Kaczmarz Algorithms Documentation

Release 0.8.0

Jacob Moorman

Jul 30, 2020

Contents:

CHAPTER 1

Reference

`kaczmarz.Base`(A, b[, x0, tol, maxiter, callback])	A base class for the Kaczmarz algorithm.
`kaczmarz.Cyclic`(*base_args, **base_kwargs)	Cycle through the equations of the system in order, repeatedly.
`kaczmarz.MaxDistance`(A, b[, x0, tol, . . .])	Choose equations which leads to the most progress.

class `kaczmarz.`**`Base`**(*A*, *b*, *x0=None*, *tol=1e-05*, *maxiter=None*, *callback=None*)

A base class for the Kaczmarz algorithm.

This class cannot be instantiated directly. Subclasses should implement `kaczmarz.Base._select_row_index()`. Subclasses will typically be constructed using `kaczmarz.Base.iterates()` or `kaczmarz.Base.solve()`.

Parameters

- **A** (*(m, n) spmatrix or array_like*) – The m-by-n matrix of the linear system.
- **b** (*(m,) or (m, 1) array_like*) – Right hand side of the linear system.
- **x0** (*(n,) or (n, 1) array_like, optional*) – Starting guess for the solution.
- **tol** (*float, optional*) – Tolerance for convergence, `norm(normalized_residual) <= tol`.
- **maxiter** (*int or float, optional*) – Maximum number of iterations.
- **callback** (*function, optional*) – User-supplied function to call after each iteration. It is called as `callback(xk)`, where xk is the current solution vector.

Notes

There may be additional parameters not listed above depending on the selection strategy subclass.

`_select_row_index`(*xk*)

Select a row to use for the next Kaczmarz update.

Parameters **xk** (*(n,) array*) – The current Kaczmarz iterate.

Returns **ik** – The index of the next row to use.

Return type int

ik

The index of the row used on the most recent iteration.

Takes the value -1 if a projection was not performed at iteration `k`.

Type int

xk

The most recent iterate.

The shape will match that of `x0` if provided, or `b` otherwise.

Type (n,) or (n, 1) array

classmethod iterates(**base_args*, ***base_kwargs*)

Get the Kaczmarz iterates.

Note: This method takes the same parameters as `kaczmarz.Base` or the subclass from which it is called. For example, `kaczmarz.Cyclic.iterates()` takes the same arguments as `kaczmarz.Cyclic`.

Parameters

- **base_args** (*tuple*) – Positional arguments for `kaczmarz.Base` constructor or the subclass in use.
- **base_kwargs** (*dict*) – Keyword arguments for `kaczmarz.Base` constructor or the subclass in use.

Returns **iterates** – An iterable of the Kaczmarz iterates. The shapes will match that of `x0` if provided, or `b` otherwise.

Return type iterable((n,) or (n, 1) array)

classmethod solve(**base_args*, ***base_kwargs*)

Solve a linear system of equations using the Kaczmarz algorithm.

Note: This method takes the same parameters as `kaczmarz.Base` or the subclass from which it is called. For example, `kaczmarz.Cyclic.solve()` takes the same arguments as `kaczmarz.Cyclic`.

Parameters

- **base_args** (*tuple*) – Positional arguments for `kaczmarz.Base` constructor or the subclass in use.
- **base_kwargs** (*dict*) – Keyword arguments for `kaczmarz.Base` constructor or the subclass in use.

Returns **x** – The solution to the system `A @ x = b`. The shape will match that of `x0` if provided, or `b` otherwise.

Return type (n,) or (n, 1) array

class `kaczmarz.`**`Cyclic`**(**base_args*, ***base_kwargs*)

Bases: `kaczmarz._abc.Base`

Cycle through the equations of the system in order, repeatedly.

References

1. S. Kaczmarz. "Angenäherte Auflösung von Systemen linearer Gleichungen." *Bulletin International de l'Académie Polonaise des Sciences et des Lettres. Classe des Sciences Mathématiques et Naturelles. Série A, Sciences Mathématiques*, 35, 335–357, 1937

class `kaczmarz.`**`MaxDistance`**(*A*, *b*, *x0=None*, *tol=1e-05*, *maxiter=None*, *callback=None*)

Bases: `kaczmarz._abc.Base`

Choose equations which leads to the most progress.

