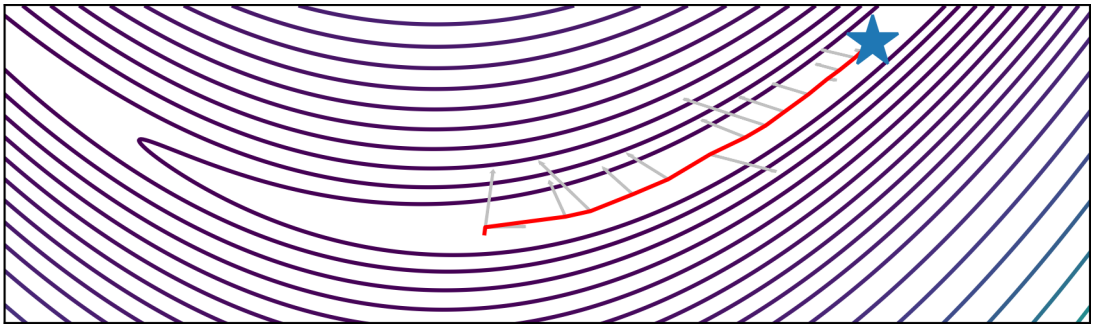
Dict Minimize Documentation

Dict Minimize Team

Oct 29, 2020

CONTENTS

Contents:



CHAPTER

ONE

THE DICT MINIMIZE PACKAGE

Access `scipy` optimizers from your favorite deep learning framework.

1.1 Installation

Only `Python>=3.6` is officially supported, but older versions of Python likely work as well.

The core package itself can be installed with:

```
pip install dict_minimize
```

To also get the dependencies for all the supported frameworks (torch, JAX, tensorflow) in the README install with

```
pip install dict_minimize[framework]
```

See the GitHub, PyPI, and Read the Docs.

1.2 Example Usage

In these examples we optimize a modified Rosenbrock function. However, the arguments have been split into two chunks and stored as two entries in a dictionary. This is to illustrate how this package optimizes *dictionaries* of (tensor) parameters rather then vectors. We also pass in an extra `shift` argument to demonstrate how `minimize` allows extra constant arguments to be passed into the objective.

1.2.1 PyTorch

```
import torch
from dict_minimize.torch_api import minimize

def rosen_obj(params, shift):
    """Based on augmented Rosenbrock from botorch."""
    X, Y = params["x_half_a"], params["x_half_b"]
    X = X - shift
    Y = Y - shift
    obj = 100 * (X[1] - X[0] ** 2) ** 2 + 100 * (Y[1] - Y[0] ** 2) ** 2
    return obj

def d_rosen_obj(params, shift):
    obj = rosen_obj(params, shift=shift)
```

(continues on next page)

(continued from previous page)

```
    da, db = torch.autograd.grad(obj, [params["x_half_a"], params["x_half_b"]])
    d_obj = OrderedDict([("x_half_a", da), ("x_half_b", db)])
    return obj, d_obj

torch.manual_seed(123)

n_a = 2
n_b = 2
shift = -1.0

params = OrderedDict([("x_half_a", torch.randn((n_a,))), ("x_half_b", torch.randn((n_
↪b,)))])

params = minimize(d_rosen_obj, params, args=(shift,), method="L-BFGS-B", options={
↪"disp": True})
```

1.2.2 TensorFlow

```
import tensorflow as tf
from dict_minimize.tensorflow_api import minimize

def rosen_obj(params, shift):
    """Based on augmented Rosenbrock from botorch."""
    X, Y = params["x_half_a"], params["x_half_b"]
    X = X - shift
    Y = Y - shift
    obj = 100 * (X[1] - X[0] ** 2) ** 2 + 100 * (Y[1] - Y[0] ** 2) ** 2
    return obj

def d_rosen_obj(params, shift):
    with tf.GradientTape(persistent=True) as t:
        t.watch(params["x_half_a"])
        t.watch(params["x_half_b"])

        obj = rosen_obj(params, shift=shift)

    da = t.gradient(obj, params["x_half_a"])
    db = t.gradient(obj, params["x_half_b"])
    d_obj = OrderedDict([("x_half_a", da), ("x_half_b", db)])
    del t  # Explicitly drop the reference to the tape
    return obj, d_obj

tf.random.set_seed(123)

n_a = 2
n_b = 2
shift = -1.0

params = OrderedDict([("x_half_a", tf.random.normal((n_a,))), ("x_half_b", tf.random.
↪normal((n_b,)))])

params = minimize(d_rosen_obj, params, args=(shift,), method="L-BFGS-B", options={
↪"disp": True})
```

