tf.contrib.kfac.loss_functions.LossFunction

Contents

Class **LossFunction**

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

Abstract base class for loss functions.

Note that unlike typical loss functions used in neural networks these are summed and not averaged across cases in the batch, since this is what the users of this class (FisherEstimator and MatrixVectorProductComputer) will be expecting. The implication of this is that you will may want to normalize things like Fisher-vector products by the batch size when you use this class. It depends on the use case.

Properties

`hessian_factor_inner_shape`

The shape of the tensor returned by multiply_hessian_factor.

`hessian_factor_inner_static_shape`

Static version of hessian_factor_inner_shape.

`inputs`

The inputs to the loss function (excluding the targets).

Methods

`__init__`

```
__init__(targets=None)
```

`evaluate`

```
evaluate()
```

Evaluate the loss function.

`multiply_hessian`

```
multiply_hessian(vector)
```

Right-multiply a vector by the Hessian.

Here the 'Hessian' is the Hessian matrix (i.e. matrix of 2nd-derivatives) of the loss function with respect to its inputs.

Args:

- `vector`: The vector to multiply. Must be the same shape(s) as the 'inputs' property.

Returns:

The vector right-multiplied by the Hessian. Will be of the same shape(s) as the 'inputs' property.

`multiply_hessian_factor`

```
multiply_hessian_factor(vector)
```

Right-multiply a vector by a factor B of the Hessian.

Here the 'Hessian' is the Hessian matrix (i.e. matrix of 2nd-derivatives) of the loss function with respect to its inputs. Typically this will be block-diagonal across different cases in the batch, since the loss function is typically summed across cases.

Note that B can be any matrix satisfying B * B^T = H where H is the Hessian, but will agree with the one used in the other methods of this class.

Args:

- `vector`: The vector to multiply. Must be of the shape given by the 'hessian_factor_inner_shape' property.

Returns:

The vector right-multiplied by B. Will be of the same shape(s) as the 'inputs' property.

`multiply_hessian_factor_replicated_one_hot`

```
multiply_hessian_factor_replicated_one_hot(index)
```

Right-multiply a replicated-one-hot vector by a factor B of the Hessian.

Here the 'Hessian' is the Hessian matrix (i.e. matrix of 2nd-derivatives) of the loss function with respect to its inputs. Typically this will be block-diagonal across different cases in the batch, since the loss function is typically summed across cases.

A 'replicated-one-hot' vector means a tensor which, for each slice along the batch dimension (assumed to be dimension 0), is 1.0 in the entry corresponding to the given index and 0 elsewhere.

Note that B can be any matrix satisfying B * B^T = H where H is the Hessian, but will agree with the one used in the other

methods of this class.

Args:

- `index`: A tuple representing in the index of the entry in each slice that is 1.0. Note that len(index) must be equal to the number of elements of the 'hessian_factor_inner_shape' tensor minus one.

Returns:

The vector right-multiplied by B^T. Will be of the same shape(s) as the 'inputs' property.

`multiply_hessian_factor_transpose`

```
multiply_hessian_factor_transpose(vector)
```

Right-multiply a vector by the tranpose of a factor B of the Hessian.

Here the 'Hessian' is the Hessian matrix (i.e. matrix of 2nd-derivatives) of the loss function with respect to its inputs. Typically this will be block-diagonal across different cases in the batch, since the loss function is typically summed across cases.

Note that B can be any matrix satisfying B * B^T = H where H is the Hessian, but will agree with the one used in the other methods of this class.

Args:

- `vector`: The vector to multiply. Must be the same shape(s) as the 'inputs' property.

Returns:

The vector right-multiplied by B^T. Will be of the shape given by the 'hessian_factor_inner_shape' property.
