1 Experiment 1

This experiment is aimed to compare the influence from number of dimensions and conditions valuse to number of iterations between gradient descent method and conjugate gradient method. Settings of experiment in this part:

- randomseed: 45
- number of dimensions: [10, 100, 1000]
- value of condition: [100, 200,..., 1900, 2000]. In total 20.
- for every n, 10 funcs are randomly generated by k.
- for every generated function, 5 start points are ramdomly chosen and iteration for the funcion is their mean iterations.
- the curve with deep color is the average curve of all launches with an n.

fig.1 shows the result of gradient descent method.

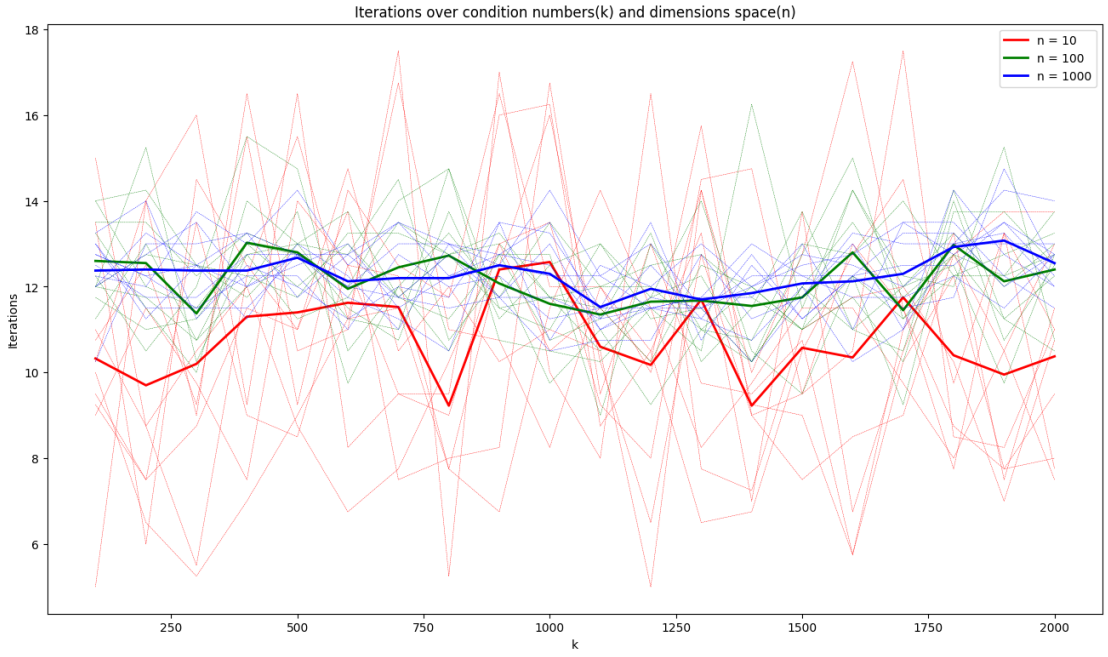

fig.1

fig.2 shows the result of method conjugate gradient method.

