Homework1: Regression and Classification

Mingyi Xue*

April 30, 2018

1 Problem1 Regression

The loss function is defined as follows,

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} \{ \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x_i^T}\boldsymbol{\omega} - y_i)^2 + \frac{\lambda}{2} \|\boldsymbol{\omega}\|^2 \} \tag{1}$$

Derive the gradient of equation(1),

$$\frac{\partial f(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} = \frac{2}{n} \boldsymbol{X^T}(\boldsymbol{X}\boldsymbol{\omega} - Y) + \lambda \boldsymbol{\omega} \tag{2}$$

Algorithm 1 Gradient Descent with Fixed Step Size

Input: η: step size, ϵ: stopping condition, $\boldsymbol{\omega_0}$: initial solution , $iter$: number of iteration
Output: $\boldsymbol{\omega}$

```
ω ← ω_0
r_0 ← ‖∇f(ω)‖
for i = 0 → iter − 1 do
    g = ∇f(ω)
    if ‖g‖ ≤ εr_0 then
        break
    end if
    ω ← ω_0 − ηg
end for
```

Problem 1.1

pretreatment

We apply the gradient descent algorithm to dataset "cpusmall", normalizing dataset so that the value of loss function and gradient won't exceed the maximum of real number in python.

parameters

"cpusmall" is a dataset of 8192 sample and 12 features. $\boldsymbol{\omega_0}$ is set to a column of 12 zeros in each outer interation for a specific stepsize. Other parameters are as follows,

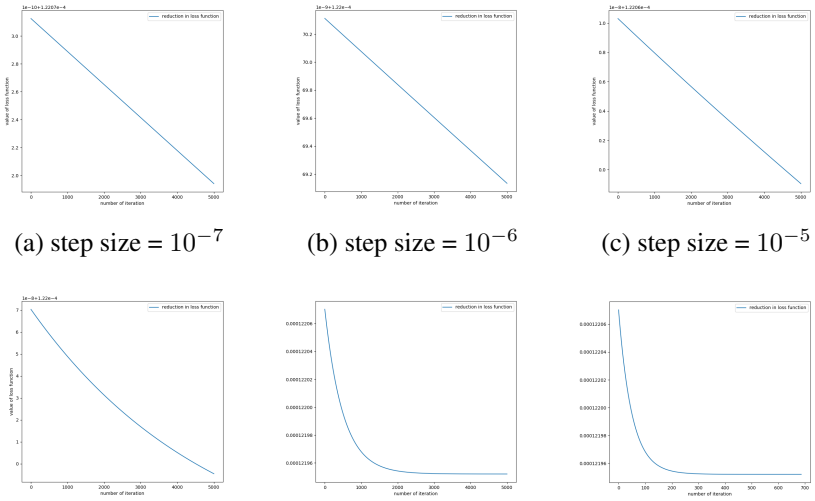
*GSP student from Nanjing University

Table 1: default parameters

name	value
λ	1
ϵ	0.001
# of iterations	5000

results

The final output records the value of loss function $f(\boldsymbol{\omega})$ in the last iteration, whether $\boldsymbol{\omega}$ has converged or not within 5000 iterations and the plot of reduction in the value of loss function. The results are as follows,



(a) step size = 10^{-7} (b) step size = 10^{-6} (c) step size = 10^{-5}

(d) step size = 10^{-4} (e) step size = 10^{-3} (f) step size = 10^{-2}

Figure 1: results of different step size

Table 2: results of different step size

step size η	final value of $f(\boldsymbol{\omega})$	converge
10^{-7}	1.22070×10^{-4}	not yet
10^{-6}	1.22069×10^{-4}	not yet
10^{-5}	1.22059×10^{-4}	not yet
10^{-4}	1.21996×10^{-4}	not yet
10^{-3}	1.21952×10^{-4}	not yet
10^{-2}	1.21952×10^{-4}	yes

Table 3: three best step size and its weight vector

	10^{-4}	10^{-3}	10^{-2}
ω_1	7.49×10^{-5}	7.45×10^{-5}	2.96×10^{-5}
ω_2	9.02×10^{-5}	8.97×10^{-5}	3.56×10^{-5}
ω_3	1.84×10^{-4}	1.83×10^{-4}	7.27×10^{-5}
ω_4	1.61×10^{-4}	1.60×10^{-4}	6.35×10^{-5}
ω_5	1.52×10^{-4}	1.51×10^{-4}	6.00×10^{-5}
ω_6	1.29×10^{-4}	1.28×10^{-4}	5.08×10^{-5}
ω_7	9.91×10^{-5}	9.85×10^{-5}	3.91×10^{-5}
ω_8	1.38×10^{-4}	1.37×10^{-4}	5.44×10^{-5}
ω_9	1.21×10^{-4}	1.21×10^{-4}	4.79×10^{-5}
ω_{10}	4.23×10^{-6}	4.20×10^{-6}	1.68×10^{-6}
ω_{11}	1.49×10^{-4}	1.48×10^{-4}	5.89×10^{-5}
ω_{12}	2.38×10^{-4}	2.36×10^{-4}	9.36×10^{-5}

conclusion

- When the step size is too small, the speed of reduction in loss function will be slow, and $\boldsymbol{\omega}$ more iterations and time are needed to converge.
- When the step size is very large, $\boldsymbol{\omega}$ may diverge. If we do not normalize the dataset at the beginning, $10^{-7} \sim 10^{-2}$ will be too large for the program and a step size smaller than 10^{-13} is doable.
- As a result, choosing the appropriate step size or learning rate is of significantly importance in machine learning. 10^{-3} and 10^{-2} are acceptable step sizes for this problem.