1.2.3 NumPy

```
import numpy as np
from scipy.optimize import rosen, rosen_der
from dict_minimize.numpy_api import minimize

def rosen_obj(params, shift):
    val = rosen(params["x_half_a"] - shift) + rosen(params["x_half_b"] - shift)

    dval = OrderedDict(
        [
            ("x_half_a", rosen_der(params["x_half_a"] - shift)),
            ("x_half_b", rosen_der(params["x_half_b"] - shift)),
        ]
    )
    return val, dval

np.random.seed(0)

n_a = 3
n_b = 5
shift = -1.0

params = OrderedDict([("x_half_a", np.random.randn(n_a)), ("x_half_b", np.random.
↪randn(n_b))])

params = minimize(rosen_obj, params, args=(shift,), method="L-BFGS-B", options={"disp
↪": True})
```

1.2.4 JAX

```
from jax import random, value_and_grad
import jax.numpy as np
from dict_minimize.jax_api import minimize

def rosen(x):
    r = np.sum(100.0 * (x[1:] - x[:-1] ** 2.0) ** 2.0 + (1 - x[:-1]) ** 2.0, axis=0)
    return r

def rosen_obj(params, shift):
    val = rosen(params["x_half_a"] - shift) + rosen(params["x_half_b"] - shift)
    return val

n_a = 3
n_b = 5
shift = -1.0

# Jax makes it this simple:
d_rosen_obj = value_and_grad(rosen_obj, argnums=0)

# Setup randomness in JAX
key = random.PRNGKey(0)
key, subkey_a = random.split(key)
key, subkey_b = random.split(key)
```

(continues on next page)

(continued from previous page)

```
params = OrderedDict(
    [("x_half_a", random.normal(subkey_a, shape=(n_a,))), ("x_half_b", random.
↪normal(subkey_b, shape=(n_b,)))]
)

params = minimize(d_rosen_obj, params, args=(shift,), method="L-BFGS-B", options={
↪"disp": True})
```

1.3 Contributing

The following instructions have been tested with Python 3.7.4 on Mac OS (10.14.6).

1.3.1 Install in editable mode

First, define the variables for the paths we will use:

```
GIT=/path/to/where/you/put/repos
ENVS=/path/to/where/you/put/virtualenvs
```

Then clone the repo in your git directory `$GIT`:

```
cd $GIT
git clone https://github.com/twitter/dict_minimize.git
```

Inside your virtual environments folder `$ENVS`, make the environment:

```
cd $ENVS
virtualenv dict_minimize --python=python3.7
source $ENVS/dict_minimize/bin/activate
```

Now we can install the pip dependencies. Move back into your git directory and run

```
cd $GIT/dict_minimize
pip install -r requirements/base.txt
pip install -e .  # Install the package itself
```

1.3.2 Contributor tools

First, we need to setup some needed tools:

```
cd $ENVS
virtualenv dict_minimize_tools --python=python3.7
source $ENVS/dict_minimize_tools/bin/activate
pip install -r $GIT/dict_minimize/requirements/tools.txt
```

To install the pre-commit hooks for contributing run (in the `dict_minimize_tools` environment):

```
cd $GIT/dict_minimize
pre-commit install
```

To rebuild the requirements, we can run:

```
cd $GIT/dict_minimize

# Check if there any discrepancies in the .in files
pipreqs dict_minimize/core/ --diff requirements/base.in
pipreqs dict_minimize/ --diff requirements/frameworks.in
pipreqs tests/ --diff requirements/tests.in
pipreqs docs/ --diff requirements/docs.in

# Regenerate the .txt files from .in files
pip-compile-multi --no-upgrade
```

1.3.3 Generating the documentation

First setup the environment for building with `Sphinx`:

```
cd $ENVS
virtualenv dict_minimize_docs --python=python3.7
source $ENVS/dict_minimize_docs/bin/activate
pip install -r $GIT/dict_minimize/requirements/docs.txt
```

Then we can do the build:

```
cd $GIT/dict_minimize/docs
make all
open _build/html/index.html
```

Documentation will be available in all formats in `Makefile`. Use `make html` to only generate the HTML documentation.