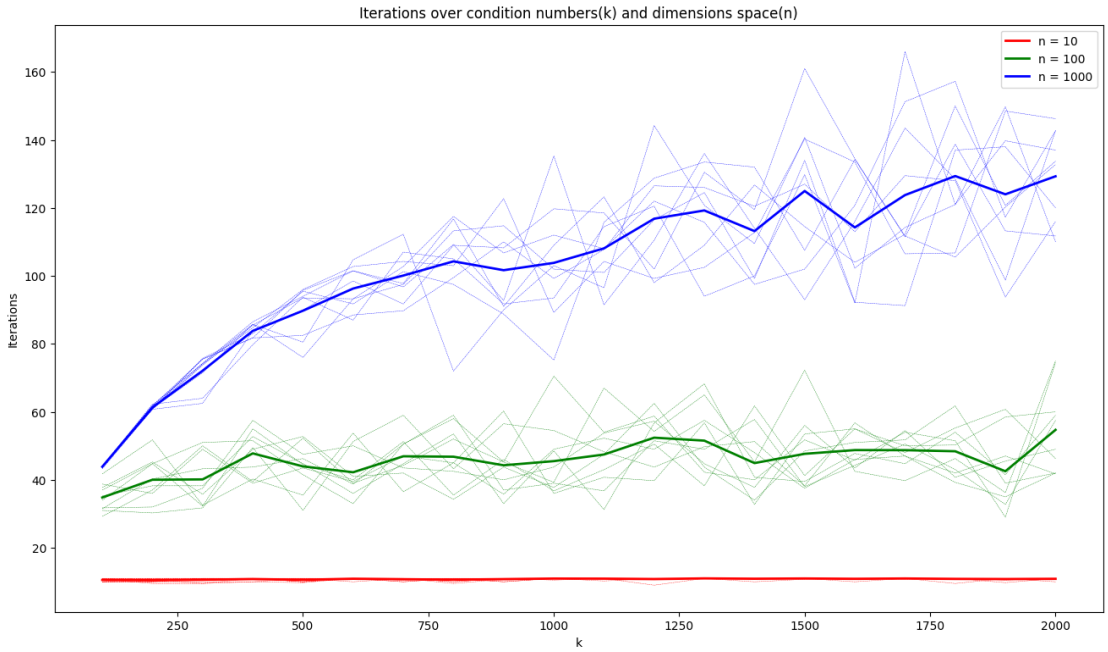

fig.2

For gradient descent method, as before, the number of iterations are not much influnced by k and n. But for conjugate gradient method, with larger n the number of iterations are obviously raising. For n=10, the number of iterations remains relatively constant even as k increases. For n=100, the number of iterations fluctuate between 40 and 50 as k increases. For n = 1000, it's very obvious that number of iterations increases as k increases. Therefore, unlike the gradient descent method, the conjugate gradient method's number of iterations increases as both n and k increase.

2 Experiment 2

The experiment shows the descending process of $\frac{||\nabla f(x_k)||_2^2}{||\nabla f(x_0)||_2^2}$ by number of iterations and time with differenct sizes of history cahce for lbfgs method. Settings of experiment in this part:

- history cache size - n: [1, 5, 10, 25, 50, 75, 100]

fig.3 shows the result of descending process of $\frac{||\nabla f(x_k)||_2^2}{||\nabla f(x_0)||_2^2}$ by number of iterasions.

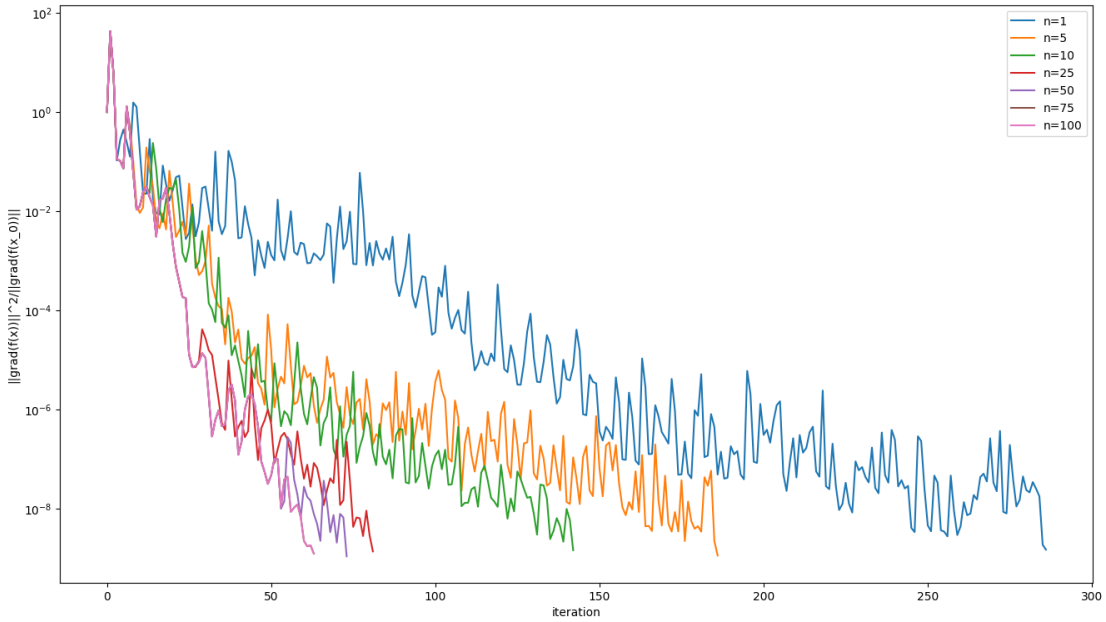

fig.3

fig.4 shows the result of descending process of $\frac{||\nabla f(x_k)||_2^2}{||\nabla f(x_0)||_2^2}$ by time.

