4.3.4 Problem Set: Implementing and Testing Gradient Descent

In the first part of this problem set, you will implement in `gd.py` the gradient descent algorithm for a generic problem (not specific to the cell tower base station optimization). The file `gd_tests.py` contains a couple of objective functions on which you can test your implementation. You will also complete `gd_tests.py` to plot contours and the gradient descent history for an objective function that models a single cell tower.

1. Complete the main loop of the `gradient_descent` function in `gd.py`. When you have completed the function, test it by running `run_test0` in `gd_tests.py`. The objective function for this test case is:

$$J_0(x) = \left(x - \frac{1}{2}\right)^2 \tag{4.46}$$

and the calculation of J_0 and its gradient $\mathrm{d}J_0/\mathrm{d}x$ have already been fully implemented for you in the `J0` function in `gd_tests.py`. If your `gradient_descent` function is correct, you should see identical output as below when you run `gd_tests.py`:

```
Running test0
Iter=0, J = 1.00e+00
Iter=1: J = 6.40e-01, dJ = -3.60e-01, max dx = 2.00e-01 for state i=0
Iter=2: J = 4.10e-01, dJ = -2.30e-01, max dx = 1.60e-01 for state i=0
Iter=3: J = 2.62e-01, dJ = -1.47e-01, max dx = 1.28e-01 for state i=0
Iter=4: J = 1.68e-01, dJ = -9.44e-02, max dx = 1.02e-01 for state i=0
Iter=5: J = 1.07e-01, dJ = -6.04e-02, max dx = 8.19e-02 for state i=0
...
Iter=39: J = 2.76e-08, dJ = -1.55e-08, max dx = 4.15e-05 for state i=0
Iter=40: J = 1.77e-08, dJ = -9.94e-09, max dx = 3.32e-05 for state i=0
In run_test0: xopt = 5.00e-01, Jmin = 1.77e-08
```

Note that the printouts for each iteration are activated by setting the `verbose` parameter to `True`. You must implement this printout – conditioned on `verbose` – within the gradient descent loop. The specific meaning of `dJ` and `max dx` are:

- `dJ` is the change in J from the last iteration, i.e., `dJ =` $J^n - J^{n-1}$
- `max dx` is the largest magnitude change in any component of x. You'll need to calculate $x^n - x^{n-1}$ and then find the component with the largest magnitude.

Discussions

All posts sorted by recent activity

Contour plot and f values Hi all, I have a
Wemans
1 4

- The index of that component with the largest magnitude change should also be reported (this is the `for state i=0` part). Since there is only one state for J_0, then in this case the index will always be `i=0`.

Note: There are many ways to find where the maximum component changes, but for a NumPy approach, consider using `np.argmax` or related functions.

2. Next, you will complete the `J1` and `run_test1` functions in `gd_tests.py`. The `J1` function calculates a two-dimensional objective function J_1:

$$J_1\left(x,y\right) = -\frac{1}{1+\left(x-x^u\right)^2+\left(y-y^u\right)^2} \tag{4.47}$$

< Previous Next >

the negative signal power that a user located at (x^u, y^u) receives from a tower located at (x, y). (We use the *negative*

```
iter=1: J = -8.04e-01, dJ = -4.19e-02, max dx = 5.80e-
```