# tf.gradients

```
gradients(
    ys,
    xs,
    grad_ys=None,
    name='gradients',
    colocate_gradients_with_ops=False,
    gate_gradients=False,
    aggregation_method=None,
    stop_gradients=None
)
```

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

See the guide: Training > Gradient Computation

Constructs symbolic derivatives of sum of `ys` w.r.t. x in `xs`.

`ys` and `xs` are each a `Tensor` or a list of tensors. `grad_ys` is a list of `Tensor`, holding the gradients received by the `ys`. The list must be the same length as `ys`.

`gradients()` adds ops to the graph to output the derivatives of `ys` with respect to `xs`. It returns a list of `Tensor` of length `len(xs)` where each tensor is the `sum(dy/dx)` for y in `ys`.

`grad_ys` is a list of tensors of the same length as `ys` that holds the initial gradients for each y in `ys`. When `grad_ys` is None, we fill in a tensor of '1's of the shape of y for each y in `ys`. A user can provide their own initial `grad_ys` to compute the derivatives using a different initial gradient for each y (e.g., if one wanted to weight the gradient differently for each value in each y).

`stop_gradients` is a `Tensor` or a list of tensors to be considered constant with respect to all `xs`. These tensors will not be backpropagated through, as though they had been explicitly disconnected using `stop_gradient`. Among other things, this allows computation of partial derivatives as opposed to total derivatives. For example:

a = tf.constant(0.) b = 2 * a g = tf.gradients(a + b, [a, b], stop_gradients=[a, b])

Here the partial derivatives `g` evaluate to `[1.0, 1.0]`, compared to the total derivatives `tf.gradients(a + b, [a, b])`, which take into account the influence of `a` on `b` and evaluate to `[3.0, 1.0]`. Note that the above is equivalent to:

a = tf.stop_gradient(tf.constant(0.)) b = tf.stop_gradient(2 * a) g = tf.gradients(a + b, [a, b])

`stop_gradients` provides a way of stopping gradient after the graph has already been constructed, as compared to `tf.stop_gradient` which is used during graph construction. When the two approaches are combined, backpropagation stops at both `tf.stop_gradient` nodes and nodes in `stop_gradients`, whichever is encountered first.

## Args:

- `ys`: A `Tensor` or list of tensors to be differentiated.
- `xs`: A `Tensor` or list of tensors to be used for differentiation.
- `grad_ys`: Optional. A `Tensor` or list of tensors the same size as `ys` and holding the gradients computed for each y in `ys`.
- `name`: Optional name to use for grouping all the gradient ops together. defaults to 'gradients'.

- `colocate_gradients_with_ops`: If True, try colocating gradients with the corresponding op.
- `gate_gradients`: If True, add a tuple around the gradients returned for an operations. This avoids some race conditions.
- `aggregation_method`: Specifies the method used to combine gradient terms. Accepted values are constants defined in the class `AggregationMethod`.
- `stop_gradients`: Optional. A `Tensor` or list of tensors not to differentiate through.

## Returns:

A list of `sum(dy/dx)` for each x in `xs`.

## Raises:

- `LookupError`: if one of the operations between `x` and `y` does not have a registered gradient function.
- `ValueError`: if the arguments are invalid.
- `RuntimeError`: if called in Eager mode.

---



*Last updated November 2, 2017.*