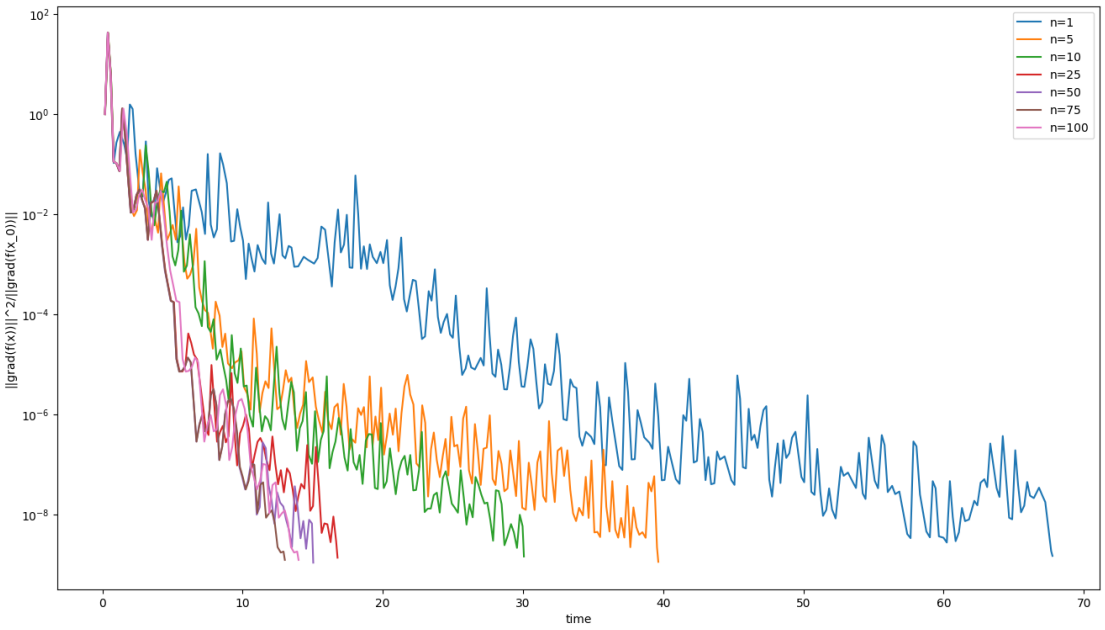

fig.4

The spatial complexity of lbfgs method is $O(n \times m)$. The time complexity of lbfgs method is $O(n \times m)$

- m - dimension of space
- n - size of hostory cache

From the results of both fig.3 and fig.4, we can conclude that with a larger size of the history cache, the function descends faster. However, beyond n=25, these curves exhibit nearly the same descending trend. Even in terms of descending time, the curve for n=100 is not the fastest.

3 Experiment3

In the experiment, for every real dataset 3 implemented methods are used to do linear regression. For each dataset, 3 graphs are created - $f(x_k)$ by number of iterations, $f(x_k)$ by time, $\frac{||\nabla f(x_k)||_2^2}{||\nabla f(x_0)||_2^2}$ by time. Methods used: gradient descent(gd), hessian free newton(hfn), lbfgs. Dataset names: gisette, w8a, news20.binary, real-sim, rcv1_train.binary .