1.3.4 Running the tests

The tests for this package can be run with:

```
cd $GIT/dict_minimize
./local_test.sh
```

The script creates an environment using the requirements found in `requirements/test.txt`. A code coverage report will also be produced in `$GIT/dict_minimize/htmlcov/index.html`.

1.3.5 Deployment

The wheel (tar ball) for deployment as a pip installable package can be built using the script:

```
cd $GIT/dict_minimize/
./build_wheel.sh
```

This script will only run if the git repo is clean, i.e., first run `git clean -x -ff -d`.

1.4 Links

The source is hosted on GitHub.

The documentation is hosted at Read the Docs.

Installable from PyPI.

1.5 License

CHAPTER

TWO

CODE OVERVIEW

2.1 PyTorch API

dict_minimize.torch_api.**minimize**(*fun: Callable*, *x0_dict: collections.OrderedDict*, ***, *lb_dict: Optional[collections.OrderedDict] = None*, *ub_dict: Optional[collections.OrderedDict] = None*, *args: Sequence = ()*, *method: Optional[str] = None*, *tol: Optional[float] = None*, *callback: Optional[Callable] = None*, *options: Optional[dict] = None*) → collections.OrderedDict

Minimization of a scalar function with a dictionary of variables as the input. It can interface to functions written for *torch*.

This is a wrapper around *scipy.optimize.minimize*.

Parameters

- **fun** (*callable*) – The objective function to be minimized, in the form of `fun(x, *args) -> (float, OrderedDict)` where *x* is an *OrderedDict* in the format of *x0_dict*, and *args* is a tuple of the fixed parameters needed to completely specify the function. The second returned variable is the gradients. It should be an *OrderedDict* with the same keys and shapes as *x*. The values should be *torch Tensor*.
- **x0_dict** (*OrderedDict*) – Initial guess. Dictionary of variables from variable name to *torch* variables.
- **lb_dict** (*OrderedDict*) – Dictionary with same keys and shapes as *x0_dict* with lower bounds for each variable. Set to *None* in an unconstrained problem.
- **ub_dict** (*OrderedDict*) – Dictionary with same keys and shapes as *x0_dict* with upper bounds for each variable. Set to *None* in an unconstrained problem.
- **args** (*tuple*) – Extra arguments passed to the objective function.
- **method** (*str*) – Type of solver. Should be one of: `CG`, `BFGS`, `L-BFGS-B`, `TNC`, `SLSQP`, or `trust-constr`. If not given, chosen to be one of `BFGS`, `L-BFGS-B`, `SLSQP`, depending if the problem has bounds. Note, only `L-BFGS-B`, `TNC`, `SLSQP` seem to strictly respect the bounds `lb_dict` and `ub_dict`.
- **tol** (*float*) – Tolerance for termination. For detailed control, use solver-specific options.
- **callback** (*callable*) – Called after each iteration. The signature is: `callback(xk)` where *xk* is the current parameter as an *OrderedDict* with the same form as the final solution *x*.
- **options** (*dict*) – A dictionary of solver options. All methods accept the following generic options: maxiter : int Maximum number of iterations to perform. Depending on

the method each iteration may use several function evaluations. disp : bool Set to *True* to print convergence messages.

Returns Final solution found by the optimizer. It has the same keys and shapes as *x0_dict*.

