_grad(
    grad_indices,
    grad_values,
    grad_shape=None,
    local_step=0,
    name=None
)
```

Attempts to apply a sparse gradient to the accumulator.

The attempt is silently dropped if the gradient is stale, i.e., local_step is less than the accumulator's global time step.

A sparse gradient is represented by its indices, values and possibly empty or None shape. Indices must be a vector representing the locations of non-zero entries in the tensor. Values are the non-zero slices of the gradient, and must have the same first dimension as indices, i.e., the nnz represented by indices and values must be consistent. Shape, if not empty or None, must be consistent with the accumulator's shape (if also provided).

Example: A tensor [[0, 0], [0. 1], [2, 3]] can be represented indices: [1,2] values: [[0,1],[2,3]] shape: [3, 2]

Args:

- `grad_indices`: Indices of the sparse gradient to be applied.
- `grad_values`: Values of the sparse gradient to be applied.
- `grad_shape`: Shape of the sparse gradient to be applied.
- `local_step`: Time step at which the gradient was computed.
- `name`: Optional name for the operation.

Returns:

The operation that (conditionally) applies a gradient to the accumulator.

Raises:

- `InvalidArgumentError`: If grad is of the wrong shape

`apply_indexed_slices_grad`

```
apply_indexed_slices_grad(
    grad,
    local_step=0,
    name=None
)
```

Attempts to apply a gradient to the accumulator.

The attempt is silently dropped if the gradient is stale, i.e., local_step is less than the accumulator's global time step.

Args:

- `grad` : The gradient IndexedSlices to be applied.
- `local_step` : Time step at which the gradient was computed.
- `name` : Optional name for the operation.

Returns:

The operation that (conditionally) applies a gradient to the accumulator.

Raises:

- `InvalidArgumentError` : If grad is of the wrong shape

`num_accumulated`

```
num_accumulated(name=None)
```

Number of gradients that have currently been aggregated in accumulator.

Args:

- `name` : Optional name for the operation.

Returns:

Number of accumulated gradients currently in accumulator.

`set_global_step`

```
set_global_step(
    new_global_step,
    name=None
)
```

Sets the global time step of the accumulator.

The operation logs a warning if we attempt to set to a time step that is lower than the accumulator's own time step.

Args:

- `new_global_step` : Value of new time step. Can be a variable or a constant

- `name`: Optional name for the operation.

Returns:

Operation that sets the accumulator's time step.

`take_grad`

```
take_grad(
    num_required,
    name=None
)
```

Attempts to extract the average gradient from the accumulator.

The operation blocks until sufficient number of gradients have been successfully applied to the accumulator.

Once successful, the following actions are also triggered: - Counter of accumulated gradients is reset to 0. - Aggregated gradient is reset to 0 tensor. - Accumulator's internal time step is incremented by 1.

Args:

- `num_required`: Number of gradients that needs to have been aggregated
- `name`: Optional name for the operation

Returns:

A tuple of indices, values, and shape representing the average gradient.

Raises:

- `InvalidArgumentError`: If num_required < 1

`take_indexed_slices_grad`

```
take_indexed_slices_grad(
    num_required,
    name=None
)
```

Attempts to extract the average gradient from the accumulator.

The operation blocks until sufficient number of gradients have been successfully applied to the accumulator.

Once successful, the following actions are also triggered: - Counter of accumulated gradients is reset to 0. - Aggregated gradient is reset to 0 tensor. - Accumulator's internal time step is incremented by 1.

Args:

- `num_required`: Number of gradients that needs to have been aggregated
- `name`: Optional name for the operation

Returns:

An IndexedSlices holding the value of the average gradient.

Raises:

- `InvalidArgumentError` : If num_required < 1