tf.train.GradientDescentOptimizer

Class **`GradientDescentOptimizer`**

Inherits From: `Optimizer`

Defined in `tensorflow/python/training/gradient_descent.py`.

See the guide: Training > Optimizers

Optimizer that implements the gradient descent algorithm.

Methods

`__init__`

```
__init__(
    learning_rate,
    use_locking=False,
    name='GradientDescent'
)
```

Construct a new gradient descent optimizer.

Args:

- `learning_rate`: A Tensor or a floating point value. The learning rate to use.
- `use_locking`: If True use locks for update operations.
- `name`: Optional name prefix for the operations created when applying gradients. Defaults to "GradientDescent".

`apply_gradients`

```
apply_gradients(
    grads_and_vars,
    global_step=None,
    name=None
)
```

Apply gradients to variables.

This is the second part of `minimize()`. It returns an `Operation` that applies gradients.

Args:

- `grads_and_vars`: List of (gradient, variable) pairs as returned by `compute_gradients()`.
- `global_step`: Optional `Variable` to increment by one after the variables have been updated.
- `name`: Optional name for the returned operation. Default to the name passed to the `Optimizer` constructor.

Returns:

An `Operation` that applies the specified gradients. If `global_step` was not None, that operation also increments `global_step`.

Raises:

- `TypeError`: If `grads_and_vars` is malformed.
- `ValueError`: If none of the variables have gradients.

`compute_gradients`

```
compute_gradients(
    loss,
    var_list=None,
    gate_gradients=GATE_OP,
    aggregation_method=None,
    colocate_gradients_with_ops=False,
    grad_loss=None
)
```

Compute gradients of `loss` for the variables in `var_list`.

This is the first part of `minimize()`. It returns a list of (gradient, variable) pairs where "gradient" is the gradient for "variable". Note that "gradient" can be a `Tensor`, an `IndexedSlices`, or `None` if there is no gradient for the given variable.

Args:

- `loss`: A Tensor containing the value to minimize.
- `var_list`: Optional list or tuple of `tf.Variable` to update to minimize `loss`. Defaults to the list of variables collected in the graph under the key `GraphKey.TRAINABLE_VARIABLES`.
- `gate_gradients`: How to gate the computation of gradients. Can be `GATE_NONE`, `GATE_OP`, or `GATE_GRAPH`.
- `aggregation_method`: Specifies the method used to combine gradient terms. Valid values are defined in the class `AggregationMethod`.
- `colocate_gradients_with_ops`: If True, try colocating gradients with the corresponding op.
- `grad_loss`: Optional. A `Tensor` holding the gradient computed for `loss`.

Returns:

A list of (gradient, variable) pairs. Variable is always present, but gradient can be `None`.

Raises:

- `TypeError` : If `var_list` contains anything else than `Variable` objects.
- `ValueError` : If some arguments are invalid.

`get_name`

```
get_name()
```

`get_slot`

```
get_slot(
    var,
    name
)
```

Return a slot named `name` created for `var` by the Optimizer.

Some `Optimizer` subclasses use additional variables. For example `Momentum` and `Adagrad` use variables to accumulate updates. This method gives access to these `Variable` objects if for some reason you need them.

Use `get_slot_names()` to get the list of slot names created by the `Optimizer` .

Args:

- `var` : A variable passed to `minimize()` or `apply_gradients()` .
- `name` : A string.

Returns:

The `Variable` for the slot if it was created, `None` otherwise.

`get_slot_names`

```
get_slot_names()
```

Return a list of the names of slots created by the `Optimizer` .

See `get_slot()` .

Returns:

A list of strings.

`minimize`

```
minimize(
    loss,
    global_step=None,
    var_list=None,
    gate_gradients=GATE_OP,
    aggregation_method=None,
    colocate_gradients_with_ops=False,
    name=None,
    grad_loss=None
)
```

Add operations to minimize `loss` by updating `var_list`.

This method simply combines calls `compute_gradients()` and `apply_gradients()`. If you want to process the gradient before applying them call `compute_gradients()` and `apply_gradients()` explicitly instead of using this function.

Args:

- `loss`: A `Tensor` containing the value to minimize.
- `global_step`: Optional `Variable` to increment by one after the variables have been updated.
- `var_list`: Optional list or tuple of `Variable` objects to update to minimize `loss`. Defaults to the list of variables collected in the graph under the key `GraphKeys.TRAINABLE_VARIABLES`.
- `gate_gradients`: How to gate the computation of gradients. Can be `GATE_NONE`, `GATE_OP`, or `GATE_GRAPH`.
- `aggregation_method`: Specifies the method used to combine gradient terms. Valid values are defined in the class `AggregationMethod`.
- `colocate_gradients_with_ops`: If True, try colocating gradients with the corresponding op.
- `name`: Optional name for the returned operation.
- `grad_loss`: Optional. A `Tensor` holding the gradient computed for `loss`.

Returns:

An Operation that updates the variables in `var_list`. If `global_step` was not `None`, that operation also increments `global_step`.

Raises:

- `ValueError`: If some of the variables are not `Variable` objects.

Class Members

GATE_GRAPH

GATE_NONE

GATE_OP