Problem 1.2

pretreatment

Normalize the dataset "cpusmall" as Problem 1.1.

Apply cross validation on the dataset. Zip matrix $\boldsymbol{X}$ with $\boldsymbol{Y}$ as a large matrix $\boldsymbol{data}$, shuffle $\boldsymbol{data}$ and split it evenly into 5 folders, treating 4 folders of which as training set and the rest one as test set.

Run gradient descent algorithm on training set and calculate MSE of the test set.

parameters

Table 4: default parameters

name	value
λ	1
step size η	10^{-2}
# of folders	5
# of iterations	1000

$$
\begin{aligned}
MSE &= \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \{(\boldsymbol{x_i^T}\boldsymbol{\omega} - y_i)^2\} \\
&= \frac{\|\boldsymbol{X}\boldsymbol{\omega} - Y\|^2}{n_{test}}
\end{aligned} \tag{3}
$$

results

Table 5: MSE

test folder	value of MSE
1	1.21159×10^{-4}
2	1.21787×10^{-4}
3	1.22295×10^{-4}
4	1.21572×10^{-4}
5	1.22356×10^{-4}
$mean$	1.21834×10^{-4}

Problem 1.3

parameters

Table 6: default parameters

name	value
ϵ	0.001
# of iterations	2000

results

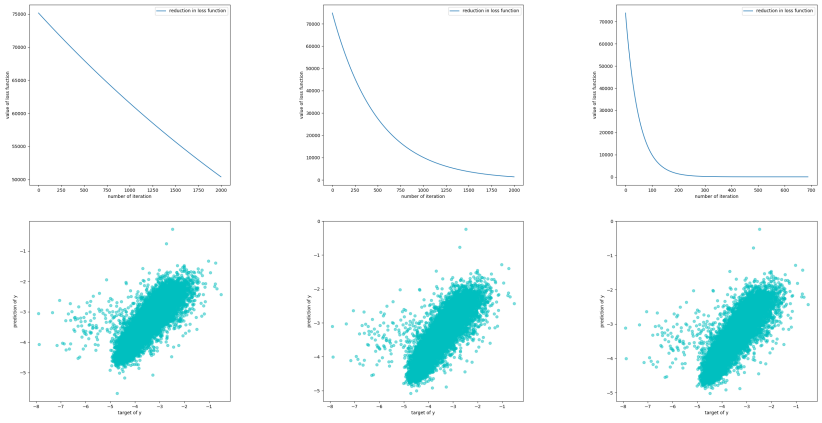
The final output records the value of test MSE, whether $\boldsymbol{\omega}$ has converged or not within 2000 iterations, the plot of reduction in the value of loss function and the scatter plot of estimated $\hat{y}$ vs. target y.

Results are as follows,

Table 7: results of different step size

stepsize	value of test MSE	final value of $f(\boldsymbol{\omega})$	converge
10^{-4}	0.156556	50392.03	not yet
10^{-3}	0.150899	1370.50	not yet
10^{-2*}	0.150888	0.7328	yes
10^{-1}	–	–	diverge

* This step size is best.



(a) step size = 10^{-4} (b) step size = 10^{-3} (c) step size = 10^{-2}

2 Problem2 Classification

Problem 2.1

The loss function for logistic regression is defined as follows,

$$\boldsymbol{\omega^*} = \arg\min_{\boldsymbol{\omega}}\{\frac{1}{n}\sum_{i=1}^{n}\log(1+\mathrm{e}^{-y_i\boldsymbol{\omega^T x_i}}) + \frac{\lambda}{2}\|\boldsymbol{\omega}\|^2\} \tag{4}$$

Derive the gradient of equation(4),

$$\frac{\partial f(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} = \frac{1}{n}\sum_{i=1}^{n}\frac{-y_i}{1+\mathrm{e}^{-y_i\boldsymbol{\omega^T x_i}}}\boldsymbol{x_i} + \lambda\boldsymbol{\omega} \tag{5}$$

In python, we can use numpy.ndarray broadcast to simplify calculation and save time,

$$\begin{aligned}\boldsymbol{K} &= np.array(\boldsymbol{X}@\boldsymbol{w}) * np.array(\boldsymbol{Y})\\ \boldsymbol{M} &= -np.array(\boldsymbol{Y}) * \frac{1}{1+\mathrm{e}^{\boldsymbol{K}}}\\ \frac{\partial f(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} &= \frac{1}{n}\boldsymbol{X^T M} + \lambda\boldsymbol{\omega}\end{aligned} \tag{6}$$

Problem 2.2

The initial $\boldsymbol{\omega_0}$ is randomly normal distributed.

Table 8: default parameters

dataset	accuracy
training set	69.92%
test set	68.28%