Return type x (OrderedDict)

2.2 TensorFlow API

dict_minimize.tensorflow_api.**minimize**(*fun: Callable*, *x0_dict: collections.OrderedDict*, ***, *lb_dict: Optional[collections.OrderedDict] = None*, *ub_dict: Optional[collections.OrderedDict] = None*, *args: Sequence = ()*, *method: Optional[str] = None*, *tol: Optional[float] = None*, *callback: Optional[Callable] = None*, *options: Optional[dict] = None*) → collections.OrderedDict

Minimization of a scalar function with a dictionary of variables as the input. It can interface to functions written for *tensorflow*.

This is a wrapper around *scipy.optimize.minimize*.

Parameters

- **fun** (*callable*) – The objective function to be minimized, in the form of `fun(x, *args) -> (float, OrderedDict)` where *x* is an *OrderedDict* in the format of *x0_dict*, and *args* is a tuple of the fixed parameters needed to completely specify the function. The second returned variable is the gradients. It should be an *OrderedDict* with the same keys and shapes as *x*. The values should be *tensorflow* variables.
- **x0_dict** (*OrderedDict*) – Initial guess. Dictionary of variables from variable name to *tensorflow* variables.
- **lb_dict** (*OrderedDict*) – Dictionary with same keys and shapes as *x0_dict* with lower bounds for each variable. Set to *None* in an unconstrained problem.
- **ub_dict** (*OrderedDict*) – Dictionary with same keys and shapes as *x0_dict* with upper bounds for each variable. Set to *None* in an unconstrained problem.
- **args** (*tuple*) – Extra arguments passed to the objective function.
- **method** (*str*) – Type of solver. Should be one of: `CG`, `BFGS`, `L-BFGS-B`, `TNC`, `SLSQP`, or `trust-constr`. If not given, chosen to be one of `BFGS`, `L-BFGS-B`, `SLSQP`, depending if the problem has bounds. Note, only `L-BFGS-B`, `TNC`, `SLSQP` seem to strictly respect the bounds `lb_dict` and `ub_dict`.
- **tol** (*float*) – Tolerance for termination. For detailed control, use solver-specific options.
- **callback** (*callable*) – Called after each iteration. The signature is: `callback(xk)` where *xk* is the current parameter as an *OrderedDict* with the same form as the final solution *x*.
- **options** (*dict*) – A dictionary of solver options. All methods accept the following generic options: maxiter : int Maximum number of iterations to perform. Depending on the method each iteration may use several function evaluations. disp : bool Set to *True* to print convergence messages.

Returns Final solution found by the optimizer. It has the same keys and shapes as *x0_dict*.

Return type x (OrderedDict)

2.3 NumPy API

dict_minimize.numpy_api.**minimize**(*fun: Callable*, *x0_dict: collections.OrderedDict*, ***, *lb_dict: Optional[collections.OrderedDict] = None*, *ub_dict: Optional[collections.OrderedDict] = None*, *args: Sequence = ()*, *method: Optional[str] = None*, *tol: Optional[float] = None*, *callback: Optional[Callable] = None*, *options: Optional[dict] = None*) → collections.OrderedDict

Minimization of a scalar function with a dictionary of variables as the input.

This is a wrapper around *scipy.optimize.minimize*.

Parameters

- **fun** (*callable*) – The objective function to be minimized, in the form of `fun(x, *args) -> (float, OrderedDict)` where *x* is an *OrderedDict* in the format of *x0_dict*, and *args* is a tuple of the fixed parameters needed to completely specify the function. The second returned variable is the gradients. It should be an *OrderedDict* with the same keys and shapes as *x*. The values should be *numpy ndarray* variables.
- **x0_dict** (*OrderedDict*) – Initial guess. Dictionary of variables from variable name to *numpy* variables.
- **lb_dict** (*OrderedDict*) – Dictionary with same keys and shapes as *x0_dict* with lower bounds for each variable. Set to *None* in an unconstrained problem.
- **ub_dict** (*OrderedDict*) – Dictionary with same keys and shapes as *x0_dict* with upper bounds for each variable. Set to *None* in an unconstrained problem.
- **args** (*tuple*) – Extra arguments passed to the objective function.
- **method** (*str*) – Type of solver. Should be one of: `CG`, `BFGS`, `L-BFGS-B`, `TNC`, `SLSQP`, or `trust-constr`. If not given, chosen to be one of `BFGS`, `L-BFGS-B`, `SLSQP`, depending if the problem has bounds. Note, only `L-BFGS-B`, `TNC`, `SLSQP` seem to strictly respect the bounds `lb_dict` and `ub_dict`.
- **tol** (*float*) – Tolerance for termination. For detailed control, use solver-specific options.
- **callback** (*callable*) – Called after each iteration. The signature is: `callback(xk)` where *xk* is the current parameter as an *OrderedDict* with the same form as the final solution *x*.
- **options** (*dict*) – A dictionary of solver options. All methods accept the following generic options: maxiter : int Maximum number of iterations to perform. Depending on the method each iteration may use several function evaluations. disp : bool Set to *True* to print convergence messages.