fig.5 shows the result of dataset gisette

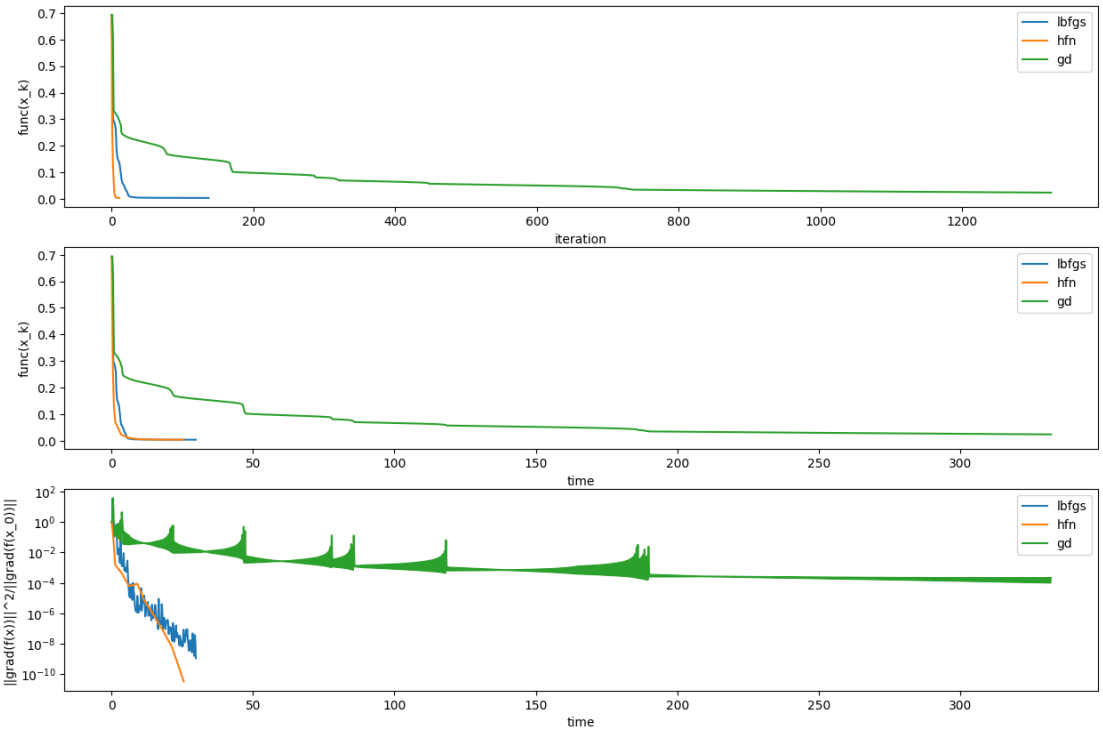

fig.5

fig.5 shows the result of dataset w8a

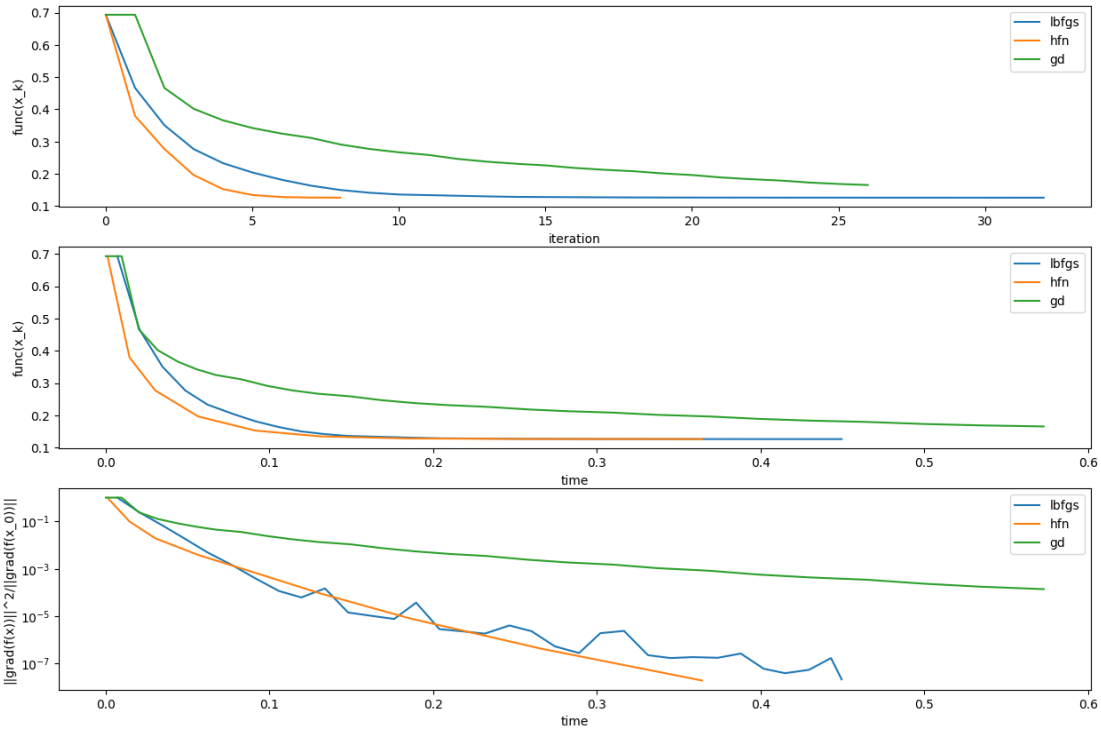

fig.6

fig.5 shows the result of dataset news20.binary

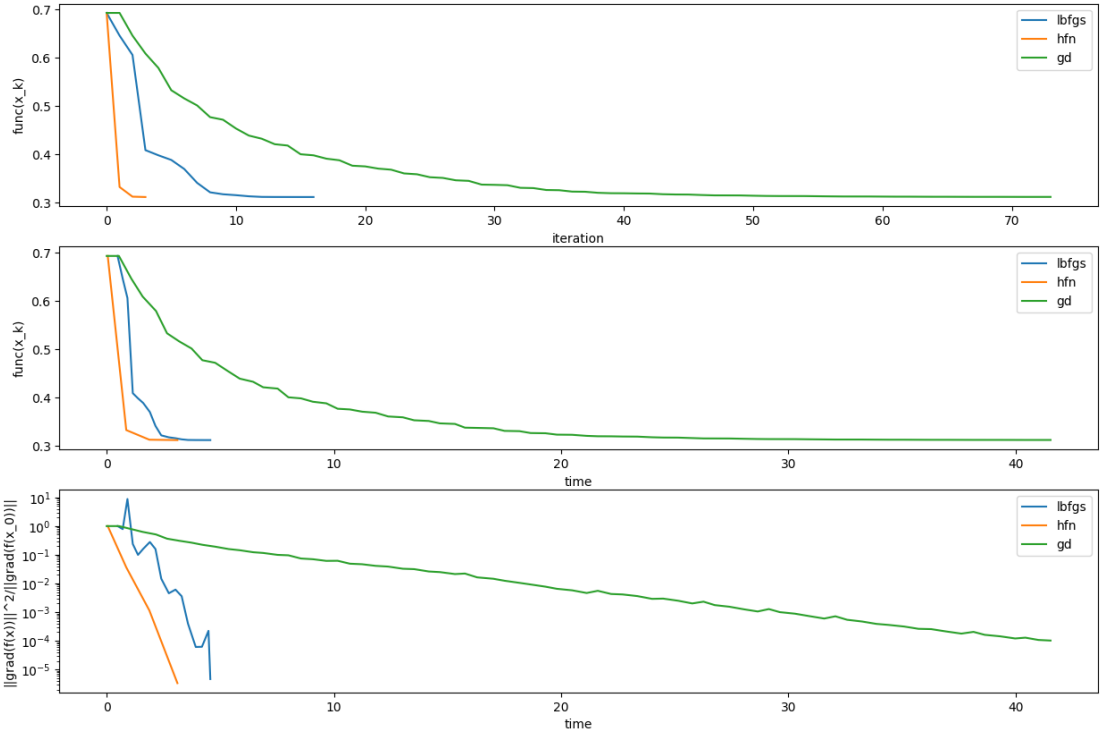

fig.7

fig.5 shows the result of dataset real-sim

