tf.contrib.opt.ExternalOptimizerInterface

Contents

Class **`ExternalOptimizerInterface`**

Defined in `tensorflow/contrib/opt/python/training/external_optimizer.py`.

Base class for interfaces with external optimization algorithms.

Subclass this and implement `_minimize` in order to wrap a new optimization algorithm.

`ExternalOptimizerInterface` should not be instantiated directly; instead use e.g. `ScipyOptimizerInterface`.

Methods

`__init__`

```
__init__(
    loss,
    var_list=None,
    equalities=None,
    inequalities=None,
    var_to_bounds=None,
    **optimizer_kwargs
)
```

Initialize a new interface instance.

Args:

- `loss`: A scalar `Tensor` to be minimized.
- `var_list`: Optional `list` of `Variable` objects to update to minimize `loss`. Defaults to the list of variables collected in the graph under the key `GraphKeys.TRAINABLE_VARIABLES`.
- `equalities`: Optional `list` of equality constraint scalar `Tensor` s to be held equal to zero.
- `inequalities`: Optional `list` of inequality constraint scalar `Tensor` s to be held nonnegative.
- `var_to_bounds`: Optional `dict` where each key is an optimization `Variable` and each corresponding value is a length-2 tuple of `(low, high)` bounds. Although enforcing this kind of simple constraint could be accomplished with the `inequalities` arg, not all optimization algorithms support general inequality constraints, e.g. L-BFGS-B. Both `low` and `high` can either be numbers or anything convertible to a NumPy array that can be broadcast to the shape of `var` (using `np.broadcast_to`). To indicate that there is no bound, use `None` (or

`+/- np.infty`). For example, if `var` is a 2x3 matrix, then any of the following corresponding `bounds` could be supplied:

  - `(0, np.infty)`: Each element of `var` held positive.
  - `(-np.infty, [1, 2])`: First column less than 1, second column less than 2.
  - `(-np.infty, [[1], [2], [3]])`: First row less than 1, second row less than 2, etc.
  - `(-np.infty, [[1, 2, 3], [4, 5, 6]])`: Entry `var[0, 0]` less than 1, `var[0, 1]` less than 2, etc.
- `**optimizer_kwargs`: Other subclass-specific keyword arguments.

minimize

```
minimize(
    session=None,
    feed_dict=None,
    fetches=None,
    step_callback=None,
    loss_callback=None,
    **run_kwargs
)
```

Minimize a scalar `Tensor`.

Variables subject to optimization are updated in-place at the end of optimization.

Note that this method does *not* just return a minimization `Op`, unlike `Optimizer.minimize()`; instead it actually performs minimization by executing commands to control a `Session`.

Args:

- `session`: A `Session` instance.
- `feed_dict`: A feed dict to be passed to calls to `session.run`.
- `fetches`: A list of `Tensor`s to fetch and supply to `loss_callback` as positional arguments.
- `step_callback`: A function to be called at each optimization step; arguments are the current values of all optimization variables flattened into a single vector.
- `loss_callback`: A function to be called every time the loss and gradients are computed, with evaluated fetches supplied as positional arguments.
- `**run_kwargs`: kwargs to pass to `session.run`.

Last updated November 2, 2017.

Stay Connected

Blog

GitHub

Twitter

Support

Issue Tracker

Release Notes

Stack Overflow

English

Terms | Privacy