Returns Final solution found by the optimizer. It has the same keys and shapes as *x0_dict*.

Return type x (OrderedDict)

2.4 JAX API

dict_minimize.jax_api.**minimize**(*fun: Callable*, *x0_dict: collections.OrderedDict*, ***, *lb_dict: Optional[collections.OrderedDict] = None*, *ub_dict: Optional[collections.OrderedDict] = None*, *args: Sequence = ()*, *method: Optional[str] = None*, *tol: Optional[float] = None*, *callback: Optional[Callable] = None*, *options: Optional[dict] = None*) → collections.OrderedDict

Minimization of a scalar function with a dictionary of variables as the input. It can interface to functions written for *jax*.

This is a wrapper around *scipy.optimize.minimize*.

Parameters

- **fun** (*callable*) – The objective function to be minimized, in the form of `fun(x, *args) -> (float, OrderedDict)` where *x* is an *OrderedDict* in the format of *x0_dict*, and *args* is a tuple of the fixed parameters needed to completely specify the function. The second returned variable is the gradients. It should be an *OrderedDict* with the same keys and shapes as *x*. The values should be *jax ndarray* variables.
- **x0_dict** (*OrderedDict*) – Initial guess. Dictionary of variables from variable name to *jax ndarray*.
- **lb_dict** (*OrderedDict*) – Dictionary with same keys and shapes as *x0_dict* with lower bounds for each variable. Set to *None* in an unconstrained problem.
- **ub_dict** (*OrderedDict*) – Dictionary with same keys and shapes as *x0_dict* with upper bounds for each variable. Set to *None* in an unconstrained problem.
- **args** (*tuple*) – Extra arguments passed to the objective function.
- **method** (*str*) – Type of solver. Should be one of: `CG`, `BFGS`, `L-BFGS-B`, `TNC`, `SLSQP`, or `trust-constr`. If not given, chosen to be one of `BFGS`, `L-BFGS-B`, `SLSQP`, depending if the problem has bounds. Note, only `L-BFGS-B`, `TNC`, `SLSQP` seem to strictly respect the bounds `lb_dict` and `ub_dict`.
- **tol** (*float*) – Tolerance for termination. For detailed control, use solver-specific options.
- **callback** (*callable*) – Called after each iteration. The signature is: `callback(xk)` where *xk* is the current parameter as an *OrderedDict* with the same form as the final solution *x*.
- **options** (*dict*) – A dictionary of solver options. All methods accept the following generic options: maxiter : int Maximum number of iterations to perform. Depending on the method each iteration may use several function evaluations. disp : bool Set to *True* to print convergence messages.

Returns Final solution found by the optimizer. It has the same keys and shapes as *x0_dict*.

Return type x (OrderedDict)

2.5 Internals

The dict minimize core routines.

Utility for minimization of a scalar function with a dictionary of variables as the input. It can interface to functions written outside of *numpy* (e.g., *tensorflow*, *torch* or *jax*).

This is a wrapper around *scipy.optimize.minimize*.

CHAPTER

THREE

CREDITS

3.1 Development lead

Ryan Turner - Twitter (rdturnermtl)

3.2 Contributors

- Gerard Casas Saez - Twitter (casassg)

PYTHON MODULE INDEX

d

INDEX

D

M