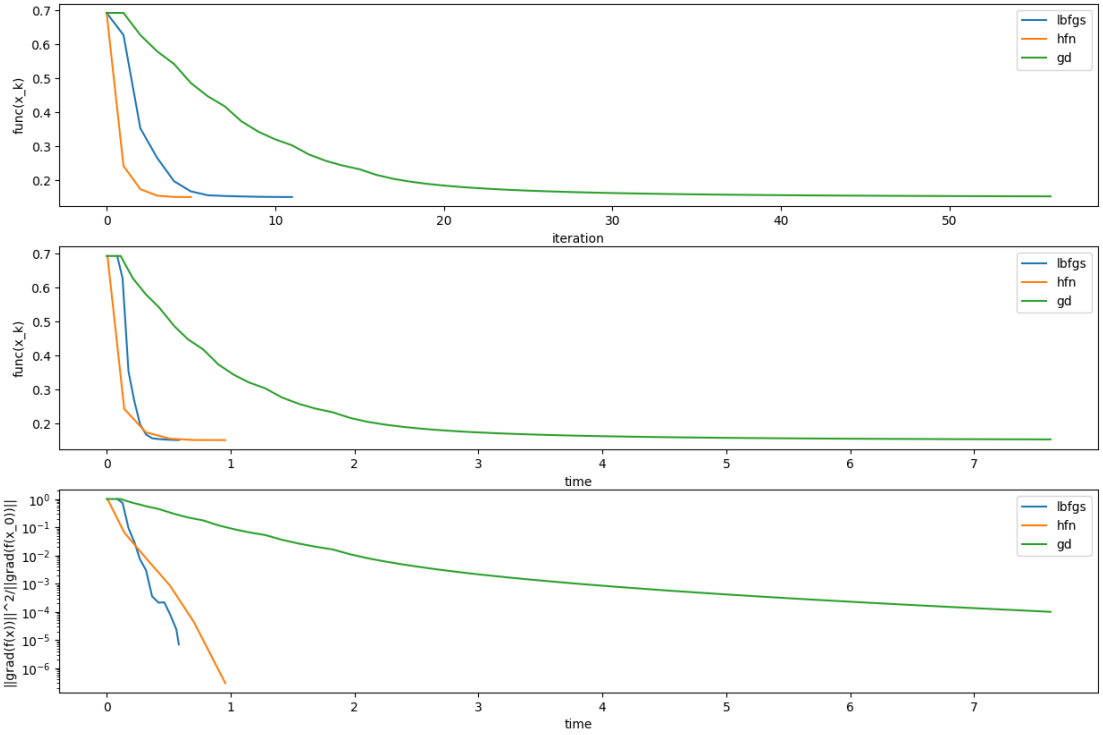

fig.8

fig.5 shows the result of dataset rcv1_train.binary

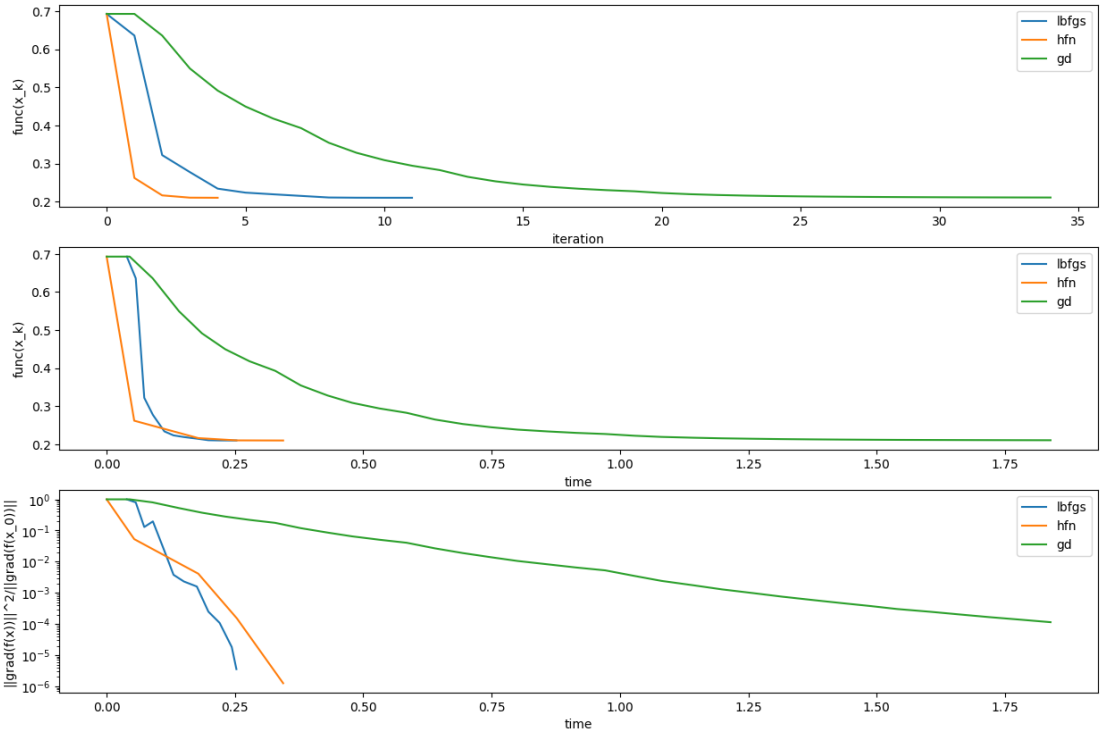

fig.9

The dataset with corresponding best method:

- gisette - hfn
- w8a - hfn
- news20.binary - hfn
- real-sim - lbfgs
- rcv1_train.binary - lbfgs

It can be seen from the results that the quasi-Newton method undoubtedly defeated the gradient descent method in terms of time and number of iterations no matter using lbfgs or hfn. Whether choose hfn or lbfgs is based on the dataset itself, sometimes lbfgs is better sometimes hfn is better and they don't have much difference in terms of descending time. For number of iterations hfn gets better result.