This selection strategy is also known as *Motzkin's method*.

References

1. T. S. Motzkin and I. J. Schoenberg. "The relaxation method for linear inequalities." *Canadian Journal of Mathematics*, 6:393–404, 1954.

CHAPTER 2

Kaczmarz Algorithms

Variants of the Kaczmarz algorithm for solving linear systems in Python.

2.1 Installation

To install Kaczmarz Algorithms, run this command in your terminal:

```
$ pip install -U kaczmarz-algorithms
```

This is the preferred method to install Kaczmarz Algorithms, as it will always install the most recent stable release.

If you don't have pip installed, these installation instructions can guide you through the process.

2.2 Usage

First, import the `kaczmarz` package.

```
>>> import kaczmarz
```

2.2.1 Solving a system of equations

To solve the system of equations `3 * x0 + x1 = 9` and `x0 + 2 * x1 = 8` using the Kaczmarz algorithm with the cyclic selection rule, use the `kaczmarz.Cyclic.solve()` function.

```
>>> A = [[3, 1],
...      [1, 2]]
>>> b = [9, 8]
>>> x = kaczmarz.Cyclic.solve(A, b)
>>> x
array([2., 3.])
```

2.2.2 Inspecting the Kaczmarz iterates

To access the iterates of the Kaczmarz algorithm with the cyclic selection rule, use the `kaczmarz.Cyclic.iterates()` function.

```
>>> A = [[1, 0, 0],
...      [0, 1, 0],
...      [0, 0, 1]]
>>> b = [1, 1, 1]
>>> x0 = [0, 0, 0]  # Initial iterate
>>> for xk in kaczmarz.Cyclic.iterates(A, b, x0):
...     xk
array([0., 0., 0.])
array([1., 0., 0.])
array([1., 1., 0.])
array([1., 1., 1.])
```

2.2.3 Inspecting the rows/equations used

To access the row index used at each iteration of the Kaczmarz algorithm, use the `ik` attribute of the iterates. For example,

```
>>> iterates = kaczmarz.Cyclic.iterates(A, b, x0)
>>> for xk in iterates:
...     print("Row used:", iterates.ik)
Row used: -1
Row used: 0
Row used: 1
Row used: 2
```

The initial value of `iterates.ik` is `-1`, since no projections have been performed yet at the start of the algorithm.

2.2.4 Optional arguments

The `solve()` and `iterates()` functions take optional arguments of `maxiter` and `tol` to specify a limit on the number of iterations and the desired accuracy of the solution respectively.

2.2.5 Creating your own selection strategy

To implement a selection strategy of your own, inherit from `kaczmarz.Base` and implement the `_select_row_index()` method. For example, to implement a strategy which uses of the equations of your system in reverse cyclic order:

```
>>> class ReverseCyclic(kaczmarz.Base):
...     def __init__(self, A, *args, **kwargs):
...         super().__init__(A, *args, **kwargs)
...         self.n_rows = len(A)
...         self.row_index = None
...
...     def _select_row_index(self, xk):
...         if self.row_index is None:
...             self.row_index = self.n_rows
...         self.row_index = (self.row_index - 1) % self.n_rows
...         return self.row_index
```

Your new class will inherit `solve()` and `iterates()` class methods which work the same way as `kaczmarz.Cyclic.solve()` and `kaczmarz.Cyclic.iterates()` described above.

```
>>> iterates = ReverseCyclic.iterates(A, b, x0)
>>> for xk in iterates:
...     print("Row used:", iterates.ik)
...     print("Iterate:", xk)
Row used: -1
Iterate: [0. 0. 0.]
Row used: 2
Iterate: [0. 0. 1.]
Row used: 1
Iterate: [0. 1. 1.]
Row used: 0
Iterate: [1. 1. 1.]
```

For information about the optional arguments of `solve()` and `iterates()`, as well as the other selection strategies available other than `Cyclic`, see readthedocs.io.

2.3 Citing

If you use our code in an academic setting, please consider citing our code. You can find the appropriate DOI for whichever version you are using on zenodo.org.

2.4 Development

See CONTRIBUTING.md for information related to developing the code.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Index

Symbols

B

C

I

M

S

X