tf.contrib.kfac.optimizer.KfacOptimizer

Class `KfacOptimizer`

Inherits From: `GradientDescentOptimizer`

Defined in `tensorflow/contrib/kfac/python/ops/optimizer.py`.

The KFAC Optimizer (https://arxiv.org/abs/1503.05671).

Properties

`damping`

`variables`

Methods

`__init__`

```
__init__(
    learning_rate,
    cov_ema_decay,
    damping,
    layer_collection,
    momentum=0.0,
    momentum_type='regular',
    norm_constraint=None,
    name='KFAC'
)
```

Initializes the KFAC optimizer with the given settings.

Args:

- `learning_rate`: The base learning rate for the optimizer. Should probably be set to 1.0 when using momentum_type = 'qmodel', but can still be set lowered if desired (effectively lowering the trust in the quadratic model.)
- `cov_ema_decay`: The decay factor used when calculating the covariance estimate moving averages.
- `damping`: The damping factor used to stabilize training due to errors in the local approximation with the Fisher information matrix, and to regularize the update direction by making it closer to the gradient. (Higher damping means

the update looks more like a standard gradient update - see Tikhonov regularization.)

- `layer_collection`: The layer collection object, which holds the fisher blocks, kronecker factors, and losses associated with the graph. The layer_collection cannot be modified after KfacOptimizer's initialization.
- `momentum`: The momentum value for this optimizer. Only applies when momentum_type is 'regular' or 'adam'. (Default: 0)
- `momentum_type`: The type of momentum to use in this optimizer, one of 'regular', 'adam', or 'qmodel'. (Default: 'regular')
- `norm_constraint`: float or Tensor. If specified, the update is scaled down so that its approximate squared Fisher norm v^T F v is at most the specified value. May only be used with momentum type 'regular'. (Default: None)
- `name`: The name for this optimizer. (Default: 'KFAC')

Raises:

- `ValueError`: If the momentum type is unsupported.
- `ValueError`: If clipping is used with momentum type other than 'regular'.
- `ValueError`: If no losses have been registered with layer_collection.
- `ValueError`: If momentum is non-zero and momentum_type is not 'regular' or 'adam'.

`apply_gradients`

```
apply_gradients(
    grads_and_vars,
    *args,
    **kwargs
)
```

Applies gradients to variables.

Args:

- `grads_and_vars`: List of (gradient, variable) pairs.
- `*args`: Additional arguments for super.apply_gradients.
- `**kwargs`: Additional keyword arguments for super.apply_gradients.

Returns:

An `Operation` that applies the specified 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(
    *args,
    **kwargs
)
```

Class Members

GATE_GRAPH

GATE_NONE

GATE_OP

Last updated November 2, 2017.